**Region proposal net work**

- Cho ảnh qua backbone CNN được feature map (thường feature map sẽ giảm đi còn khoảng 20 x 20 , 15 x 15, 30 x 30 )

- tạo ra các anchor (mỗI pixel trong feature map tương ứng vớI 1 điểm anchor trong ảnh ban đầu)

- tạo ra các anchor box (vớI mỗI anchor tạo ra tổng cộng 9 anchor box tương ứng vớI tỉ lệ 1:1, 1:2, 2:1 + size: thường để (2,4,6) \* img\_size[0]/feature\_size[0])

- tạo ra các pos\_anchor và neg\_anchor

- lấy n\_sample tổng cộng gồm pos\_anchor: neg\_anchor tỉ lệ 1:1 (pos thiếu thì bù bằng neg)

- slide a small network over the feature map (dùng chung 1 small net cho toàn bộ không gian của feature map)

+ input là một sliding window n\*n của feature map

+ mỗI sliding window, dự đoán tốI đa k region proposal (thường k = 9)

+ mỗI sliding window được map vào lower-dimension feature

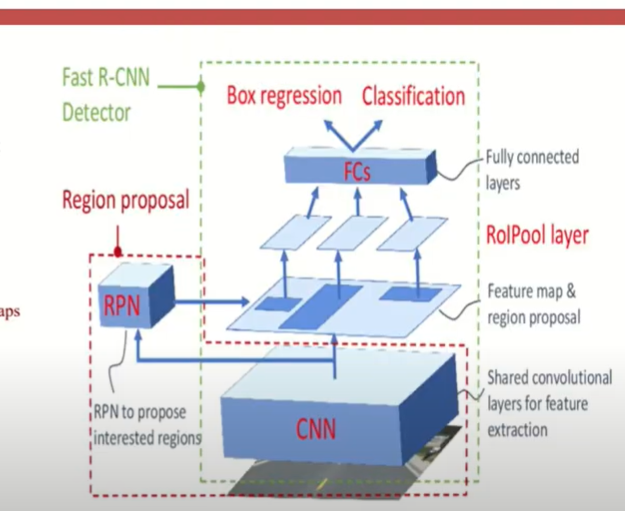
+ sau đó cho qua 2 fullly connected layers:

. box regression (dự đoán offset) (output gồm 4k giá trị tọa độ ở mỗI anchor)

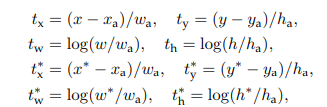
(offset là giá trị vị trí tương đốI của pred bbox vớI gt box, dùng để chỉnh pred cho về gần vớI gt)

. box classification (dự đoán xem vùng đó chứa object vớI tỷ lệ bao nhiêu) (output gồm 2k giá trị xác suất có object của vùng)

+ Kiến trúc tổng của small net ( n\*n convolution layers + (2 fully connected layer))



- VớI bounding box regression, sử dụng tham số sau (để dự đoán offset):



vớI:

- x : predict

- x\_a:

- x\*: gound truth box

- tương tự vớI các tham số còn lạI

Sau khi lấy đc region proposal từ RPN thì dùng roi\_pooling (các region proposal không giống nhau về kích thước) ở feature\_map để lấy vùng trên feature\_map dùng cho phân loạI vùng đó là đốI tượng nào

**Train RPN:**

- khi train RPN, ta không dùng nhiều ảnh mà chỉ dùng 1 ảnh duy nhất và dùng các anchor của 1 ảnh để tạo batch

- anchor box được coi là pos nếu tạI anchor đó, box có iou cao nhất vớI gt hoặc box có iou >= 0.7

- anchor box được coi là neg nếu tạI anchor đó, box có iou < 0.3

- các anchor box không là pos và không là neg thì không dùng để train

- vấn đề ở đây là vớI mỗi mini batch lấy trong 1 ảnh nó bao gồm rất nhiều pos và neg example anchor, có thể tốI ưu loss function cho tất cả các anchor đó nhưng vớI việc có quá nhiều neg example thì nó sẽ thiên về các neg example

- thay vào đó:

- lấy khoảng 1/10 số anchor hợp lệ trong ảnh (lũy thừa của 2 là tốt) để tính loss function cho mini-batch (lấy được nhiều pos là do tạo được nhiều điểm anchor trong ảnh , khoảng cách giữa các anchor nhỏ và scale của anchor boxes cũng sẽ có khoảng rộng hơn)

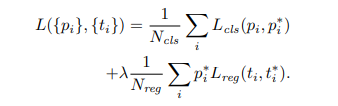
- Các anchor được lấy mẫu vớI tỉ lệ 1 pos : 1 neg

- nếu đủ hoặc thừa pos thì chỉ lấy đủ pos và lấy neg tương ứng

- nếu thiếu pos thì lấy tất cả pos , lấy đủ neg + thêm neg để bù vào phần thiếu của pos

- làm sao để tổng số sample luôn không đổI là đc

- Hàm loss:



+ i là thứ tự của anchor ở trong mini batch

+ L\_cls tính toán log loss giữa giá trị dự đoán và giá trị thực tế, p\_\* nhận giá trị 1 nếu anchor box là pos, 0 nếu anchor box là neg

+ L\_reg là Robust loss, có 2 loạI đc đề cập là Huber Loss và log-cosh loss, chỉ kích hoạt khi p\_\* là positive anchor (nếu là neg anchor thì nhân vớI 0 = 0) (ở đây chỉ tính sai số vớI positive anchor)

+ t là vector có 4 tham số đạI diện cho giá trị tọa độ của bboxes

+ N\_cls = batch\_size

+ N\_reg = số điểm anchor (cả 2 giá trị N đều để chuẩn hóa lỗi)

+ lambda để default là 10, giá trị lambda lấy trong [0.1, 1, 10, 100], kết quả không thay đổI vớI khoảng lambda rộng như ở trên

