CNN for CV

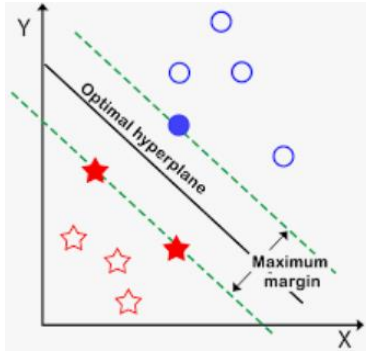
AI for CV Group
2020

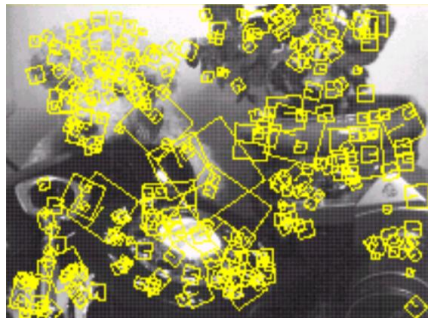
Week 10-11.
Detection

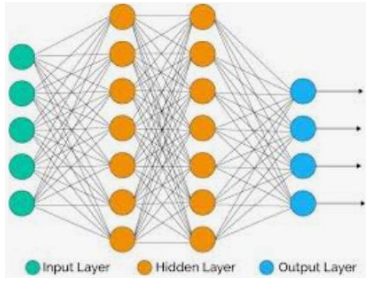
Contents:

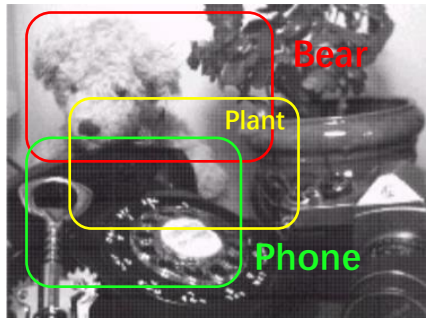
I. Two Stage Detection

Start from:









Input → Feature → Classic ML Tools → Target

Get Manually → Choose Carefully → Use Hopelessly

End up with:

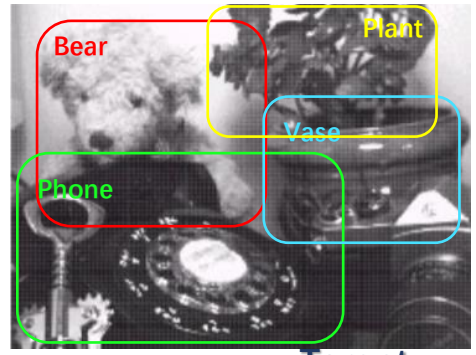
Input

Put In

CNN

Only Focus
On
This Part

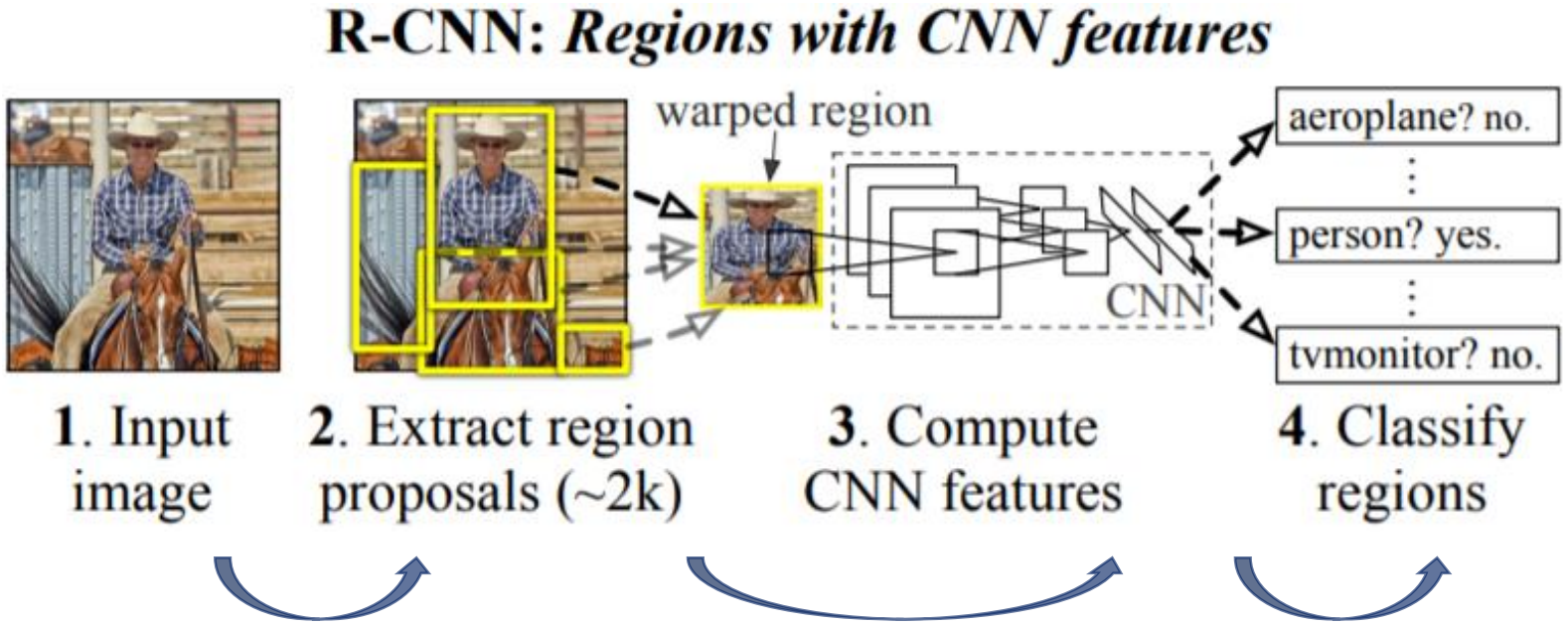
Use It



Target

I. Two Stage Detection

A. First Trial - RCNN [2014 Ross]:



A1. Region Proposal

A2. Feature Extraction

A3. Classification + Detection

A.1 Region Proposal

A.2 Feature Extraction

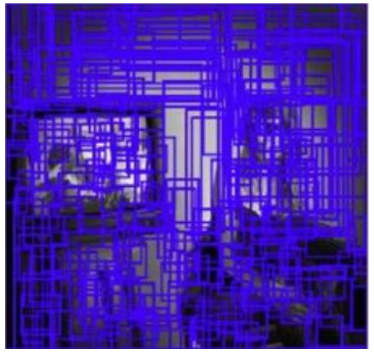
A.3 Cls + Det

I. Two Stage Detection

A. First Trial - RCNN [2014 Ross]:

A1. Region Proposal: Selective Search

Input



Get Automatically

Warp To

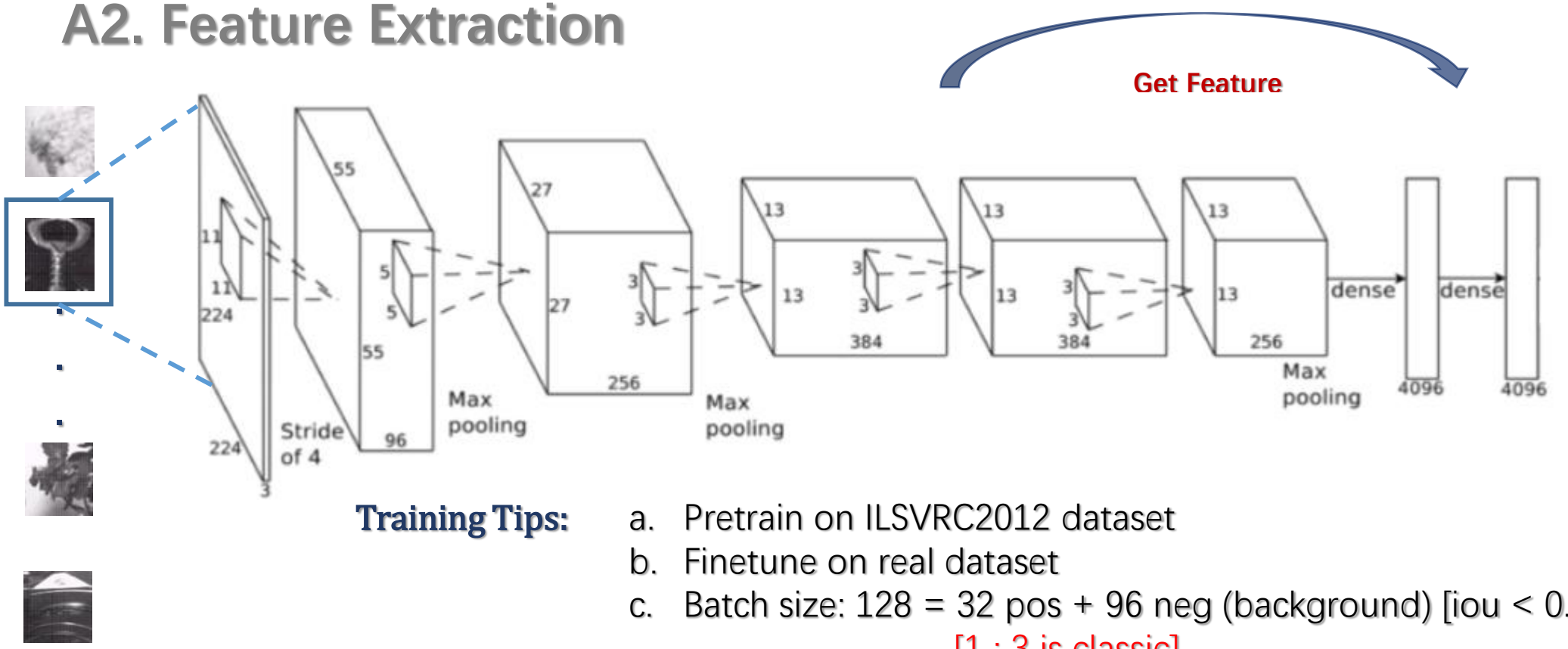
Fixed Size

Nearly 2000

I. Two Stage Detection

A. First Trial - RCNN [2014 Ross]:

A2. Feature Extraction



Training Tips:

a. Pretrain on ILSVRC2012 dataset
b. Finetune on real dataset
c. Batch size: 128 = 32 pos + 96 neg (background) [iou < 0.5]
[1 : 3 is classic]

I. Two Stage Detection

A. First Trial - RCNN [2014 Ross]:

A3. SVM Classification + NMS + BBox Regression

a. SVM Classification

- Do classification for each class
- Pos: Ground truth
  Neg: iou < 0.3 [others ignored]

[Different from CNN part]

Q: Why iou threshold is different here comparing to training AlexNet?
A: We need more data when training CNN. But here more data may be harmful.

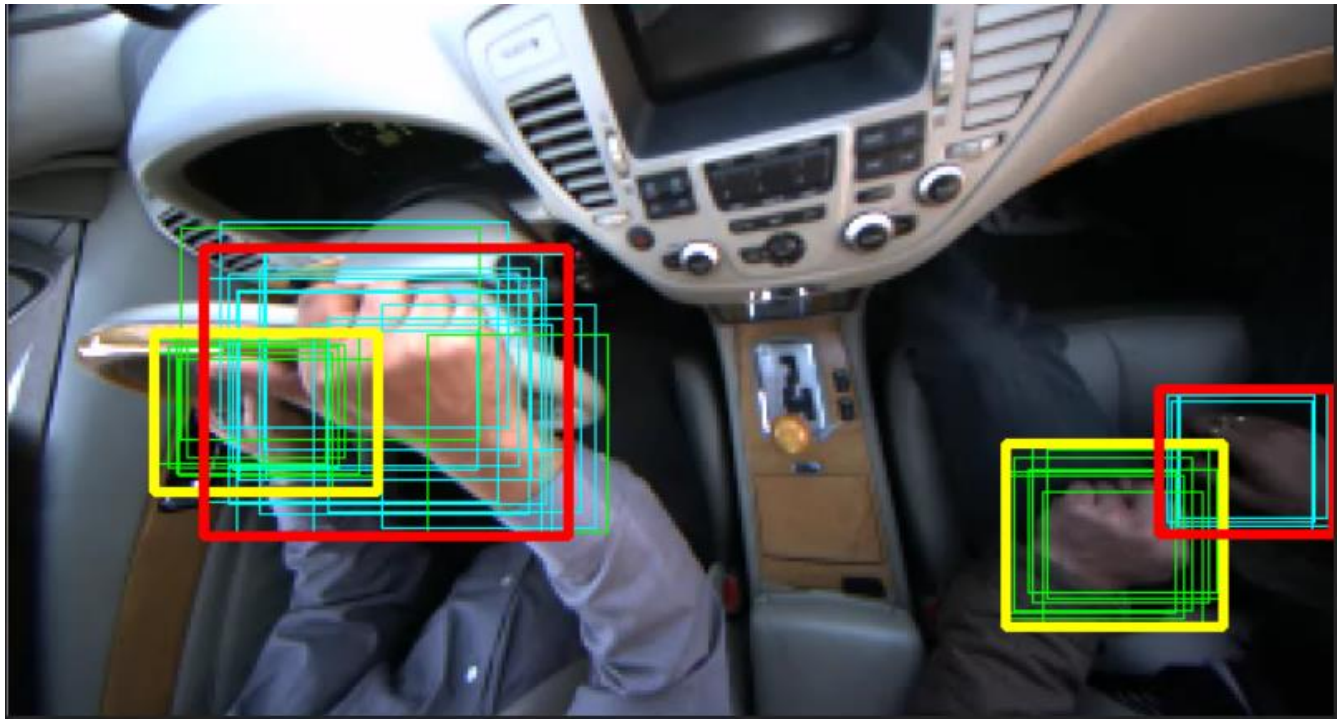
Q: Why use SVM? AlexNet has softmax at last, why not use directly?
A: AlexNet focuses on classification. Not accurate on localization
Use hard negative mining when using SVM

I. Two Stage Detection

A. First Trial - RCNN [2014 Ross]:

A3. SVM Classification + NMS + BBox Regression

b. NMS: Non-Maximum Suppression

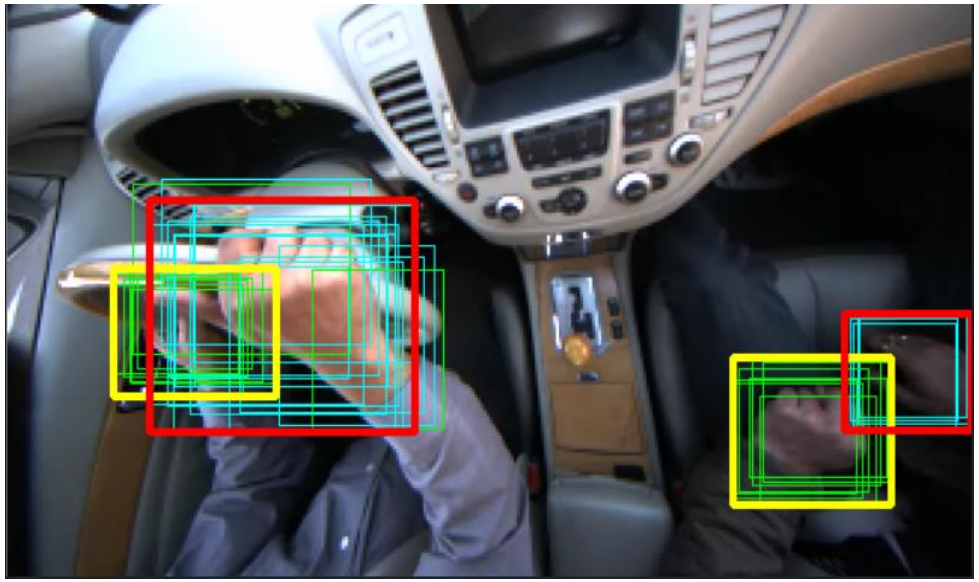


I. Two Stage Detection

A. First Trial - RCNN [2014 Ross]:

A3. SVM Classification + NMS + BBox Regression

b. NMS: Non-Maximum Suppression



- Sort bbox according to score [here class score]
- Get the 1st one. Remove those which has higher iou with it
- Then repeat the last step until there's no bbox

I. Two Stage Detection

A. First Trial - RCNN [2014 Ross]:

A3. SVM Classification + NMS + BBox Regression

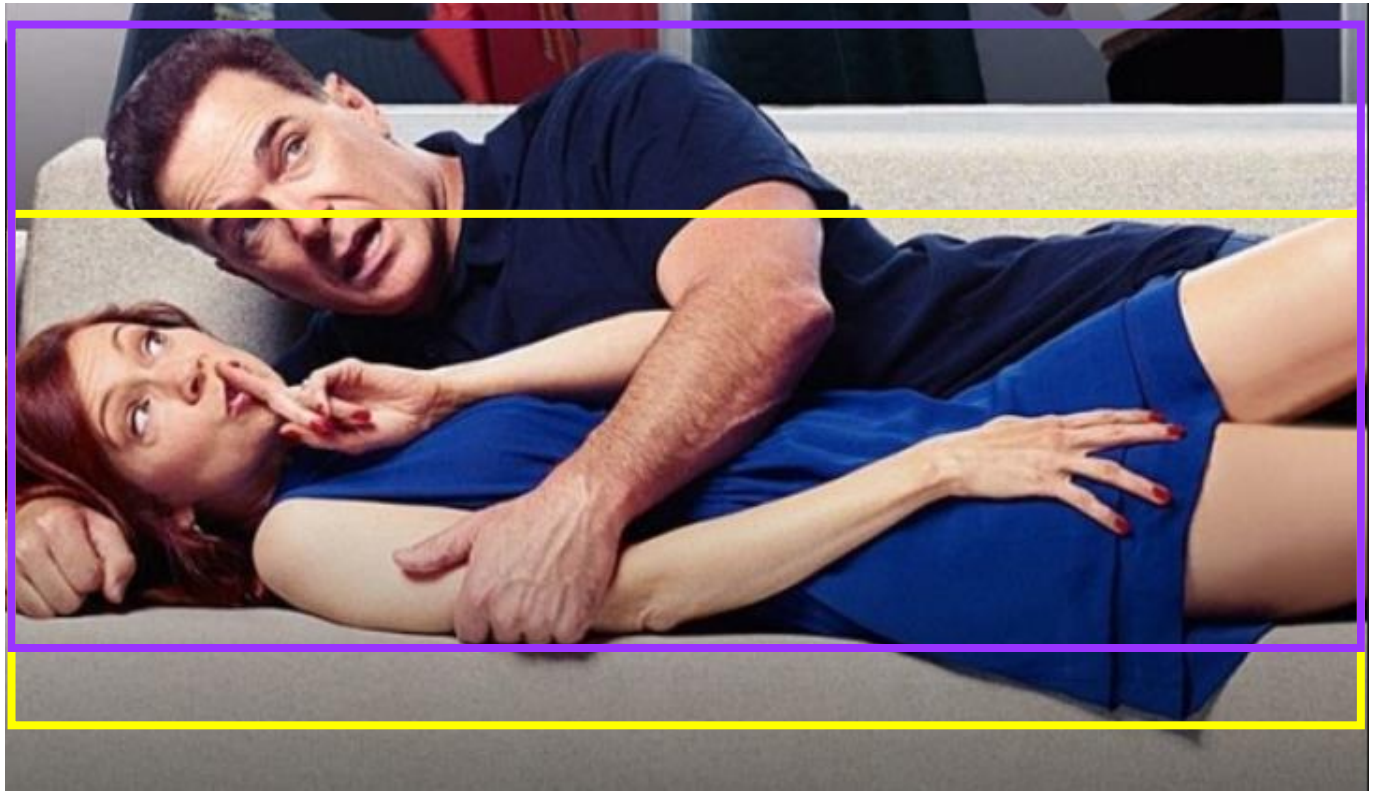
c. Bbox Regression

- Regression targets of (dx, dy, dw, dh)
- Coordinates need to be normalized
- We'll discuss in detail later

I. Two Stage Detection

A. First Trial - RCNN [2014 Ross]:

A*. Rethinking about NMS



I. Two Stage Detection

A. First Trial - RCNN [2014 Ross]:

A*. Rethinking about NMS

- If ▭ has higher score, we'll lose ▭
- If ▭ has higher score, we'll loss ▭
- That's awkward.

Let's Correct It!

I. Two Stage Detection

A. First Trial - RCNN [2014 Ross]:

A*. Rethinking about NMS

- **Brief history about NMS**
  - Traditional NMS [Here we use. Need to know how to code in C++/python]
  - Soft-NMS [2017, better to know]
  - IoU-Net [2018, Localization Confidence + PrROI Pooling, better to know]
  - Softer-NMS [2019, KL-Loss + Softer-NMS, better to know]

Important / Will cover / Remind

I. Two Stage Detection

A. First Trial - RCNN [2014 Ross]:

A*. Rethinking about NMS

- **Soft-NMS**

Input : $\mathcal{B} = \{b_1, .., b_N\}, \mathcal{S} = \{s_1, .., s_N\}, N_t$
$\mathcal{B}$ is the list of initial detection boxes
$\mathcal{S}$ contains corresponding detection scores
N_t is the NMS threshold

begin
  $\mathcal{D} \leftarrow \{\}$
  while $\mathcal{B} \neq$ *empty* **do**
    $m \leftarrow$ argmax $\mathcal{S}$
    $\mathcal{M} \leftarrow b_m$
    $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{M}; \mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
    for b_i *in* $\mathcal{B}$ **do**
      if $iou(\mathcal{M}, b_i) \geq N_t$ **then**
        $\mathcal{B} \leftarrow \mathcal{B} - b_i; \mathcal{S} \leftarrow \mathcal{S} - s_i$
      end NMS
      $s_i \leftarrow s_i f(iou(\mathcal{M}, b_i))$ Soft-NMS
    end
  end
  return $\mathcal{D}, \mathcal{S}$
end

I. Two Stage Detection

A. First Trial - RCNN [2014 Ross]:

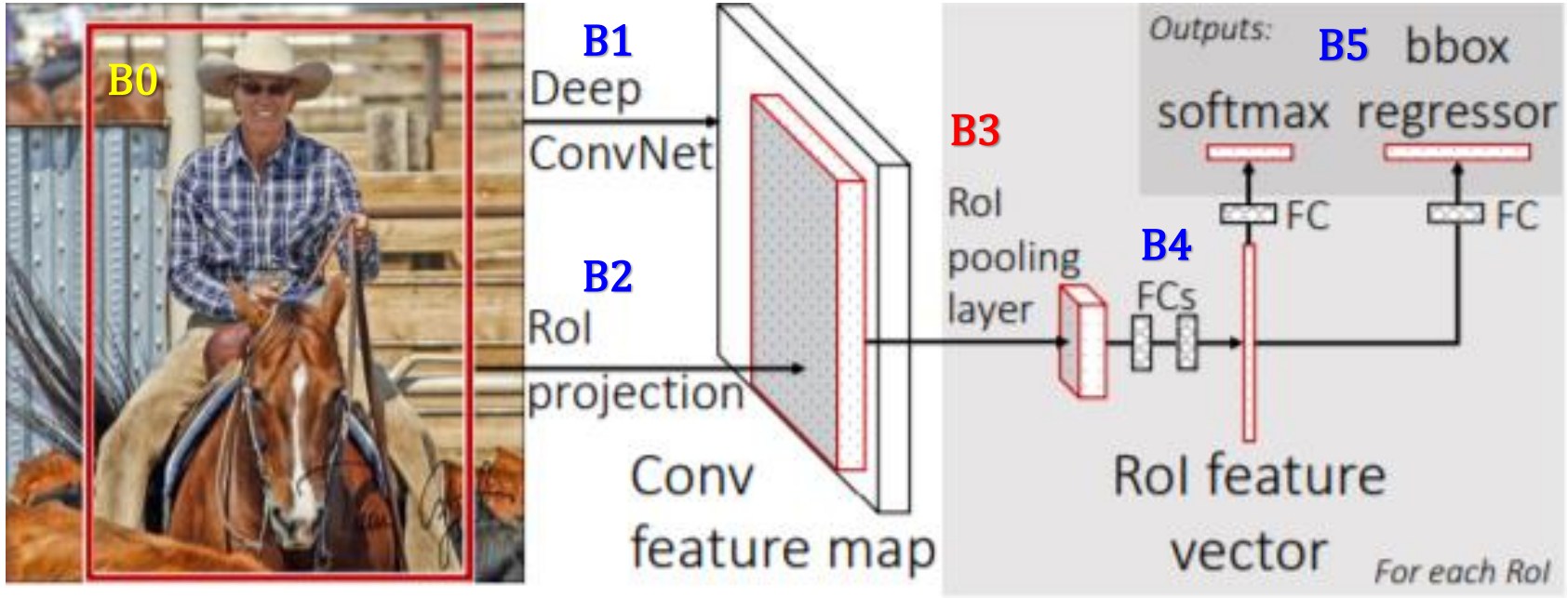
A4. RCNN Problems

- **Slow: need to run full forward pass of CNN for each region proposal**
- **SVMs and regressors are post-hoc: CNN features not updated in response to SVMs and regressors**
- **Complex multistage training pipeline**

Let's Fix It, Step by Step

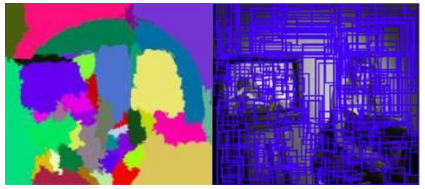
I. Two Stage Detection

B. Fast R-CNN [2015 Ross]:



I. Two Stage Detection

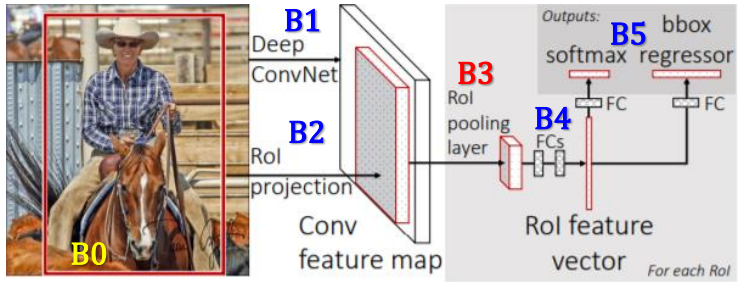
B. Fast R-CNN [2015 Ross]:



B0. Region Proposal: Same as RCNN

- 2k per image. Record location for each ROI

B1 & 2. Convolution & Projection



- Do conv for image. Project location for each ROI
- 3 basic structures provided, use Vgg16 as an E.g.
- 4 max pooling, /16

B3. ROI Pooling

- Grid each ROI in feature map to fixed size, and do max pooling within each grid
- So different size of feature maps can transfer into feature maps with same size.

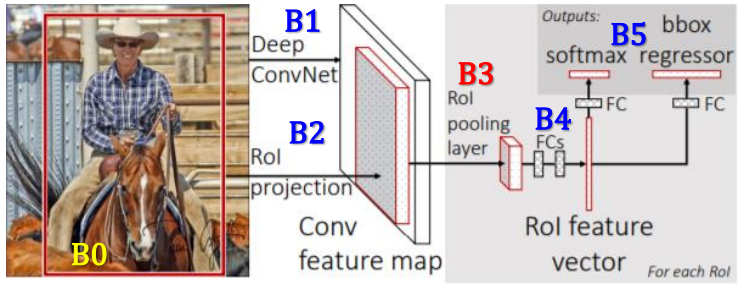
- bs = 2; 64 ROIs / image -> 128 proposals;
- ¼ pos & ¾ neg
- IoU > 0.5 pos, [0.1, 0.5) neg , [0. 0.1) hard neg
- Train on pos & neg. Test use hard neg. Retrain bad case

I. Two Stage Detection

B. Fast R-CNN [2015 Ross]:

B4. FC Layers

- FC layer cost a lot. Can use SVD to accelerate



B5. Multi-task Loss

- Classification: n + 1(background), softmax + cross entropy, L_{cls}
- Localization: Offset, SmoothL1, L_{loc}
- Total: $L(p, u, t^u, t^v) = L_{cls}(p, u) + \lambda[u \geq 1]\ L_{loc}(t^u, v)$

  p: predicted class score

  u: real class

  t^u: generated bbox [offset]

  t^v: real bbox

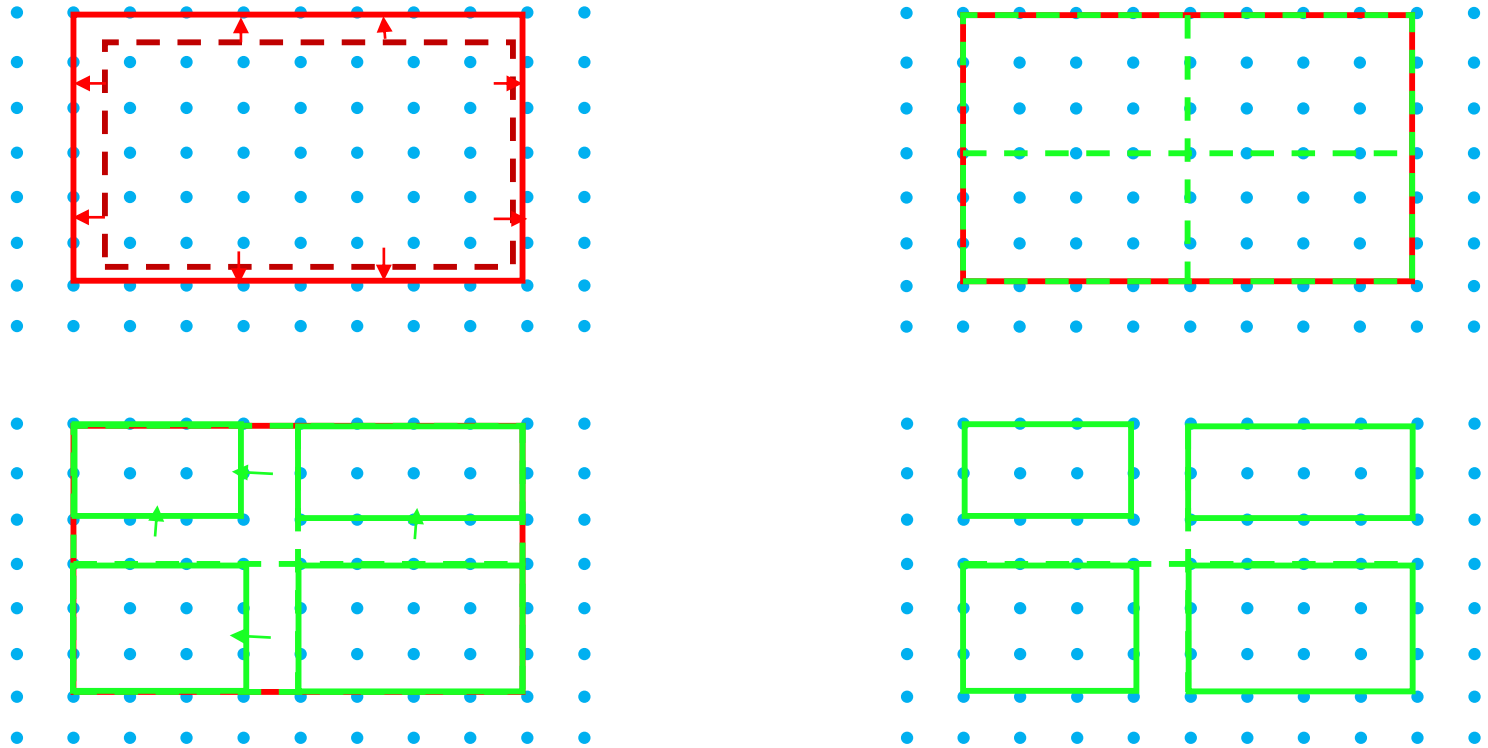
  $$\lambda = \begin{cases} 1, u \geq 1 \\ 0, u = 0 \end{cases}$$

SmoothL1 & Offset regression will be discussed in next topic

I. Two Stage Detection

B. Fast R-CNN [2015 Ross]:

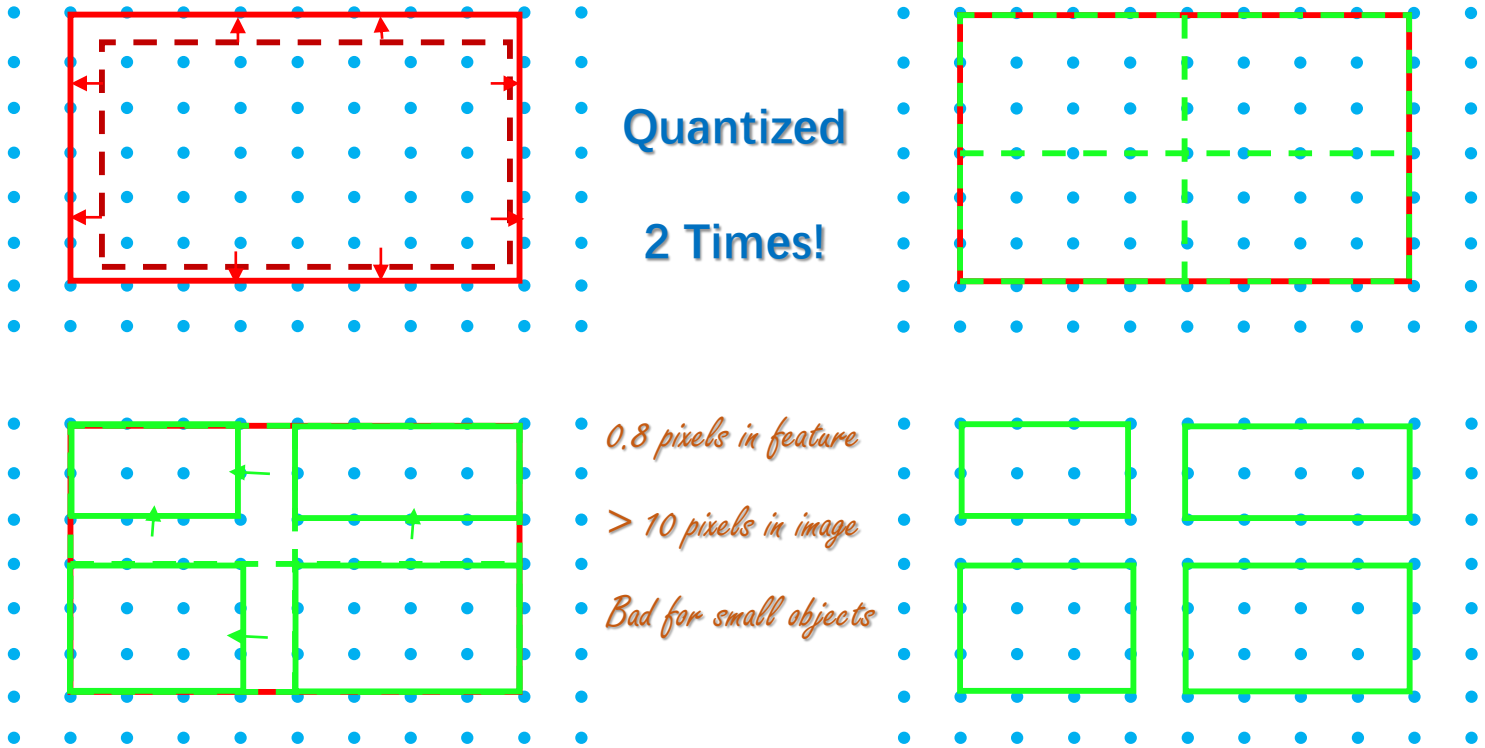
B*. Rethinking ROI Pooling



I. Two Stage Detection

B. Fast R-CNN [2015 Ross]:

B*. Rethinking ROI Pooling



I. Two Stage Detection

B. Fast R-CNN [2015 Ross]:

B*. Rethinking ROI Pooling

- **BP for ROI Pooling**

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r,j)] \frac{\partial L}{\partial y_{r,j}}$$

$y_{r,j}$: rth region, jth feature point

$i^*(r,j)$: the position where $y_{r,j}$ comes from

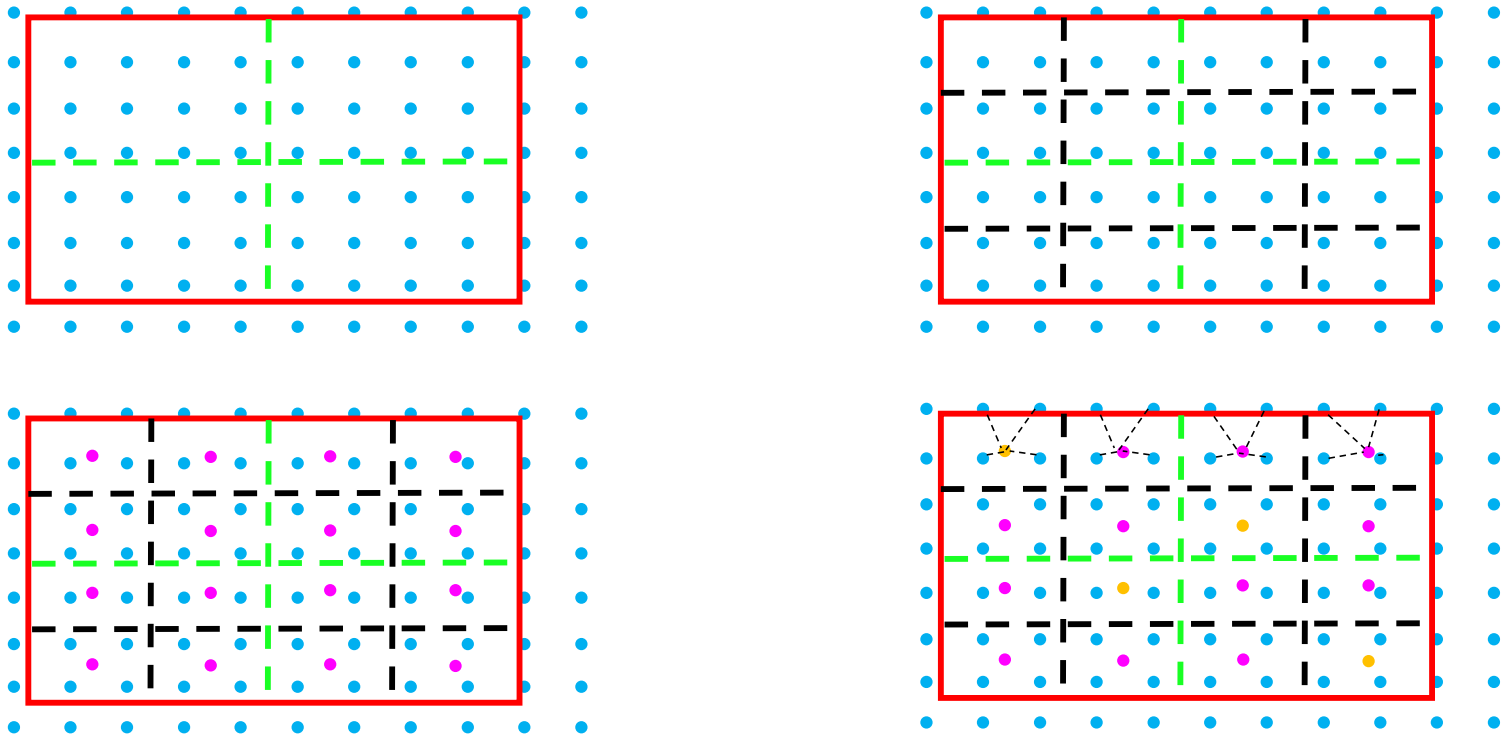
Just one point has loss passed by in each bin.

I. Two Stage Detection

B. Fast R-CNN [2015 Ross]:

B*. Rethinking ROI Pooling

- ROI Align [2017, Kaiming]

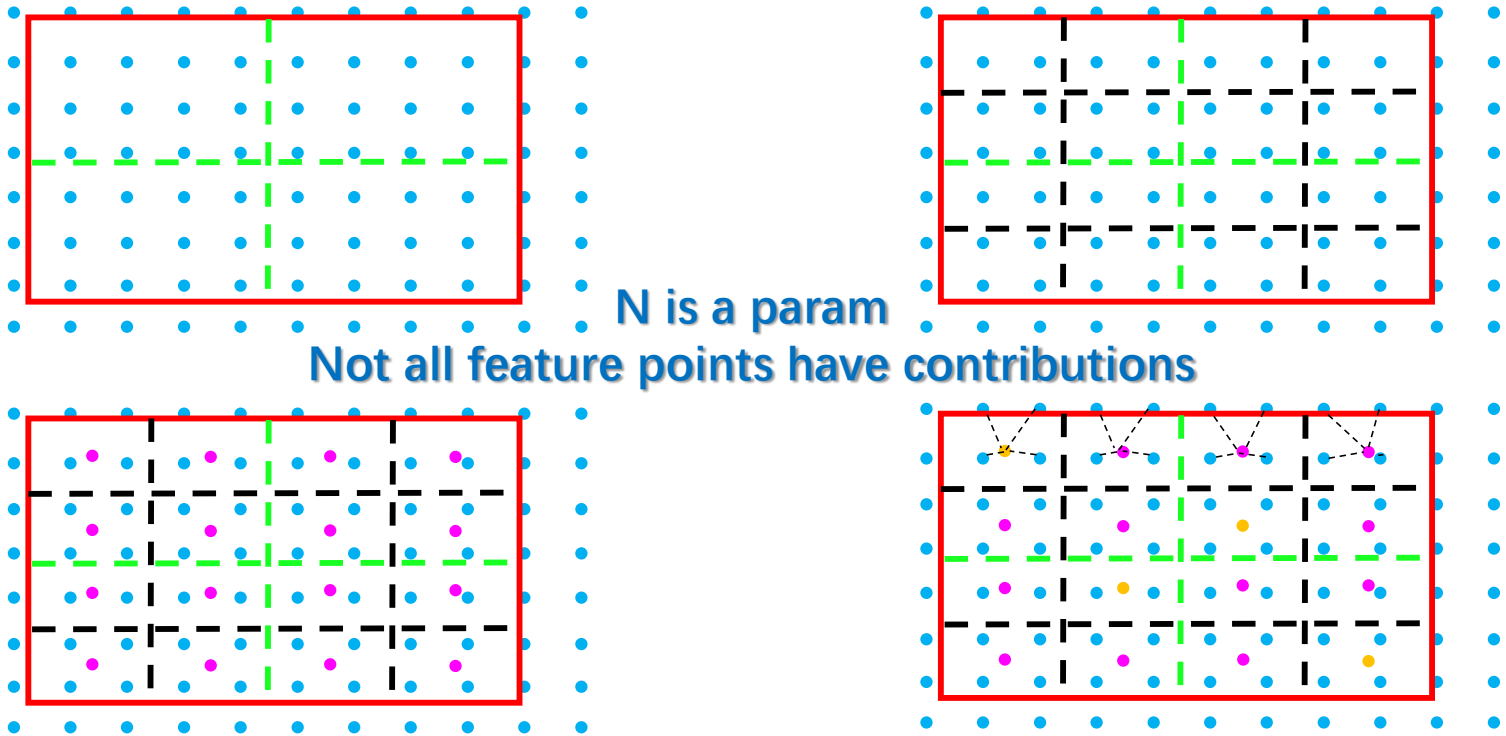


I. Two Stage Detection

B. Fast R-CNN [2015 Ross]:

B*. Rethinking ROI Pooling

- ROI Align [2017, Kaiming]



I. Two Stage Detection

B. Fast R-CNN [2015 Ross]:

B*. Rethinking ROI Pooling

- **ROI Align [2017, Kaiming] – BP**

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [d(i, i^*(r,j))](1-\Delta h)(1-\Delta w)\frac{\partial L}{\partial y_{r,j}}$$

$d(x, y)$: distance between x & y

$\Delta w, \Delta h$: width & height difference of x_i & $x_{i^*(r,j)}$

$x_{i^*(r,j)}$: a float value interpolated by the forward pass

x_i: feature point before pooling

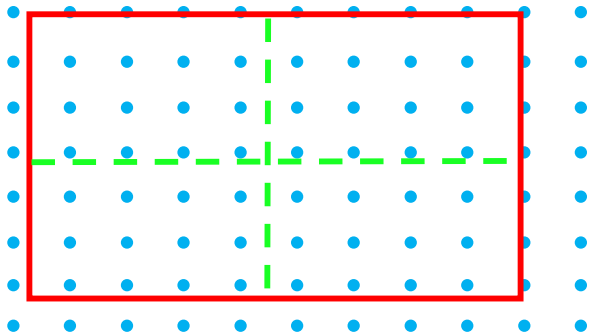
Only several points have losses passed by in each bin.

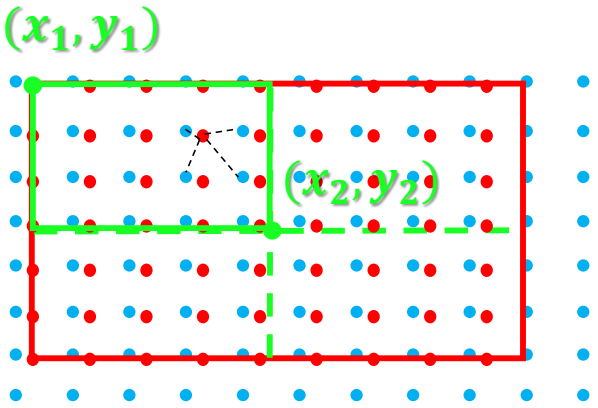
I. Two Stage Detection

B. Fast R-CNN [2015 Ross]:

B*. Rethinking ROI Pooling

- Precise ROI Pooling [2018, IoU-Net]





No Quantization

No Parameter

Pass losses for all

$$f(x,y) = \sum_{i,j} IC(x,y,i,j) * w_{i,j}$$

$$IC(x,y,i,j) = \max(0,1-|x-i|) + \max(0,1-|y-j|)$$

Do Pr ROI Pool for ▭:

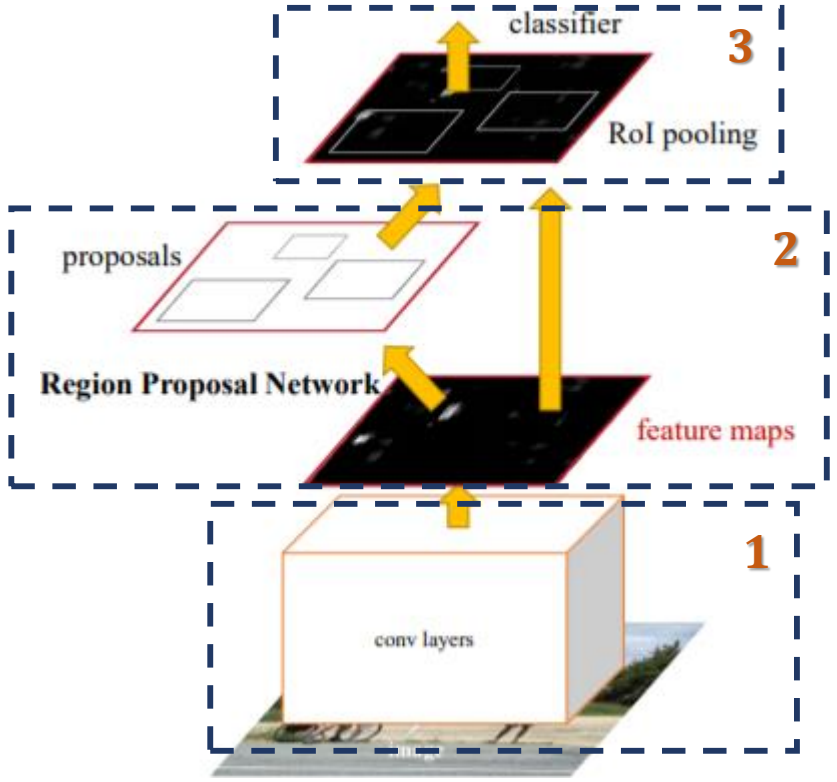
$$PrPool(bin,F) = \frac{\int_{y_1}^{y_2}\int_{x_1}^{x_2} f(x,y)dxdy}{(x_2-x_1)*(y_2-y_1)}$$

BP:

$$\frac{\partial PrPool(bin,F)}{\partial x_1} = \frac{PrPool(bin,F)}{x_2-x_1} - \frac{\int_{y_1}^{y_2} f(x,y)dy}{(x_2-x_1)*(y_2-y_1)}$$

I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:



C0. Structure

3. Fast RCNN: ROI + { Classification, Regression }

2. RPN

1. Backbone

Input

I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C1. Backbone

- **Aim:** Extract feature integratedly
  - No need to extract feature per ROI
  - Use generated feature to generate proposals

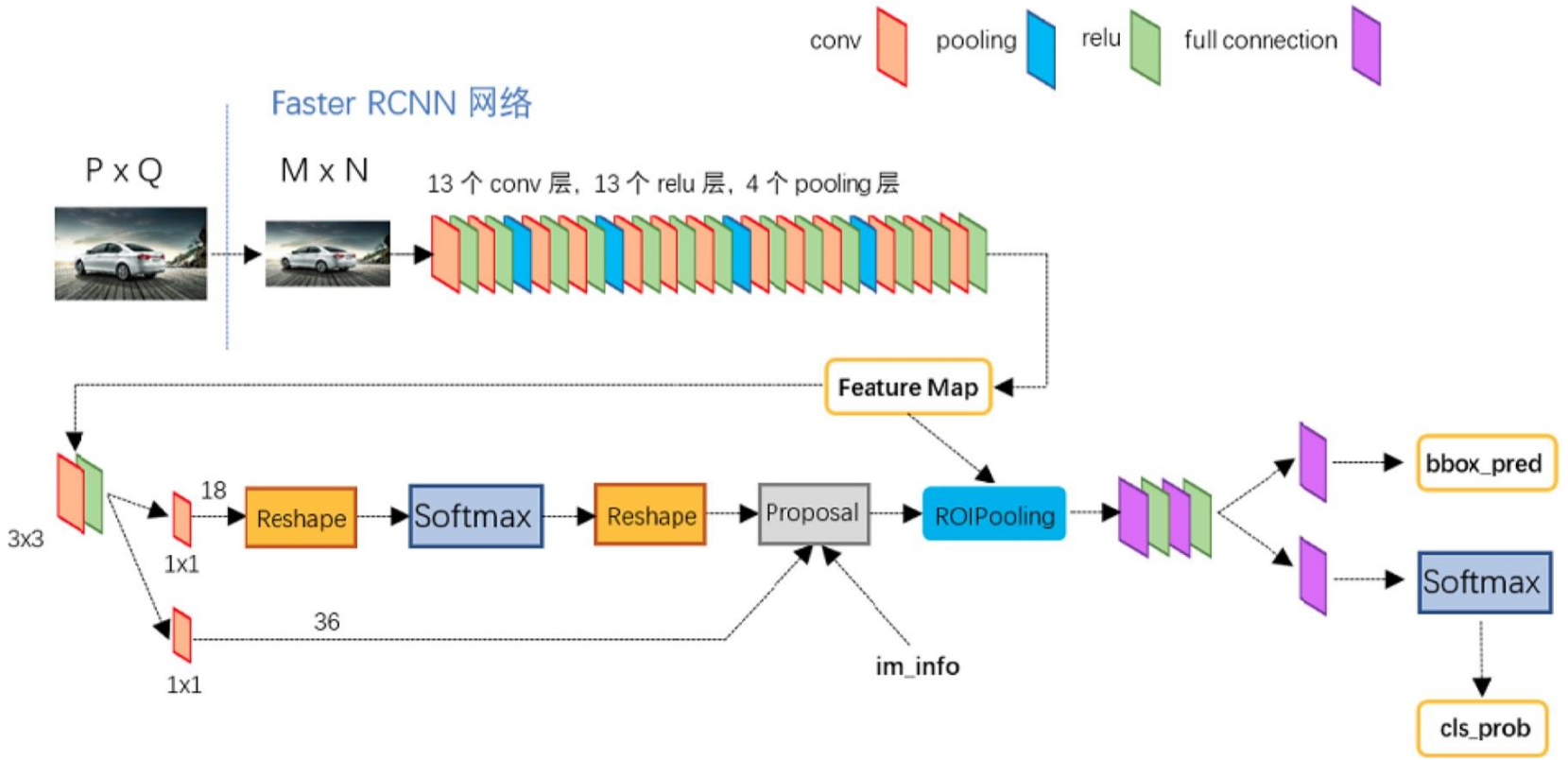
- **Structure:** ZF / Resnet / VGG16 (13 conv + 13 ReLU + 4 Pooling)
  [Do have the sense of splicing network from now on]

- **Output:** B x C x H/16 x W/16
  1 x 256 x 38 x 50
  [Feature Map]

I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

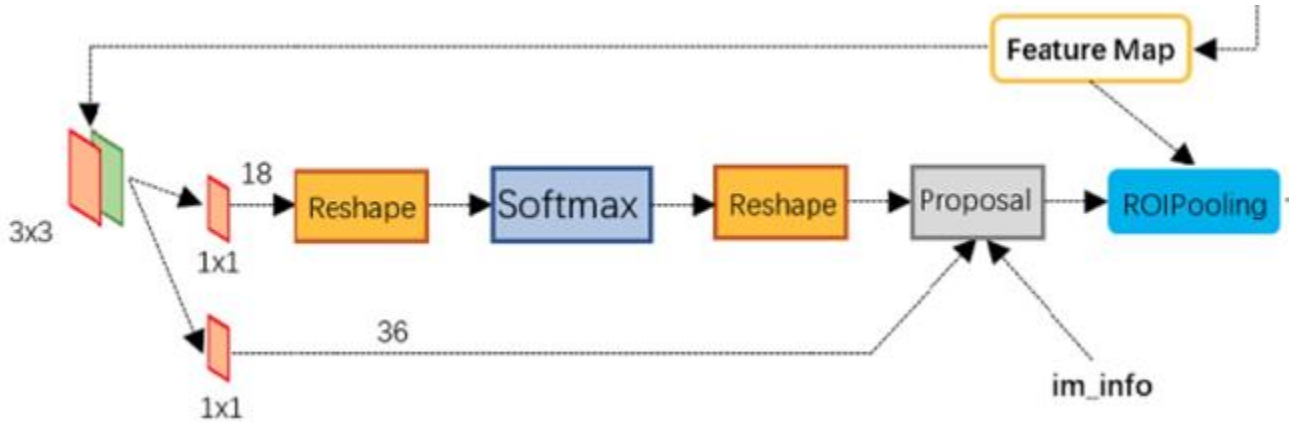
C1. Backbone



I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN



I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Aim: Generate region proposal**
  [Real detected objects are coming from those RPs]

a. **It's the reason where the name "Two-Stage" coming from:**
   RPN + Bbox Regression

b. **It's the reason why some argue two-stage detection has better results then one-stage methods.**

I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Output of RPN:**
  a. **ROIs: 128 x 5**
     [0, x1, y1, x2, y2]->physical region proposal

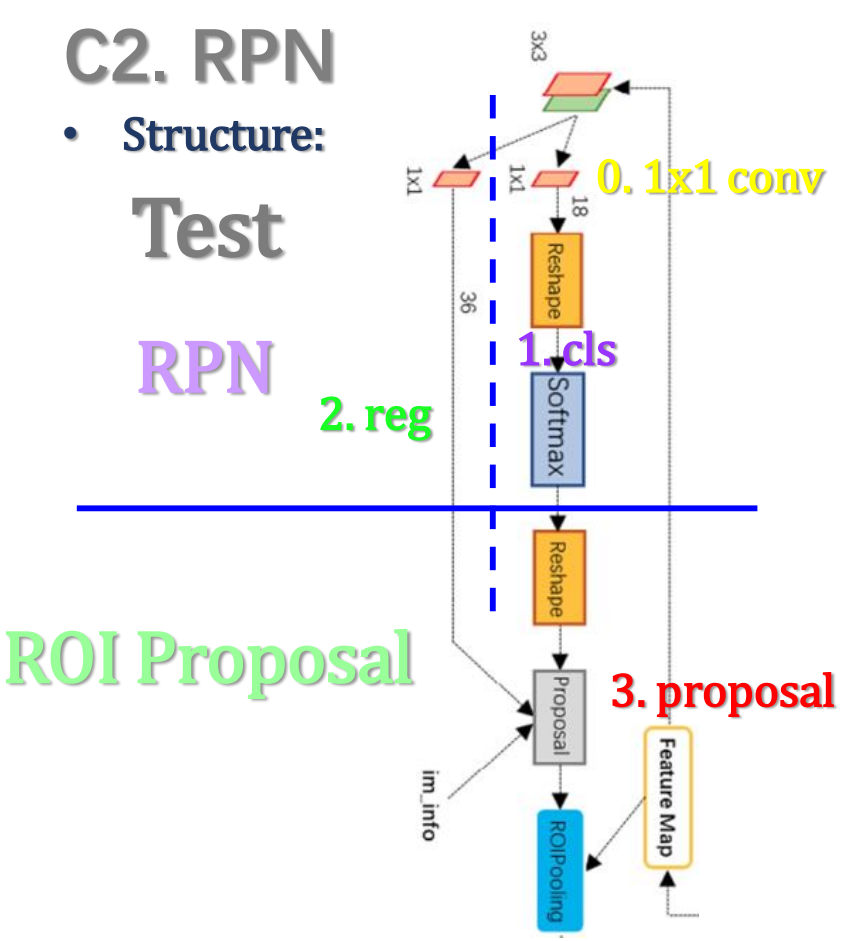
  b. **Label: 128**
     [0 ~ 20]->ROIs' classifications

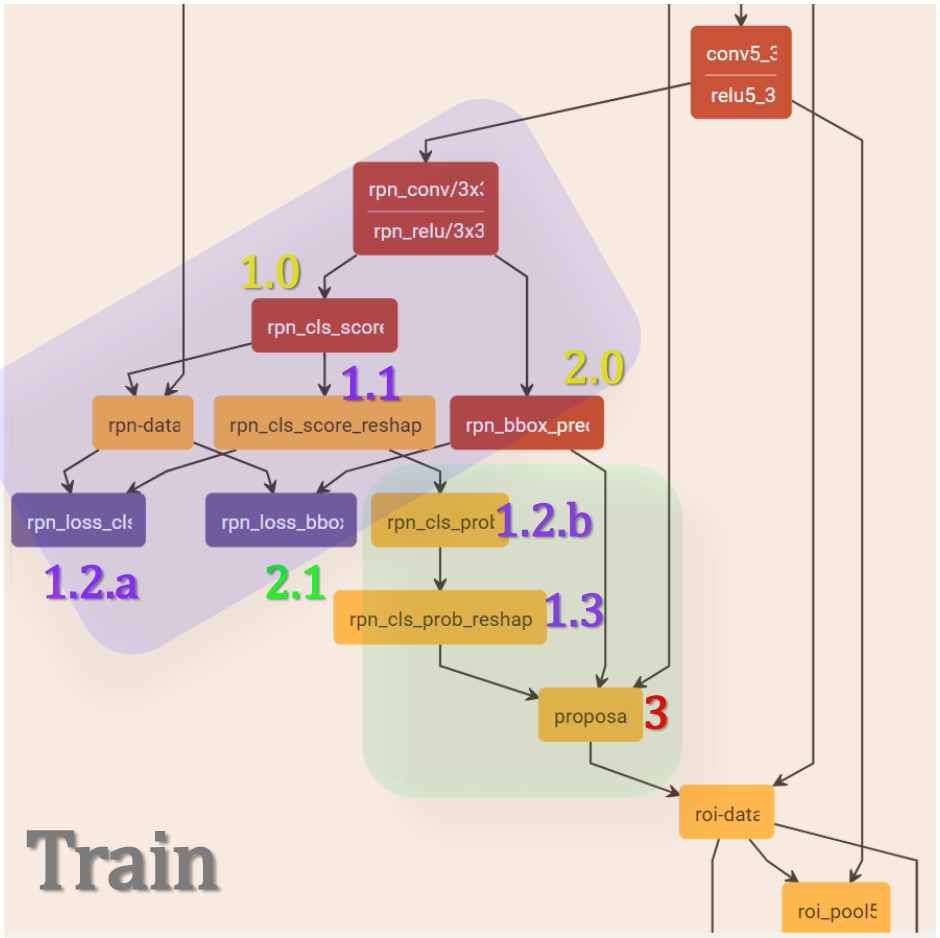
  c. **bbx_target: 128 x 84**
     [(20 + 1) x 4]->targets for bounding box regression

  d. **bbx_weight: 128 x 84**
     [0 0r 1]->weights for bbx_target when box regressing

I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:





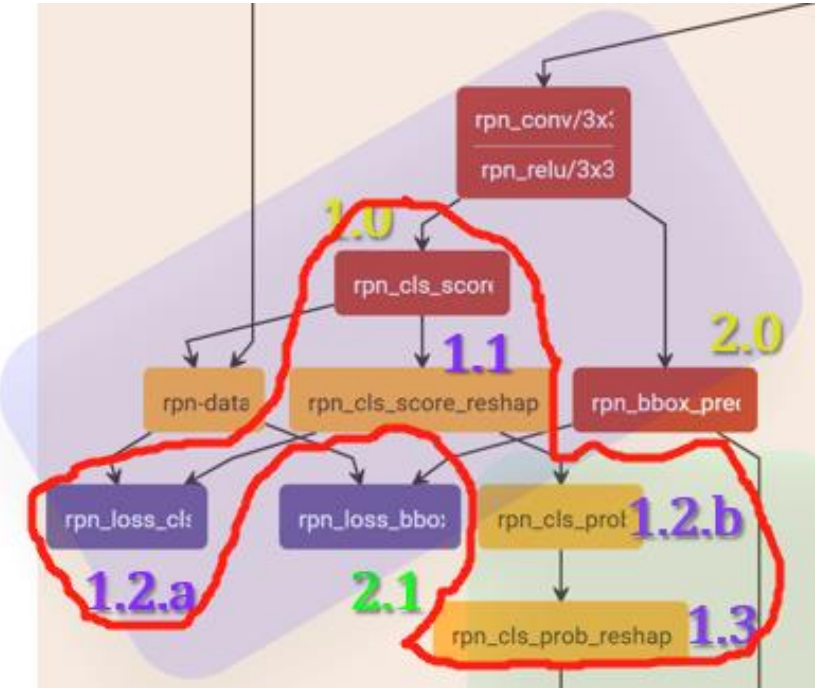
I. Two Stage Detection

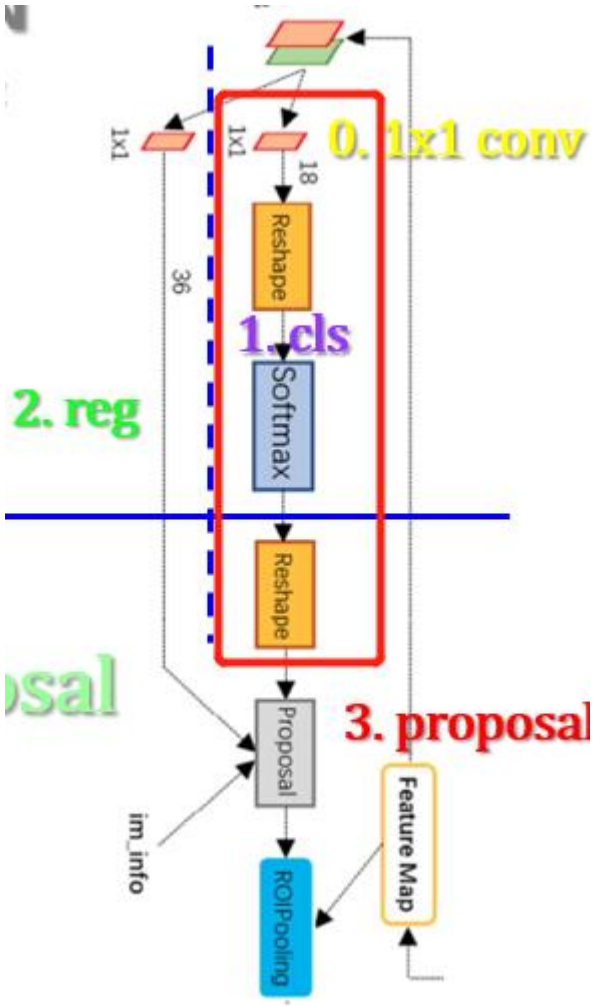
C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Structure:**

C2.1: classification branch





I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Structure:**

C2.1.0: 1x1 conv

1 x 18 x 38 x 50

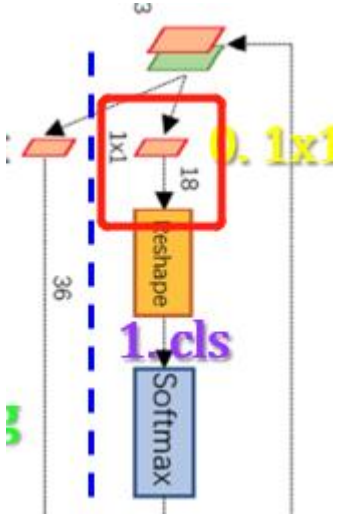
9 x 2

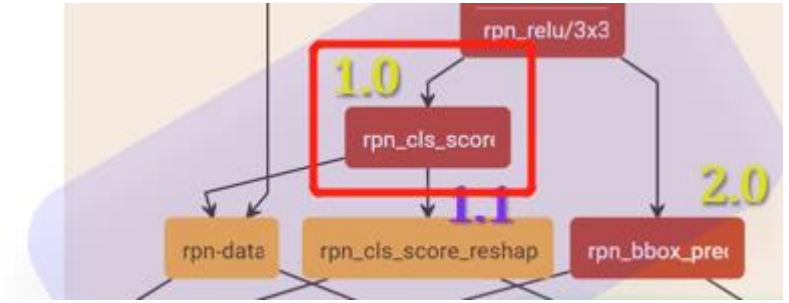
anchors 0|1

50

38

For each feature point,
We generate 9 anchors





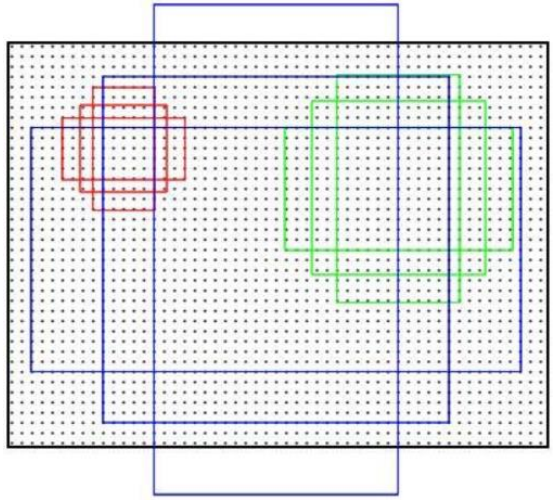
I. Two Stage Detection

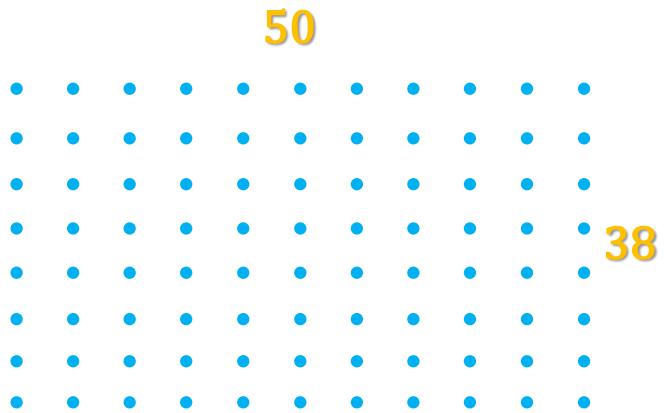
C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Structure - Anchor:**

- ❑ **Represent an area in original image (a.k.a. where objects are)**
- ❑ **Have different scale & ratio to cover all kinds of objects**
- ❑ **9 in total: 3 scales x 3 ratios**
- ❑ **1 x 9 x 38 x 50 = 17100 anchors**
- ❑ **Coords of anchors are of original imgs**





I. Two Stage Detection

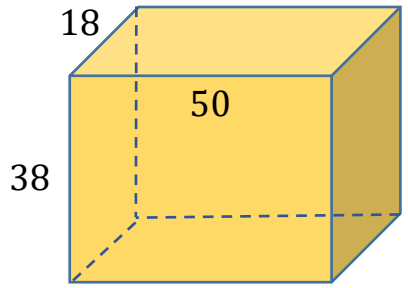
C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Structure:**

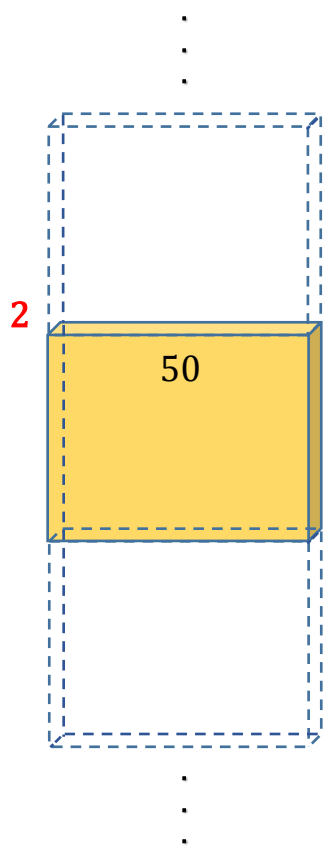
C2.1.1: cls reshape

Input:

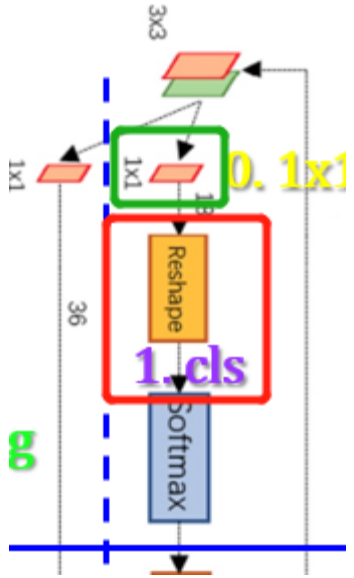


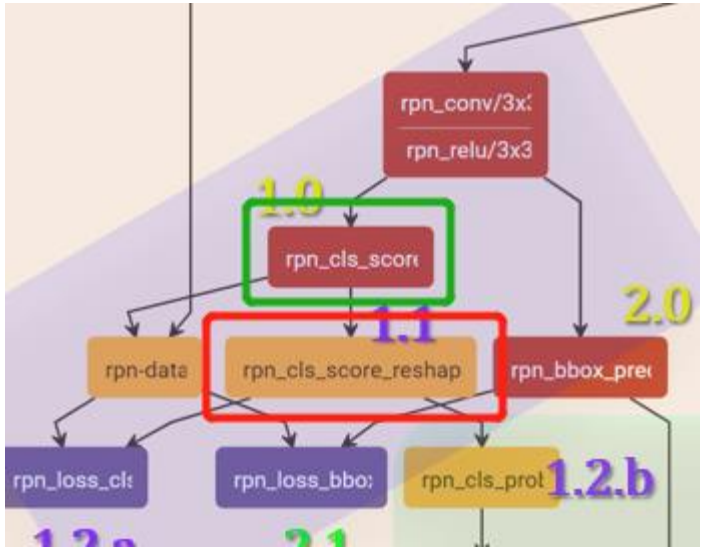
—>reshape

Output:



This is for doing **Softmax** to get **fore/background**





I. Two Stage Detection

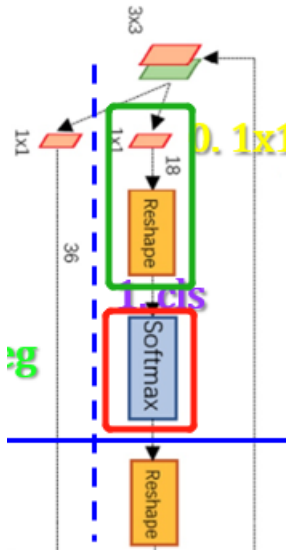
C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Structure:**

C2.1.2: Softmax

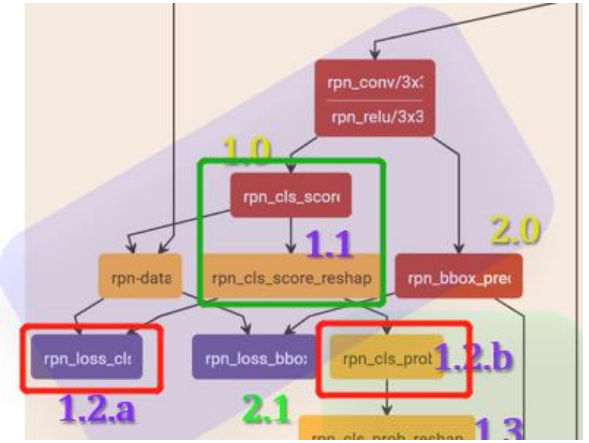
This is for doing **Softmax** to get **fore/background**



2 branches: a. bp loss & b. no loss / get score

- ❑ **a. 1 foreground & 0 background.**
  Get loss with anchor and pass back.
  Use IoU

- ❑ **b. Just do classification and get the score,**
  Use the score to select proposal



I. Two Stage Detection

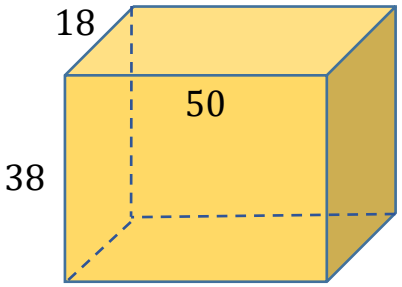
C. Faster R-CNN [2015 Ren]:

C2. RPN

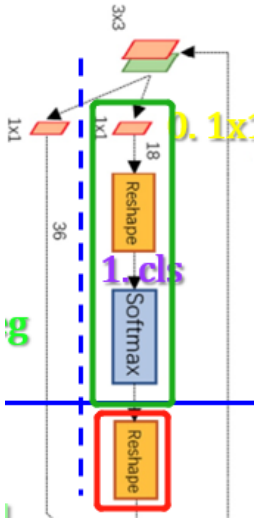
- **Structure:**

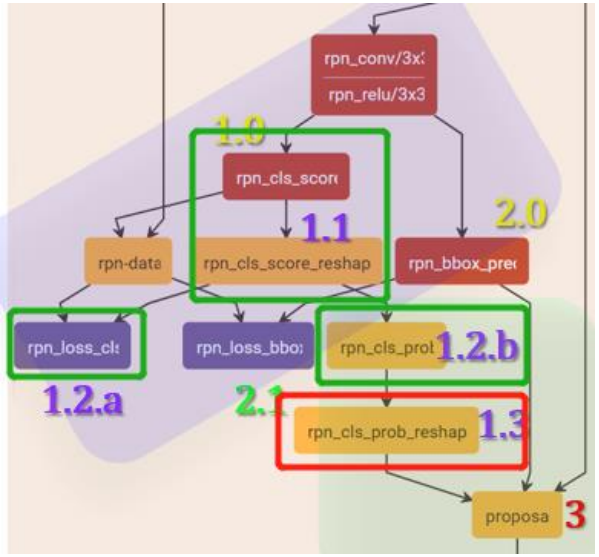
C2.1.3: cls reshape 2

Back to original shape



Each voxel represents the possibility of being a foreground or a background of an anchor





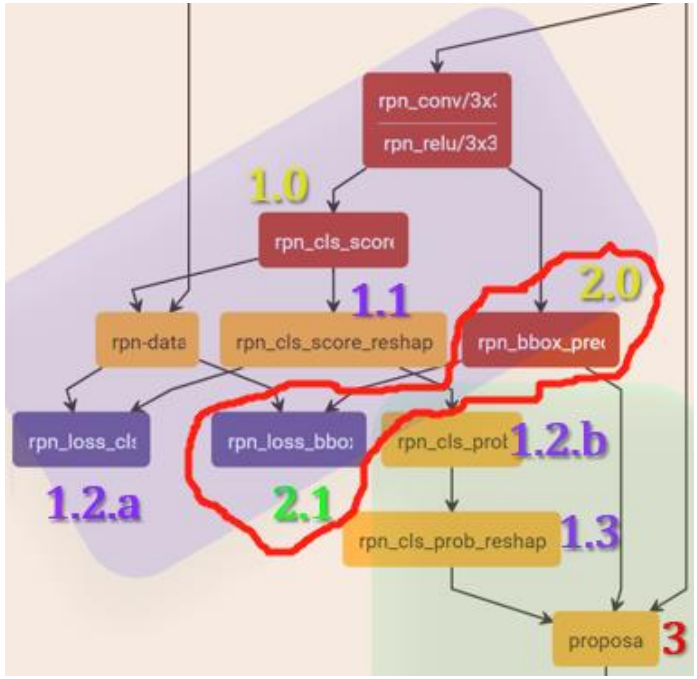
I. Two Stage Detection

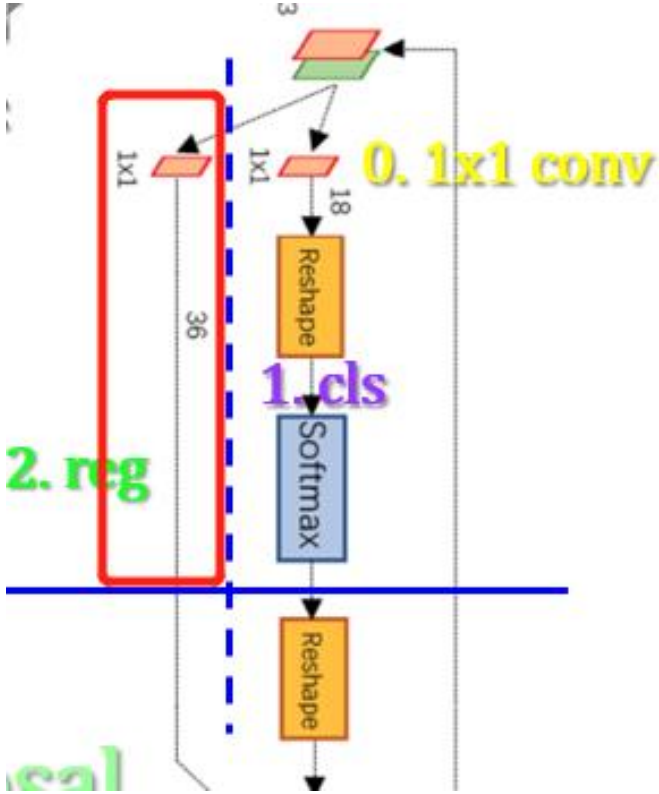
C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Structure:**

C2.2: regression





I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

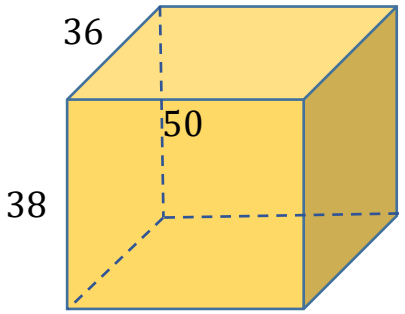
- **Structure:**

C2.2.0: 1x1 conv

1 x 36 x 38 x 50

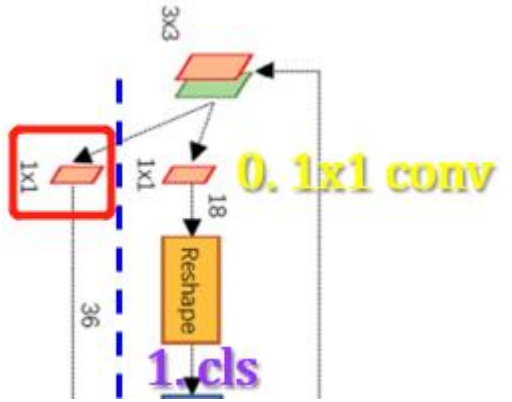
9 x 4

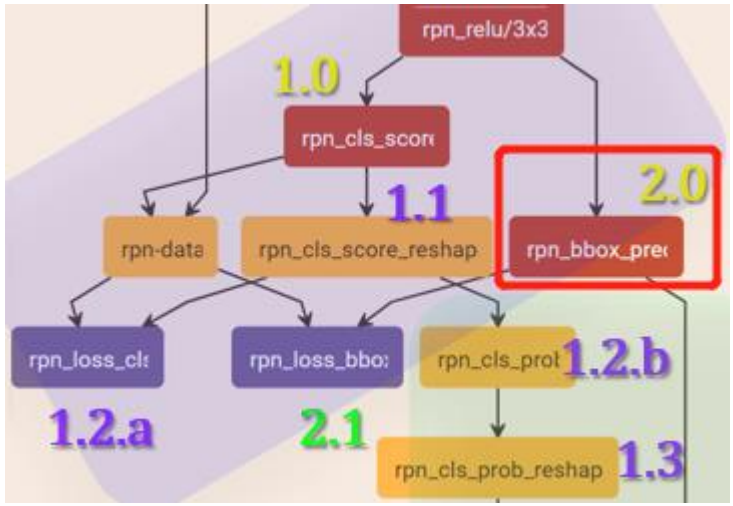
anchors 4 targets



Regress anchor to gt_bbox

Targets: proposals





I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Structure:**

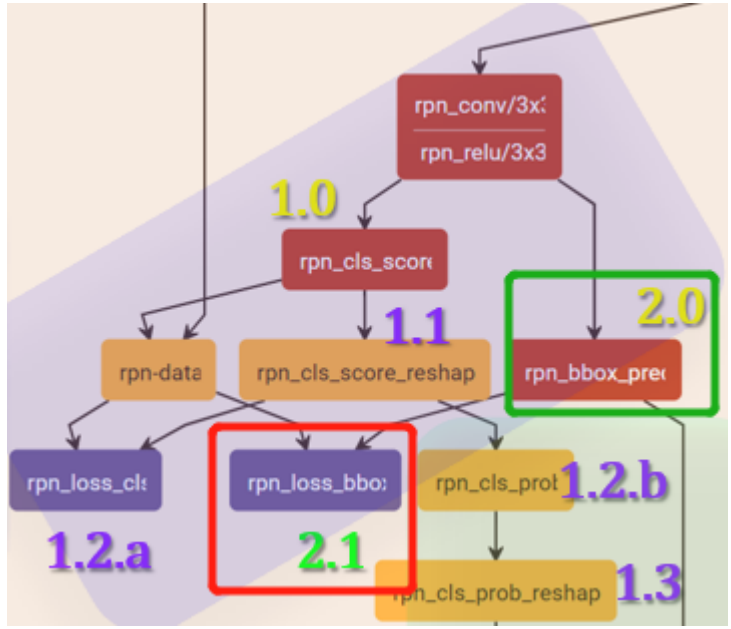
C2.2.1: Smooth L1 Loss

- **Need to multiply the mask**

  - anchor = 1, mask = 1
  - anchor = 0, -1, mask = 0

- **Smooth L1 Loss**

$$f(x) = \begin{cases} \frac{(\sigma x)^2}{2}, & if\ x < 1/\sigma^2 \\ |x| - \frac{0.5}{\sigma^2}, & otherwise \end{cases}$$



I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

- Structure:

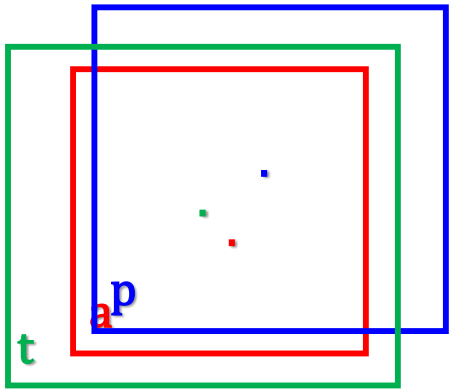
C2.2.1: Smooth L1 Loss

➢ Regress (center) offset, NOT COORDINATES

predicated bbox: x, y, w, h

anchor bbox: x_a, y_a, w_a, h_a

true bbox: x^*, y^*, w^*, h^*



We hope a -> t,
but actually a generates p.

So as long as p->t,
We get a good result.

So as long as the offset of
p - a -> t – a
We get a good result

So here we hope $t_\cdot \rightarrow t_\cdot^*$

$$f(x) = \begin{cases} t_x = \frac{x - x_a}{w_a}, t_y = \frac{y - y_a}{h_a} \\ t_w = log\left(\frac{w}{w_a}\right), t_h = log\left(\frac{h}{h_a}\right) \\ t_x^* = \frac{x^* - x_a}{w_a}, t_y^* = \frac{y^* - y_a}{h_a} \\ t_w^* = log\left(\frac{w^*}{w_a}\right), t_h = log\left(\frac{h^*}{h_a}\right) \end{cases}$$

I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

- Structure:

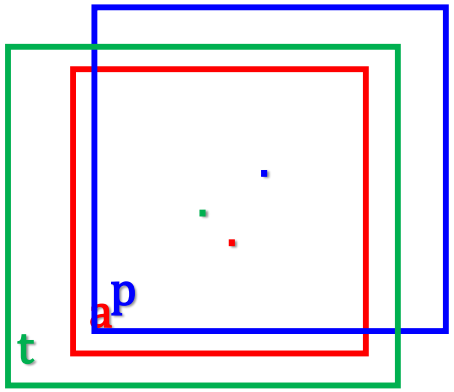
C2.2.1: Smooth L1 Loss

➢ Regress (center) offset, NOT COORDINATES

predicated bbox: x, y, w, h

anchor bbox: x_a, y_a, w_a, h_a

true bbox: x^*, y^*, w^*, h^*



So how to transfer from one box to another?

$$f(x) = \begin{cases} t_x = \frac{x - x_a}{w_a}, t_y = \frac{y - y_a}{h_a} \\ t_w = log\left(\frac{w}{w_a}\right), t_h = log\left(\frac{h}{h_a}\right) \\ t_x^* = \frac{x^* - x_a}{w_a}, t_y^* = \frac{y^* - y_a}{h_a} \\ t_w^* = log\left(\frac{w^*}{w_a}\right), t_h = log\left(\frac{h^*}{h_a}\right) \end{cases}$$

I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

- Structure:

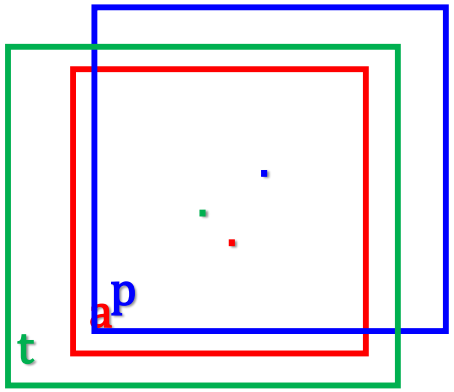
C2.2.1: Smooth L1 Loss

➢ Regress (center) offset, NOT COORDINATES

predicated bbox: x, y, w, h

anchor bbox: x_a, y_a, w_a, h_a

true bbox: x^*, y^*, w^*, h^*



So how to transfer from one box to another?

Shift + Scale

$$f(x) = \begin{cases} t_x = \frac{x - x_a}{w_a}, t_y = \frac{y - y_a}{h_a} \\ t_w = log\left(\frac{w}{w_a}\right), t_h = log\left(\frac{h}{h_a}\right) \\ t_x^* = \frac{x^* - x_a}{w_a}, t_y^* = \frac{y^* - y_a}{h_a} \\ t_w^* = log\left(\frac{w^*}{w_a}\right), t_h = log\left(\frac{h^*}{h_a}\right) \end{cases}$$

I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Structure:**

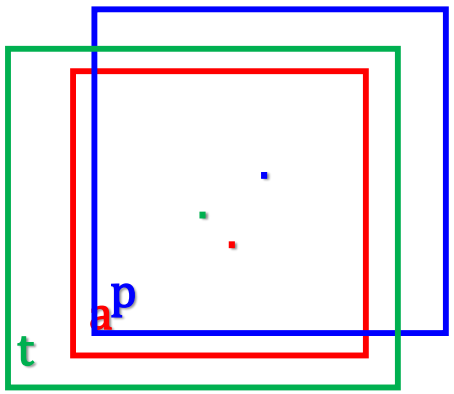
C2.2.1: Smooth L1 Loss

➢ **Regress (center) offset, NOT COORDINATES**

predicated bbox: x, y, w, h

anchor bbox: x_a, y_a, w_a, h_a

true bbox: x^*, y^*, w^*, h^*



$$x = x_a + \Delta x \quad \text{Shift}$$
$$y = y_a + \Delta y$$
$$w = w_a \cdot \Delta w \quad \text{Scale}$$
$$h = h_a \cdot \Delta h$$

$$f(x) = \begin{cases} t_x = \frac{x - x_a}{w_a}, t_y = \frac{y - y_a}{h_a} \\ t_w = log\left(\frac{w}{w_a}\right), t_h = log\left(\frac{h}{h_a}\right) \\ t_x^* = \frac{x^* - x_a}{w_a}, t_y^* = \frac{y^* - y_a}{h_a} \\ t_w^* = log\left(\frac{w^*}{w_a}\right), t_h = log\left(\frac{h^*}{h_a}\right) \end{cases}$$

I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Structure:**

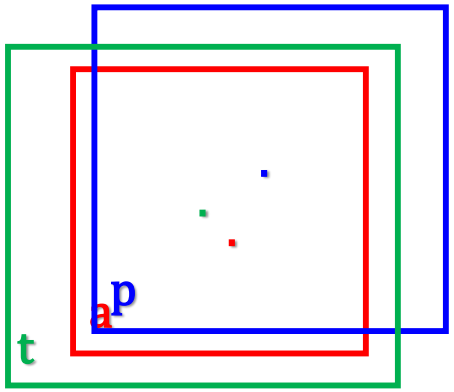
C2.2.1: Smooth L1 Loss

➢ **Regress (center) offset, NOT COORDINATES**

predicated bbox: x, y, w, h

anchor bbox: x_a, y_a, w_a, h_a

true bbox: x^*, y^*, w^*, h^*



$$x = x_a + \Delta x$$
$$y = y_a + \Delta y$$
$$w = w_a \cdot \Delta w$$
$$h = h_a \cdot \Delta h$$

Shift

Scale

$$\Delta x = w_a \cdot t_x$$
$$\Delta y = h_a \cdot t_y$$
$$\Delta w = w_a \cdot e^{t_w}$$
$$\Delta h = h_a \cdot e^{t_h}$$

$$f(x) = \begin{cases} t_x = \frac{x - x_a}{w_a}, t_y = \frac{y - y_a}{h_a} \\ t_w = log\left(\frac{w}{w_a}\right), t_h = log\left(\frac{h}{h_a}\right) \\ t_x^* = \frac{x^* - x_a}{w_a}, t_y^* = \frac{y^* - y_a}{h_a} \\ t_w^* = log\left(\frac{w^*}{w_a}\right), t_h = log\left(\frac{h^*}{h_a}\right) \end{cases}$$

I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

- Structure:

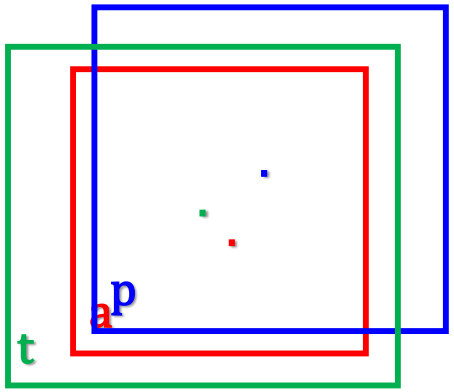
C2.2.1: Smooth L1 Loss

➢ Regress (center) offset, NOT COORDINATES

predicated bbox: x, y, w, h

anchor bbox: x_a, y_a, w_a, h_a

true bbox: x^*, y^*, w^*, h^*



Q: Why use exp

A: tx & ty is used for scaling
we need to ensure scale > 0

Q: Why use log

A: 1. reversing procedure of exp
2. Suppress greater bbox

$$f(x) = \begin{cases} t_x = \frac{x - x_a}{w_a}, t_y = \frac{y - y_a}{h_a} \\ t_w = log\left(\frac{w}{w_a}\right), t_h = log\left(\frac{h}{h_a}\right) \\ t_x^* = \frac{x^* - x_a}{w_a}, t_y^* = \frac{y^* - y_a}{h_a} \\ t_w^* = log\left(\frac{w^*}{w_a}\right), t_h = log\left(\frac{h^*}{h_a}\right) \end{cases}$$

I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

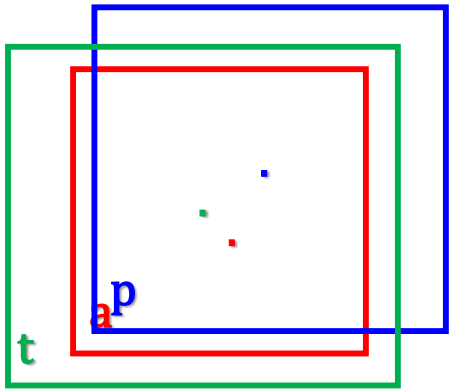
- Structure:

C2.2.1: Smooth L1 Loss

➢ Regress (center) offset, NOT COORDINATES

Q: Why x/y uses linear transferring
but w/h uses non-linear transferring?
Is that reasonable?

A: When bbox1->bbox2, then we can.



predicated bbox: x, y, w, h

anchor bbox: x_a, y_a, w_a, h_a

true bbox: x^*, y^*, w^*, h^*

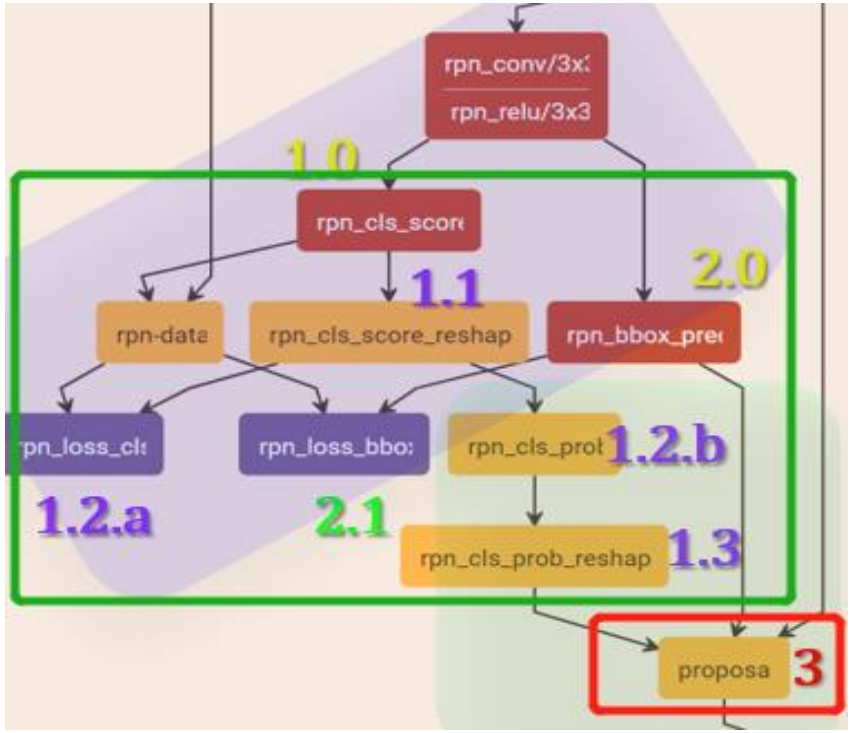
$$f(x) = \begin{cases} t_x = \frac{x - x_a}{w_a}, t_y = \frac{y - y_a}{h_a} \\ t_w = log\left(\frac{w}{w_a}\right), t_h = log\left(\frac{h}{h_a}\right) \\ t_x^* = \frac{x^* - x_a}{w_a}, t_y^* = \frac{y^* - y_a}{h_a} \\ t_w^* = log\left(\frac{w^*}{w_a}\right), t_h = log\left(\frac{h^*}{h_a}\right) \end{cases}$$

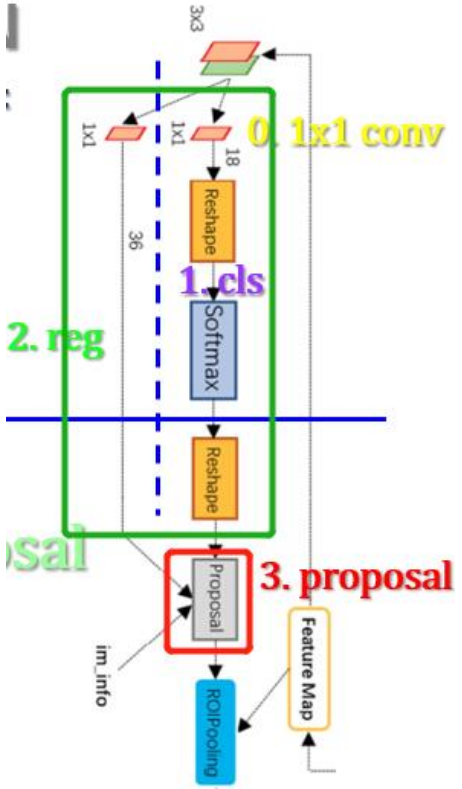
I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Structure:** C2.3: Proposal: Get proposal





I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C2. RPN

- **Structure:**

C2.3: Proposal: Get proposal

- Regard anchor as FG when IoU > 0.7
- Regard anchor as BG when IoU < 0.3
- Regardless other anchors
- Then just keep 128: 0.25 fg + 0.75 bg anchors

I. Two Stage Detection

C. Faster R-CNN [2015 Ren]:

C3. After RPN

- **Same as Fast RCNN**

C4. How to train

- **Classic: alternatively 4 steps**
  - Use ImageNet finetune our RPN
  - Use trained proposal from step1 to train an Fast RCNN
  - Use detected results to initialize RPN training where we freeze the backbone layers but to train pure RPN-related layers
  - Finetune Fast RCNN-related layers only.
- **Other method:**
  - We can also train Faster RCNN as a whole in just one step

II. One Stage Detection

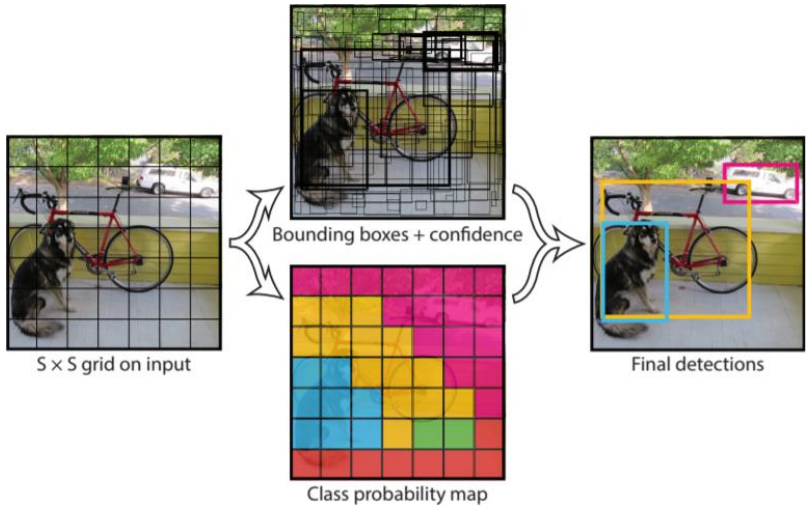
II. One Stage Detection

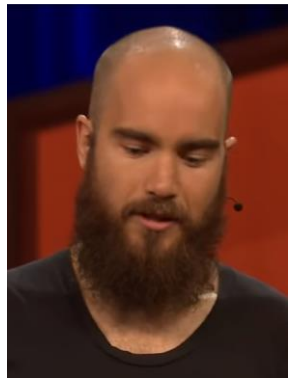
Why we have to train proposal first?

II. One Stage Detection

D. Yolo V1 [2015, Joseph]

D0. You Only Look Once!





- Really fast (18 faster rcnn [ZF] vs 45 yolo)
- Becoming prevalent (62.1 mAP vs 63.4)

II. One Stage Detection

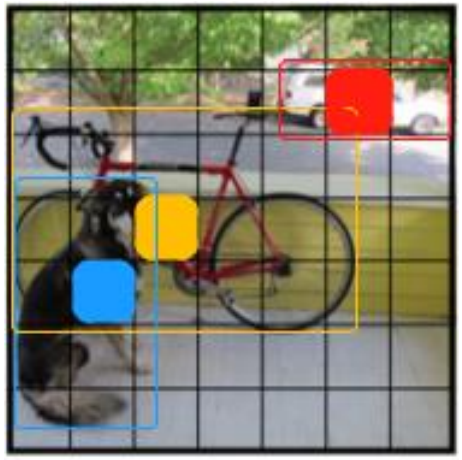
D. Yolo V1 [2015, Joseph]

D1. Procedure

- Grid an image into S x S cells [448 x 448 -> 7 x 7]

One cell will be responsible for predicting an object as long as an object's center locating in that cell.

S × S grid on input



S × S grid on input

II. One Stage Detection

D. Yolo V1 [2015, Joseph]

D1. Procedure

- Each cell predicts B bounding box with a confidence

Bounding Box: x, y, w, h (center)

Confidence: $P_r(object) \cdot IoU_{pred}^{truth}$

Final output tensor: S x S x (5 * B + C)

[7 x 7 x (5 * 2 + 20)] -> v1

II. One Stage Detection

D. Yolo V1 [2015, Joseph]

D2. Loss Function

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \quad (1)$$

2.1

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \quad (2)$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \quad (3)$$

2.2

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \quad (4)$$

2.3

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \quad (5)$$

$i: 0 \sim (S^2 - 1)$ [iterate each grid cell (0~48)]

$j: 0 \sim (B - 1)$ [iterate each bbox (0~2)]

$\mathbb{1}_{ij}^{\text{obj}}$ & $\mathbb{1}_{ij}^{\text{noobj}}$:

0	0	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

1	1	1	1	1	1	1
1	1	1	1	1	0	1
1	1	1	1	1	1	1
1	1	0	1	1	1	1
1	0	1	1	1	1	0
1	1	1	1	1	1	1
1	1	1	1	1	1	1

For $\mathbb{1}_{ij}^{\text{obj}}$, we have B predictions in each cell, only the one with largest IoU shall be labeled as 1.

II. One Stage Detection

D. Yolo V1 [2015, Joseph]

D2. Loss Function

D2.1: Coordinate loss

$$2.1 \left\{ \begin{aligned} &\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \quad (1) \\ &+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \quad (2) \end{aligned} \right.$$

x, y: predicated bbox center,
w, h: predicated bbox width & height

$\hat{x}, \hat{y}$: labeled bbox center,
$\hat{w}, \hat{h}$: labeled bbox width & height

$\sqrt{w}, \sqrt{h}$: Suppress the effect for larger bbox

λ_{coord}: 5, because there's only 8 dimensions. Too less comparing to other losses
Weighted loss essentially.

II. One Stage Detection

D. Yolo V1 [2015, Joseph]

D2. Loss Function

D2.2: Confidence loss

$\widehat{C}_i$: confidence score [IoU] of predicated and labeled bbox

C_i : predicated confidence score [IoU] generated from networks

$$2.2 \left(\right. + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \quad (3)$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 \quad (4)$$

Note: $\widehat{C}_i$ in (4) is 0

λ_{noobj}=0.5, because there's so many non-object bboxes

Train: $confidence = P_r(object) \cdot IoU_{pred}^{truth}$

$\mathbb{1}_{ij}^{\text{obj}}$ $\widehat{C}_i$

Test: Individual box confidence prediction:

$$confidence = P_r(cls_i/obj) P_r(obj) \cdot IoU_{pred}^{truth} = P_r(cls_i) \cdot IoU_{pred}^{truth}$$

II. One Stage Detection

D. Yolo V1 [2015, Joseph]

D2. Loss Function

D2.3: Classification Loss

Each sell will only predict 1 object, which is decided by the bbox with largest IoU

* Don't forget to do NMS after generating bboxes

$$\text{2.3} \quad + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (5)$$

II. One Stage Detection

D. Yolo V1 [2015, Joseph]

D3. Pros & Cons

D2.3: Classification Loss

Pros:

- One stage, really fast

Cons:

- Bad for crowed objects [1 cell 1 obj]
- Bad for small objects
- Bad for objects with new width-height ratio
- No BN

II. One Stage Detection

E. Yolo V2 [2016, Joseph]

E1. Improvements comparing to V1

E1.1: Add BN

E1.2: High Resolution Classifier [Focusing on backbone]

a. Train on ImageNet (224 x 224) // Model trained on small images may not be good
b. Resize & Finetune on ImageNet (448 x448) // So we finetune the model on larger images
c. Finetune on dataset // To let the model be used to larger images
d. We get 13 x 13 feature maps finally

E1.3: We Use Anchors [We'll talk it later]

II. One Stage Detection

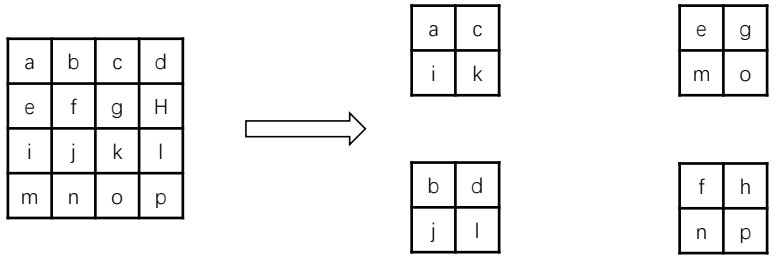
E. Yolo V2 [2016, Joseph]

E1. Improvements comparing to V1

E1.4: Fine-Grained Features

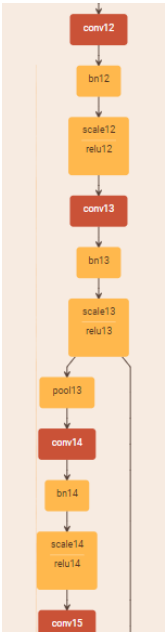
a. Lower features are concatenated directly to higher features

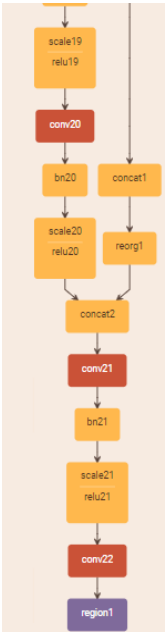
b. A new layer is added for that purpose: reorg



E1.5: Multi-Scale Training

- Remove FC layers: Can accept any size of inputs, enhance model robustness.
- Size across 320, 352, …, 608. Change per 10 epochs
  [border % 32 = 0, decided by down sampling]





II. One Stage Detection

E. Yolo V2 [2016, Joseph]

E2. Anchor in Yolo V2

E2.1: Anchor size and number

a. Faster RCNN: 9 by hands

b. Yolo: 5 by K-Means [dist: 1 – iou(bbox, cluster)]

E2.2: Anchors, Truth BBoxes & Predicted BBoxes

- Anchors: 0.57273, 0.677385, …, 9.77052, 9.16828 [10 numbers]

$$a_{w_0}, \quad a_{h_0} \quad , \ldots, \quad a_{w_4}, \quad a_{h_4}$$

$$\text{anchors}[0] = a_{w_i} = \frac{a_{w_{ori}}}{W} * 13$$ [not strict, just aiming to say how we get those numbers]

- Truth Bboxes:

II. One Stage Detection

E. Yolo V2 [2016, Joseph]

E2. Anchor in Yolo V2

E2.2: Anchors, Truth BBoxes & Predicted BBoxes

- **Anchors:** $\text{anchors}[0] = a_{w_i} = \frac{a_{w_{ori}}}{W} * 13$ [not strict, just aiming to say how we get those numbers]
- **Truth Bboxes:**

original bbox: $[x_o, y_o, w_o, h_o] \epsilon [0, W|H]$bbo

↓ **normalize in 0~1**

normalized original bbox: $[x_r, y_r, w_r, h_r] \epsilon [0,1]$

$[x_r, y_r, w_r, h_r] = [x_o/W, y_o/H, w_o/W, h_o/H]$

↓ **transfer to feature map size: 13 x 13**

box: $[x, y, w, h] \epsilon (0,13]$

$[x, y, w, h] = [x_r, y_r, w_r, h_r] * (13|13)$

↓ **save this for calculating**

↓ **transfer to 0~1 corresponding to each grid cell**

final box: $[x_f, y_f, w_f, h_f] \epsilon [0,1]$

↓ **save this for calculating**

↓ **transfer to 0~1 corresponding to each grid cell**

final box: $[x_f, y_f, w_f, h_f] \epsilon [0,1]$

$$\begin{cases} x_f = x - i \\ y_f = y - j \\ w_f = \log(w/anchors[0]) \\ h_f = \log(h/anchors[1]) \end{cases}$$

II. One Stage Detection

E. Yolo V2 [2016, Joseph]

E2. Anchor in Yolo V2

E2.2: Anchors, Truth BBoxes & Predicted BBoxes

- Output of Yolo V2: features[0:25]

$$\begin{cases} 0,1: x, y \\ 2,3: w, h \\ 4: Confidence\ [IoU] \\ 5\sim := \Pr(class/Obj) \end{cases}$$

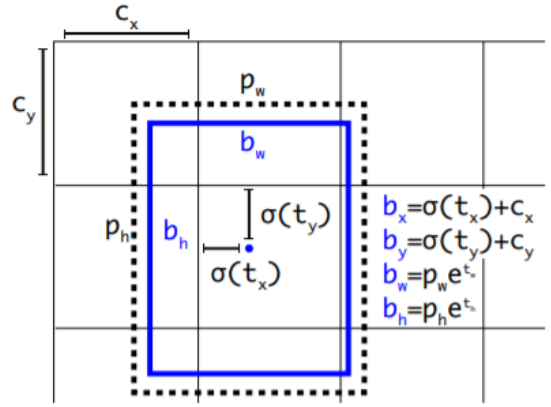


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

- Code:

```
box_xy = K.sigmoid(feats[..., :2])
box_wh = K.exp(feats[..., 2:4])
box_confidence = K.sigmoid(feats[..., 4:5])
box_class_probs = K.softmax(feats[..., 5:])

# Adjust preditions to each spatial grid point and anchor size.
# Note: YOLO iterates over height index before width index.
box_xy = (box_xy + conv_index) / conv_dims
box_wh = box_wh * anchors_tensor / conv_dims
```

II. One Stage Detection

E. Yolo V2 [2016, Joseph]

E3. Problems

- Better for small & crowded objects.

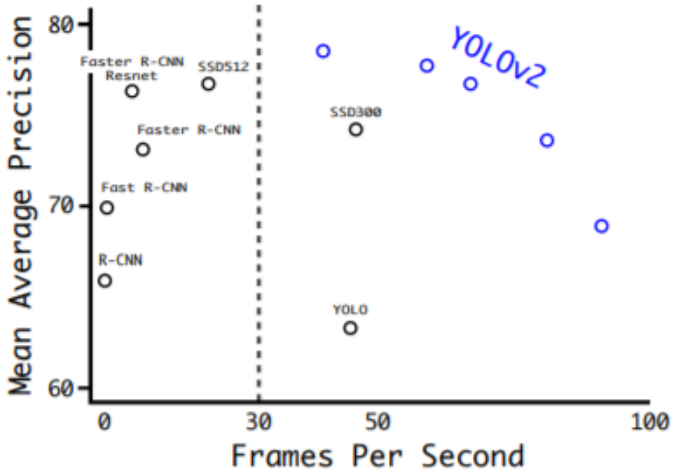


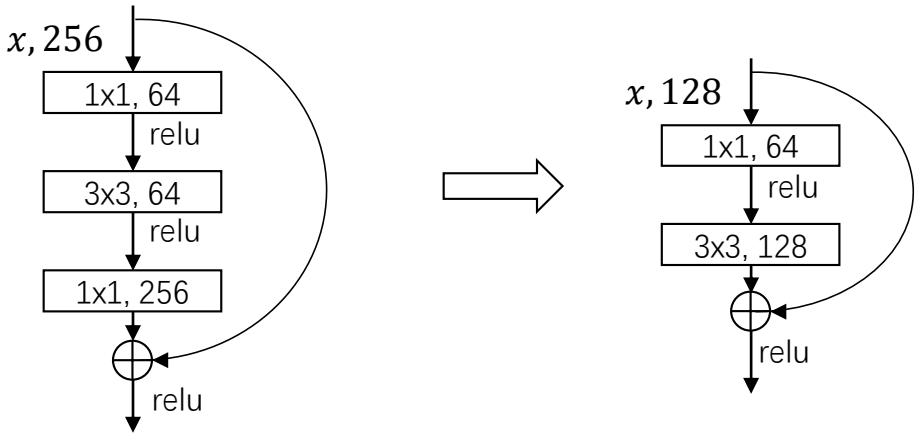
Figure 4: Accuracy and speed on VOC 2007.

II. One Stage Detection

F. Yolo V3 [2018, Joseph]

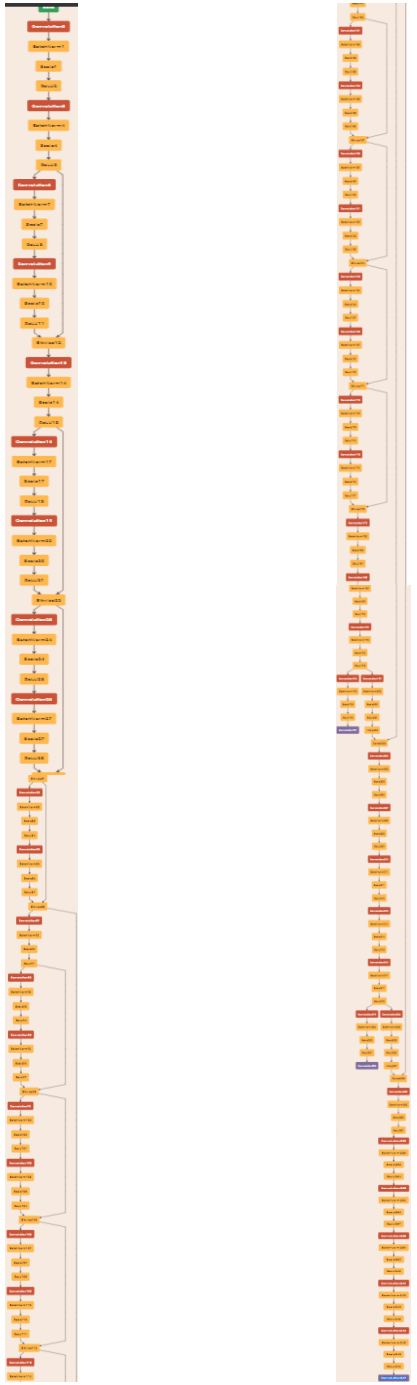
F1. Improvements

F1.1: New structure



F1.2: Multiscale Structure

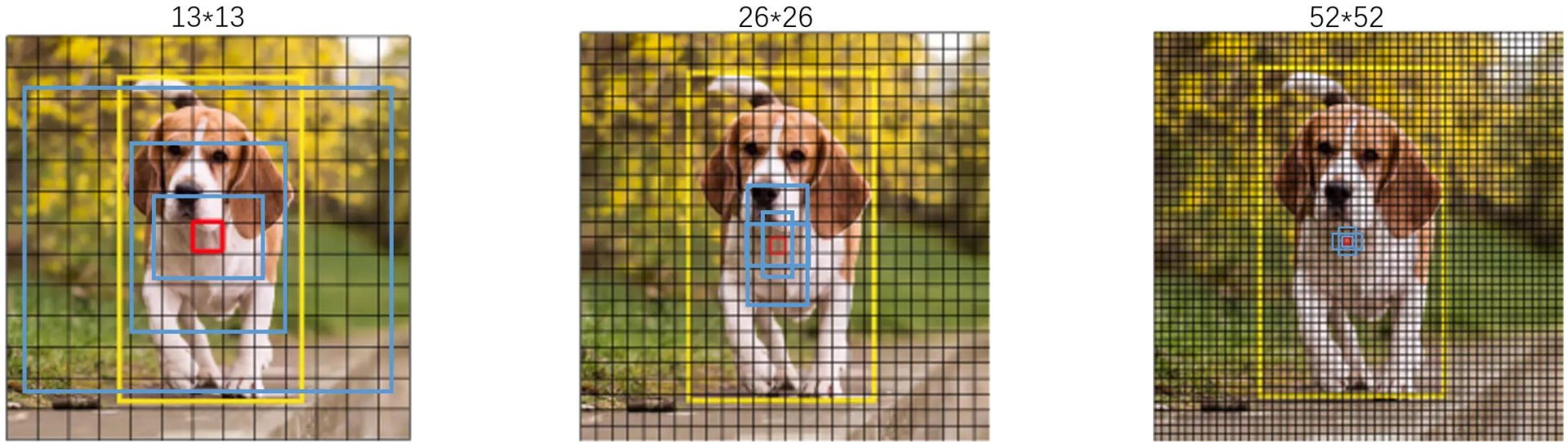
- 3 scales; 3 anchors per scale per grid
- /32, small scale (13 x 13) —> large anchor
- /16, mid scale (26 x 26) —> medium anchor
- /8, large scale (52 x 52) —> small anchor



II. One Stage Detection

F. Yolo V3 [2018, Joseph]

F1. Improvements



F1.3: Change Classification :

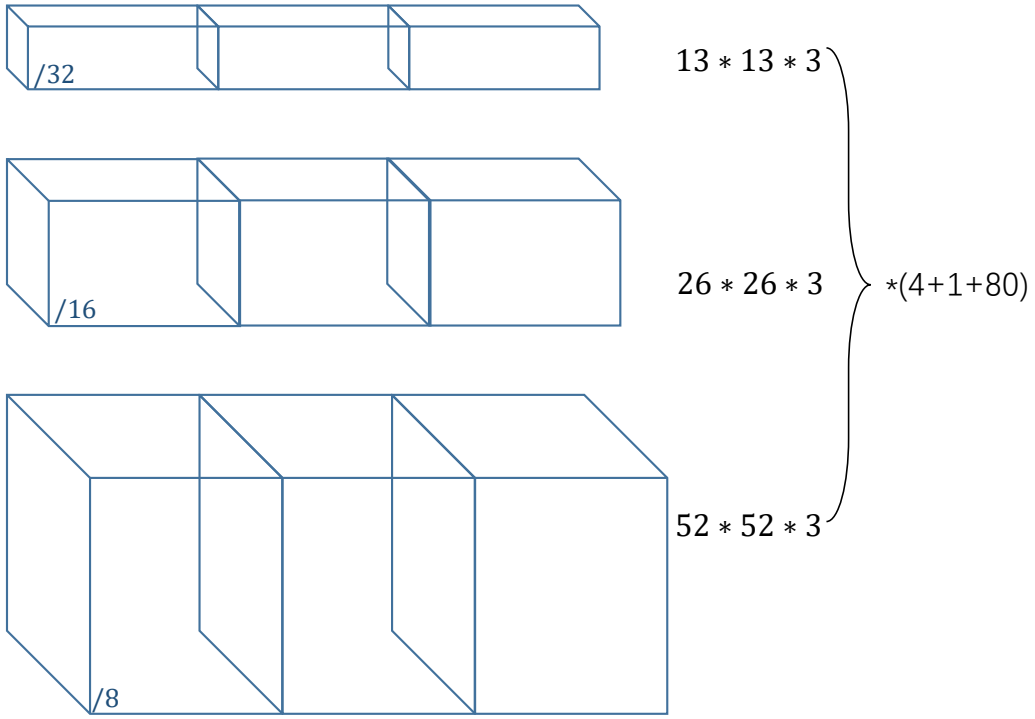
- 80 classes, from softmax —> logistic

II. One Stage Detection

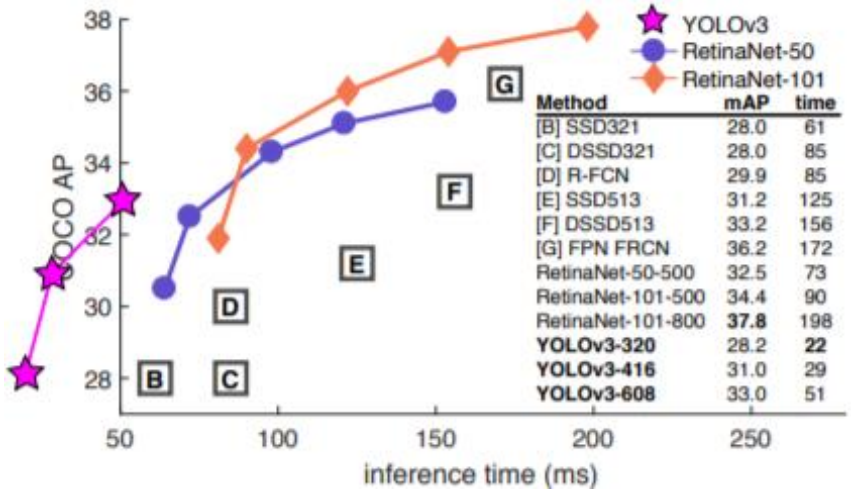
F. Yolo V3 [2018, Joseph]

F2. Summary

F2.1: Output



F2.2: Performance

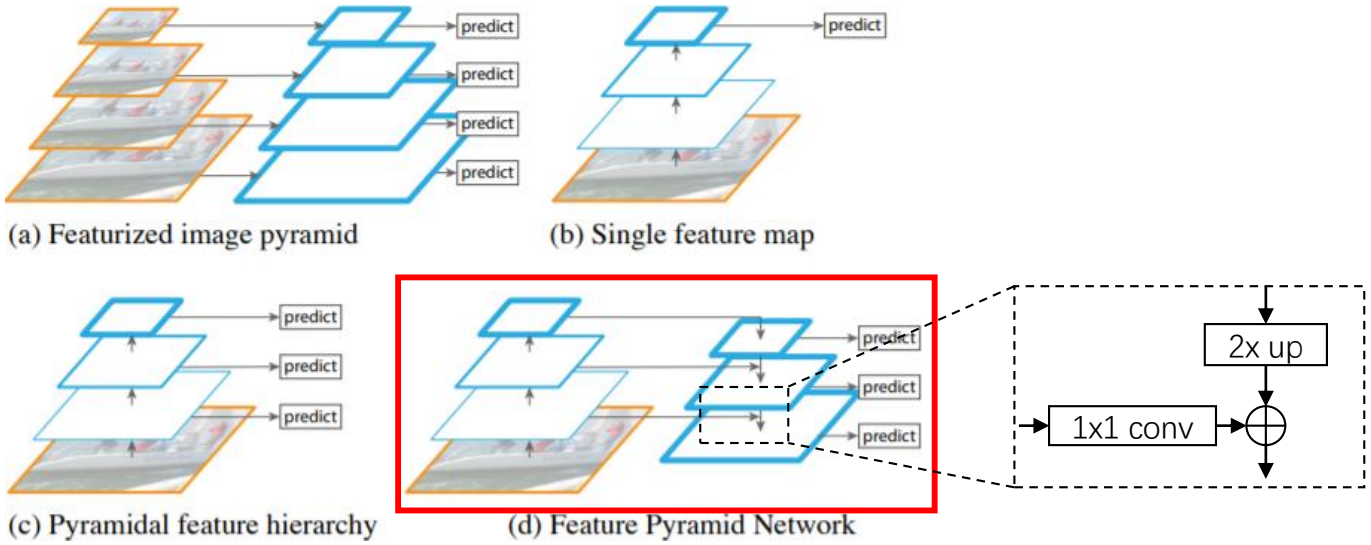


Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

II. One Stage Detection

F. Yolo V3 [2018, Joseph]

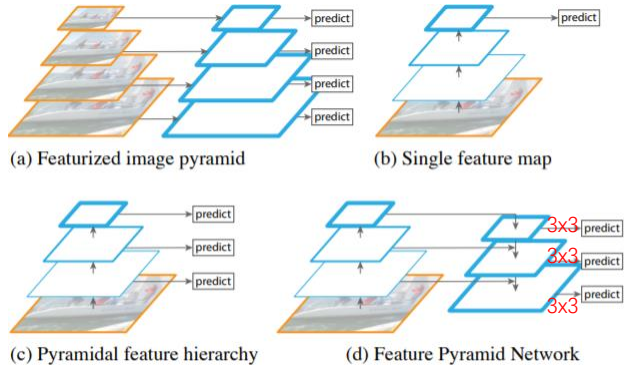
F3. FPN Net [2017, Lin]



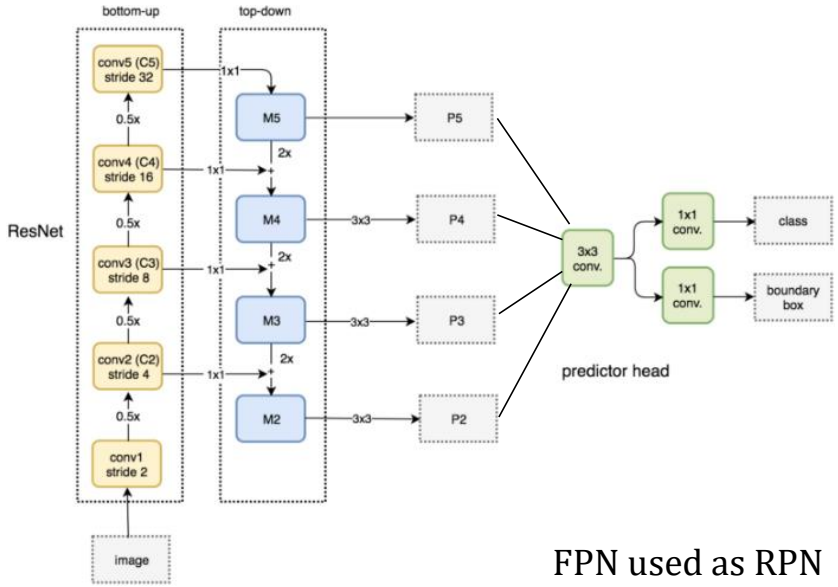
II. One Stage Detection

F. Yolo V3 [2018, Joseph]

F3. FPN Net [2017, Lin]



- **Pros:**
  1. Lower layers have accurate localization info;
     Higher layers have ample semantic info
     FPN combine them together. Like U-Net
  2. Feature detector: worked as a part in a whole bigger network.
- **Details:**
  1. 3x3convolution is appended on each merged map to generate the final feature map to reduce the aliasing effect
  2. Feature dimension at output is 256.
  3. When used in FPN. P2-P6. Anchor areas: $32^2, \sim, 512^2$ per scale, with ratio {1:2,1:1,2:1}, 15 in total. And, predictor head is shared.



FPN used as RPN

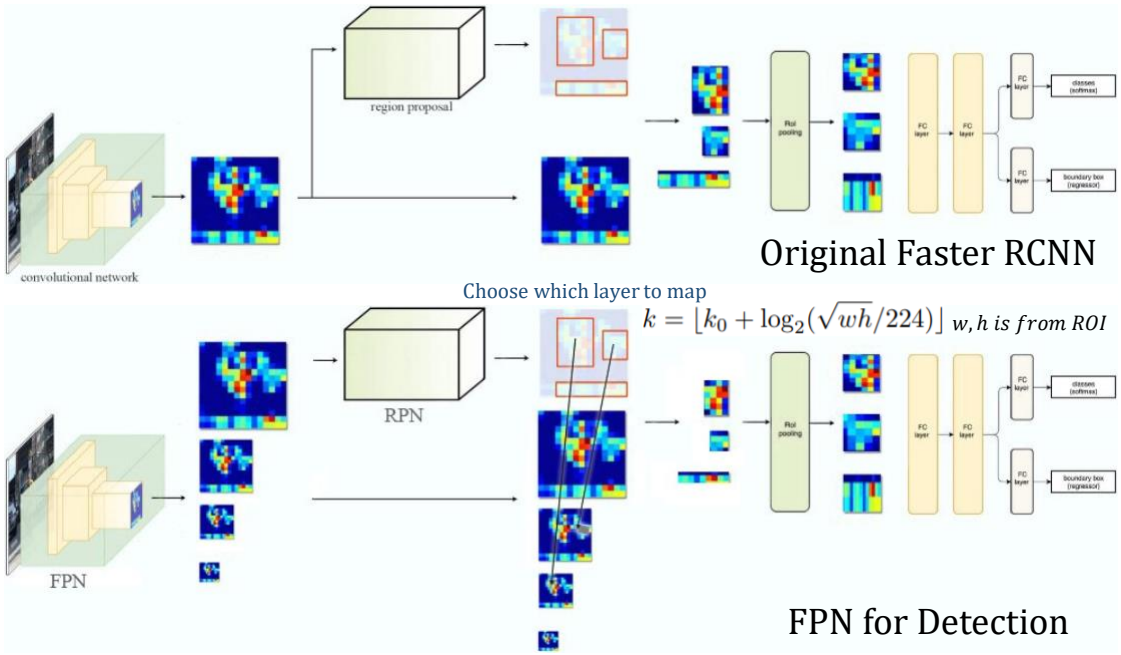
II. One Stage Detection

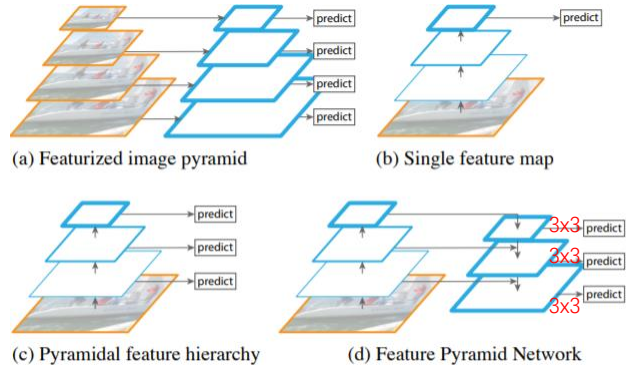
F. Yolo V3 [2018, Joseph]

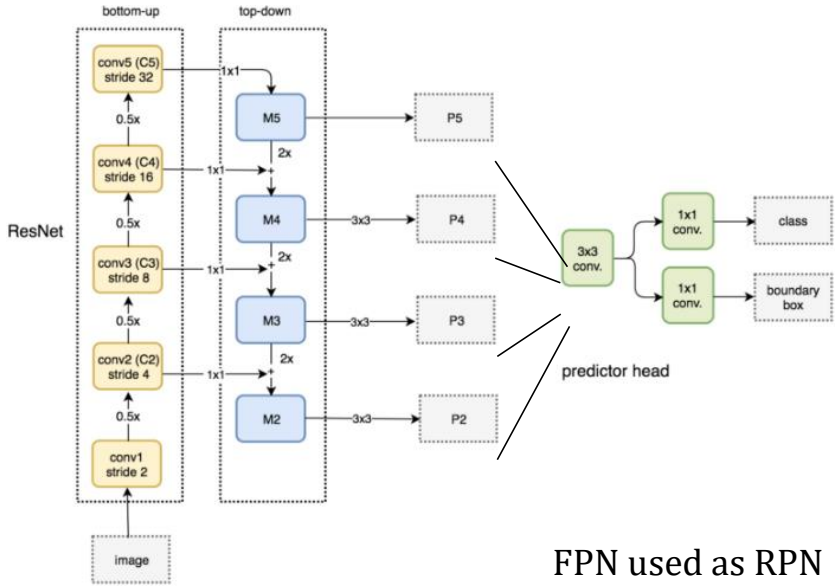
F3. FPN Net [2017, Lin]

- **Details:**

  4. When used in faster rcnn:







II. One Stage Detection

G. RetinaNet [2018, Lin]

- **Structure:**

  Resnet + FPN + FCN :

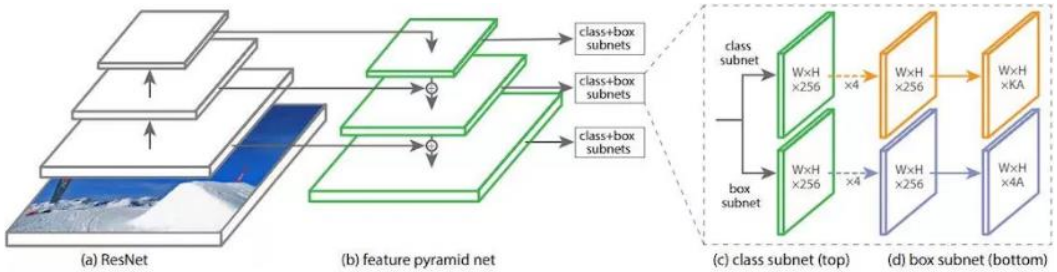


Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [20] backbone on top of a feedforward ResNet architecture [16] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN [20] while running at faster speeds.

- **Focal Loss:**

  Q. Why one-stage performs worse than two stage?

  A. 1. Because neg/pos samples are extremely unbalanced

  2. Gradient is dominated by easy samples.

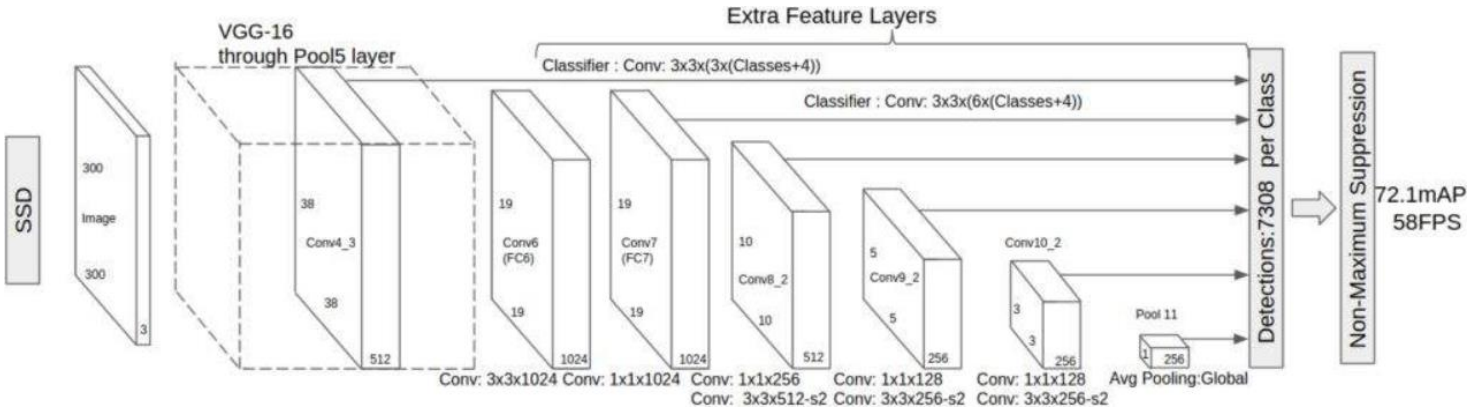
  S. We can use FOCAL LOSS to solve it.

III. Other Methods

III. Other Methods

G. Other Methods

- SSD Series: (2015, Liu, another one-stage, also popular!)



- Anchor free net is a trend
  - **Cornet, ExtremeNet**
  - **CenterNet: Objects as Points / Keypoint Triplets for Object Detection**
  - **CenterNet-Lite**