Object Detection consists of two sub tasks (i) Object Localization (ii) Object CLassification.
For implementation we need to break it into three steps instead -
[A] Region Proposal Generation. Each of them called Region of Interest (RoI).
[B] Extracting feature maps for each region proposal. Lets call it Fmap RoI.
[C] Classifying each Fmap RoI as object classes.
Here step [B] is necessary because the images or crop from a image requires to be broken down to feature space
which we are calling feature maps. Each of the convolutional filter can be responsible for extracting different image features.
Like convolutional filters responsible for edge filtering can result in edge images kind of the the canny detector output.

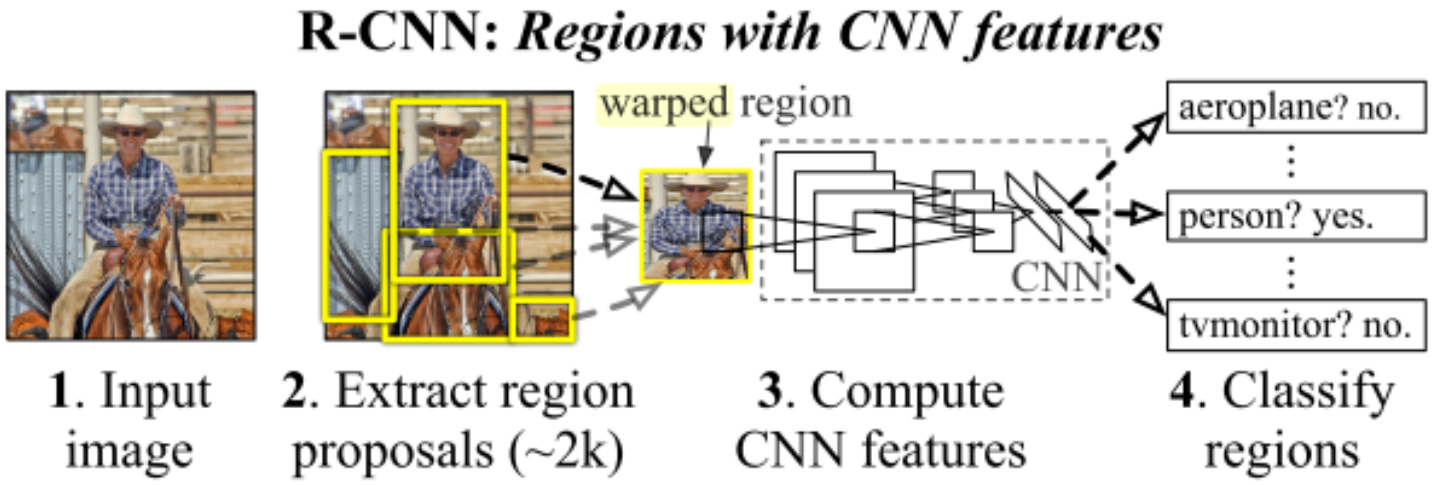
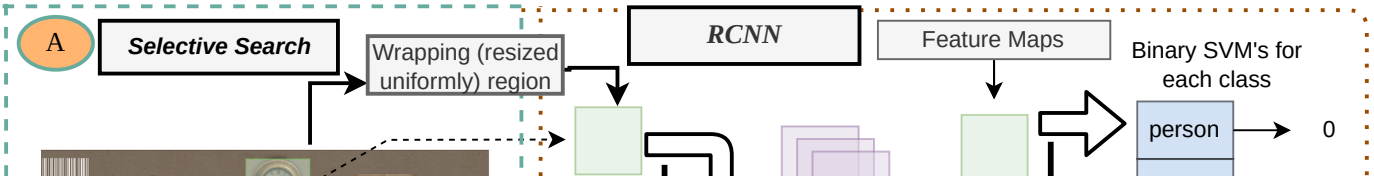


Figure : From RCNN paper

The Alorithm :

1. [A] Slective Search with 2k proposals per image. [Objectness[1] , Selective Search[2]]
2. [A] Wrapping the images [resizing with bilinear/nearest sampling shouls work here]
3. [B] Using a CNN based classifier network for feature extraction
   -> Pre training with ICSLV2012 images with 200 classes lr 0.01 on Caffe implementation of a CNN based architecture.
   -> Fine tuning with lr 0.001 with wrapped region proposal with N + 1 class. N is no of object class 1 is background
   -> Oversampling positive classes 32 positive sample + 96 neg sample per mini-batch
   -> During finetuning a region proposal with more than 0.50 IoU with a positive sample was atken as positive.
   -> During Feature Extraction the last layer is removed and features from an internal dense layer is collected. i.e 1024 or 2048 featur
4. [C] Using binary classifier for per class classification from the extracted features
   -> One classifier is used for one class
   -> Hard negative mining [4] was very useful for fast convergence. It was possible to achieve results after just one pass over the data
5. [*] Fine tuning of localization was done by bbox regression using Ridge regressor with lambda = 1000 [detail below]

Regarding using the outputs of the classifiers after fine tuning instead of additionaly using SVM :
Quote from the paper's appendix B : "We tried this and found that performance on VOC 2007 dropped from 54.2% to 50.9% mAP. This performance drop likely arises from a combination of several factors including that the definition of positive examples used in fine-tuning not emphasize precise localization and the softmax classi- fier was trained on randomly sampled negative examples rather than on the sub "hard negatives" used for SVM training."

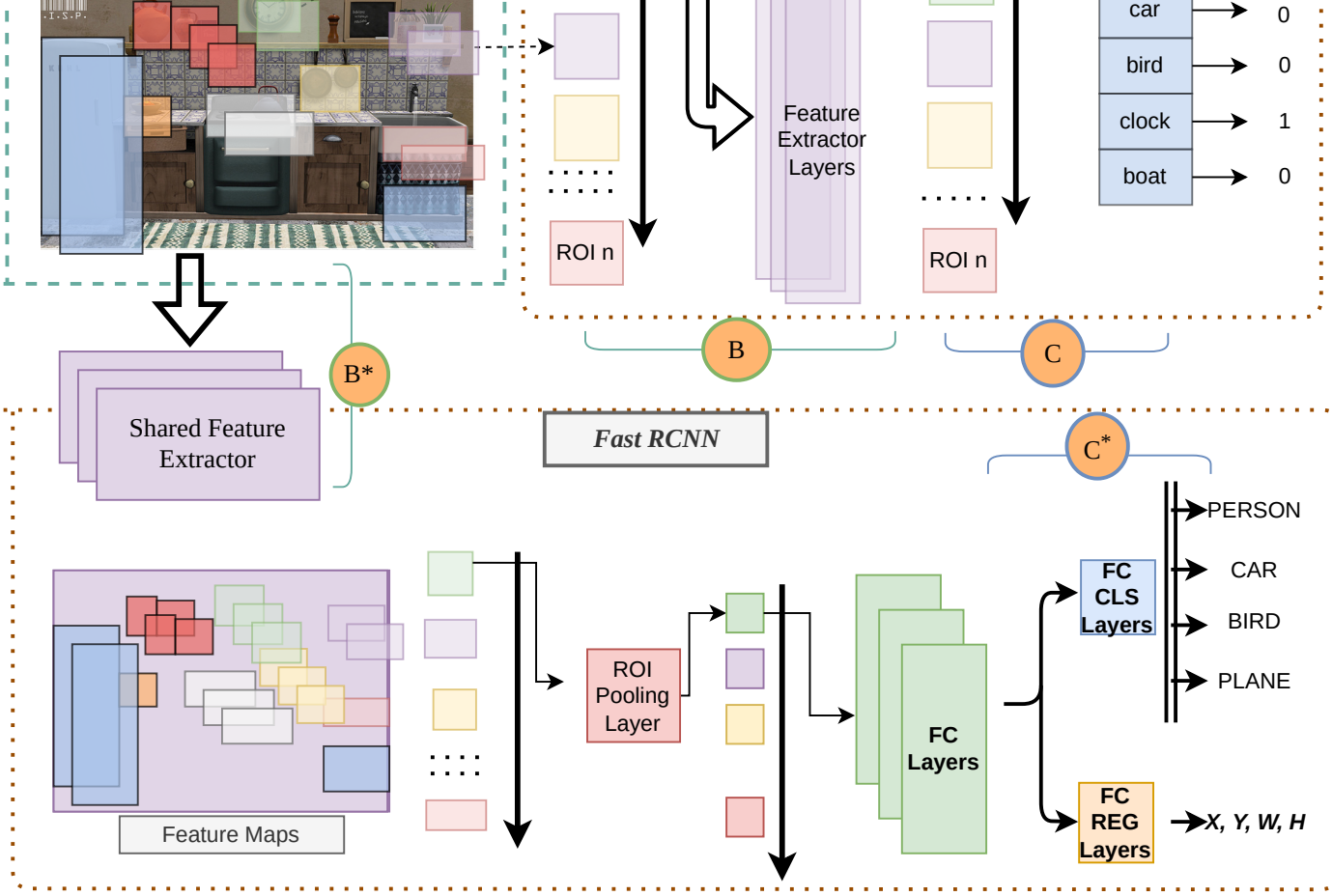


res

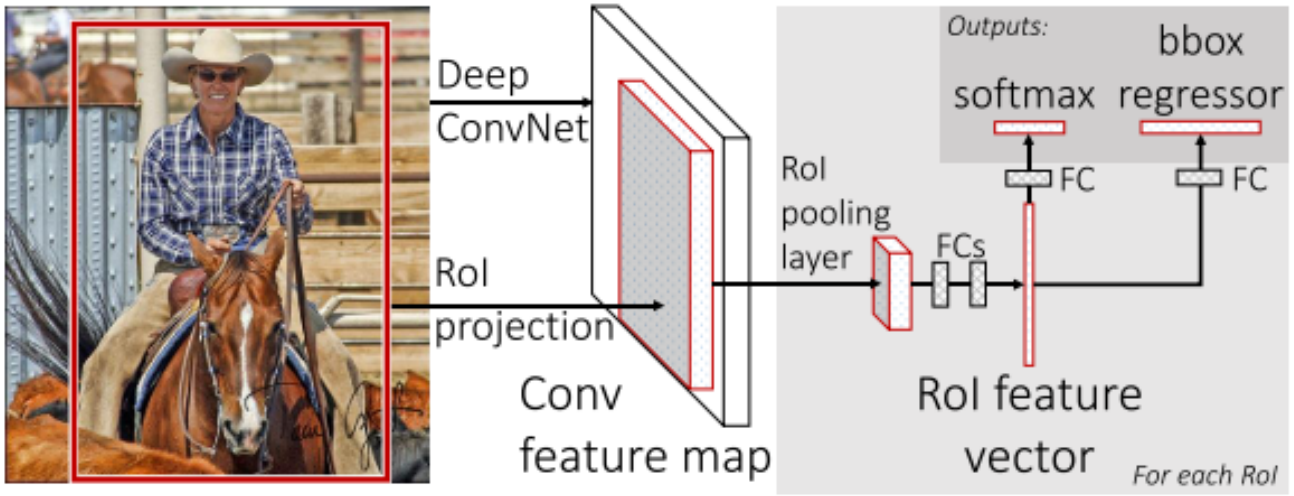
a.

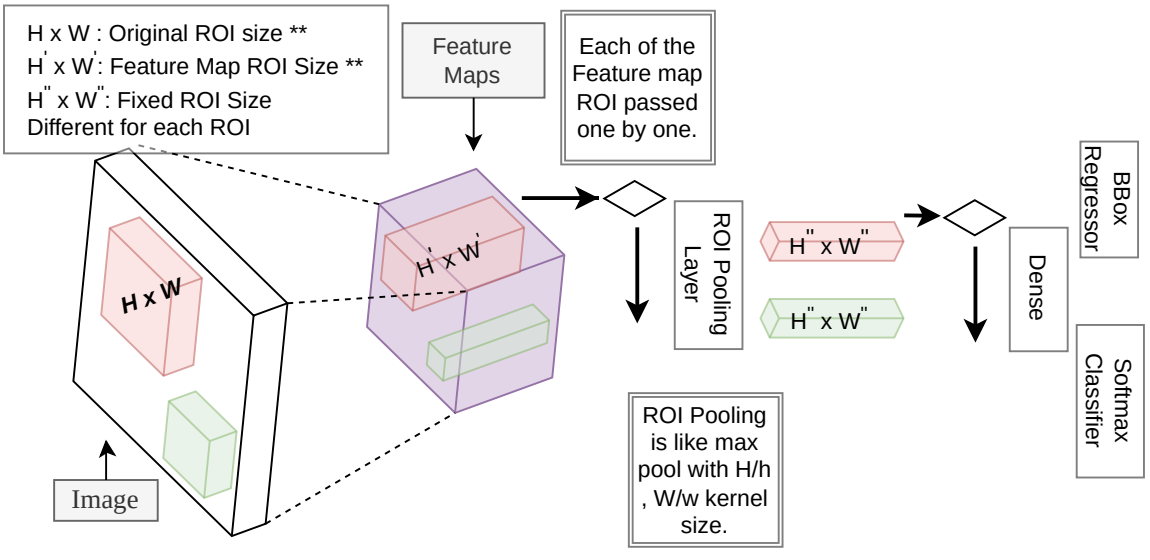
g does
bset of

Ground Truth - Gx , Gy, Gw, Gh
Predictions - Px, Py, Pw, Pz
$G_x = P_w d_x(P) + P_x$
$Gy = P_h d_y(P) + P_y$
$Gw = P_w \exp(d_w(P))$


Feature Extractor Layers
ROI n
ROI n
car
bird
clock
boat
0
0
1
0
B
C
B*
Shared Feature Extractor
Fast RCNN
C*
Feature Maps
ROI Pooling Layer
FC Layers
FC CLS Layers
FC REG Layers
PERSON
CAR
BIRD
PLANE
X, Y, W, H

fig

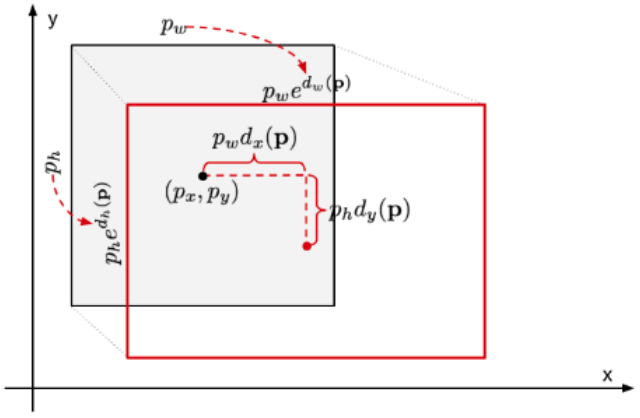

Deep ConvNet
RoI projection
Conv feature map
Outputs:
softmax
bbox regressor
FC
FC
RoI pooling layer
FCs
RoI feature vector
For each RoI


H x W : Original ROI size **
H' x W': Feature Map ROI Size **
H" x W": Fixed ROI Size
Different for each ROI
Feature Maps
Each of the Feature map ROI passed one by one.
H x W
H' x W'
Image
ROI Pooling Layer
H" x W"
H" x W"
Dense
BBox Regressor
Softmax Classifier
ROI Pooling is like max pool with H/h , W/w kernel size.

$G_h = P_h \exp(d_h(P))$.

$t_x = (G_x - P_x)/P_w$
$t_y = (G_y - P_y)/P_h$
$t_w = \log(G_w/P_h)$
$t_h = \log(G_h/P_h)$.

(ti - di(P))^2 + lambda * ||w||



: from https://lilianweng.github.io/posts/2017-12-31-object-recognition-part-3/

*** Getting rid of one by RoI input to the feature extractor.***

***

The Alorithm :

1.[A] Slective Search with 2k proposals per image. [Objectness[1] , Selective Search[2]]

2.[B] Using a CNN based classifier network for feature extraction

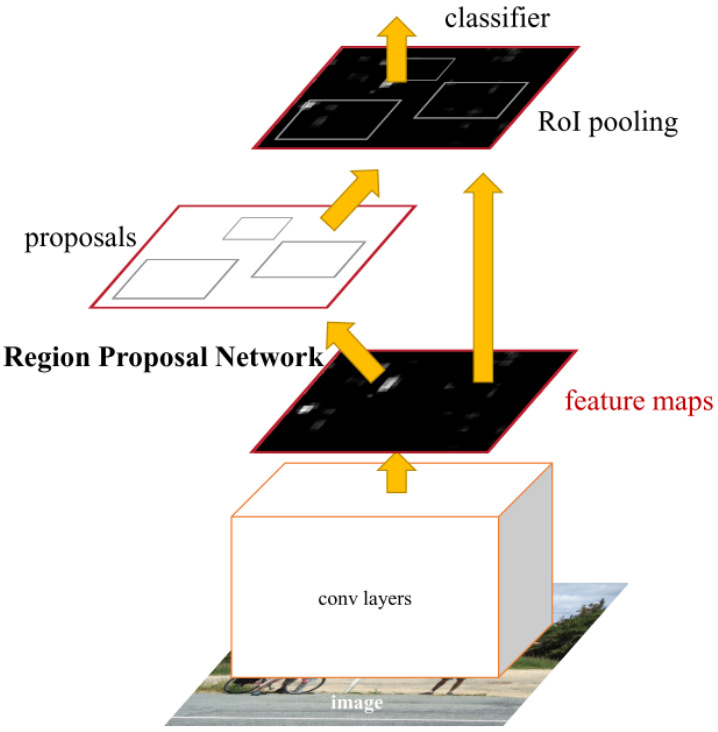
-> Pre training with ImageNet data on VGG16 (and two other smaller nets) images with 200 classes lr 0.01

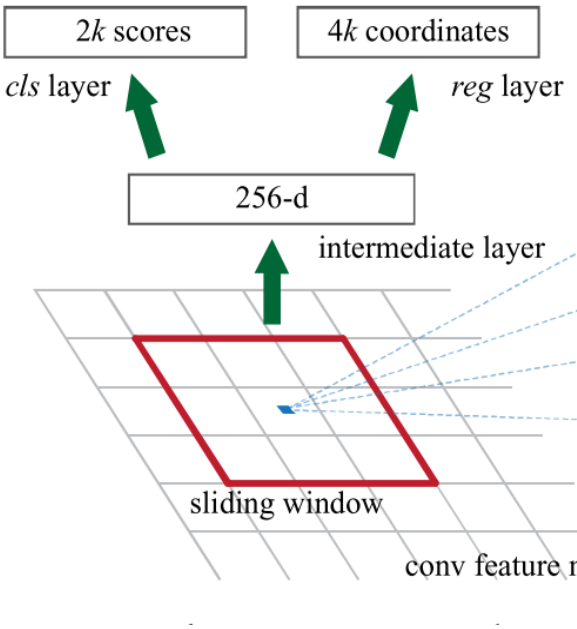
-> Fine tuning with lr 0.001 with wrapped region proposal with N + 1 class. N is no of object class 1 is background

-> Oversampling positive classes 32 positive sample + 96 neg sample per mini-batch

3.[B] The last max pool layer is replaced with a RoI Pooling layer for extracting fixed sized feature maps for every Fmap RoI so that it ca
layer (H=7, W=7 for VGG 16)

4. [C] The last fully connected layer is replaced with 2 sibling dense layer (i) one for classificatiob (ii) anther for bbox regression.





***** The selective search is replaced with Region Proposal Network that mostly shares weights with Feature extractor.***

***** The region proposals are further broken down to 3 different scales with 3 different aspect ratio called anchor boxes.***

The Alorithm :

1.[A] Region Proposal Network proposes possible box coordinates and objectness confidence at each pixel location at a sliding window r

2.[B] At each location box coordinates are predicted at 9 possible aspect ratio, so at each pixel location we can get 9 boxes with different
confidences instead of just one. It will make possible to detect two overlapping objects of different aspect ratio or scale. Please look at the
predictions on the image on top right. 3 aspect ration for big boxes , 3 for medium, 3 for small.

2.[B] Using a CNN based classifier network for feature extraction

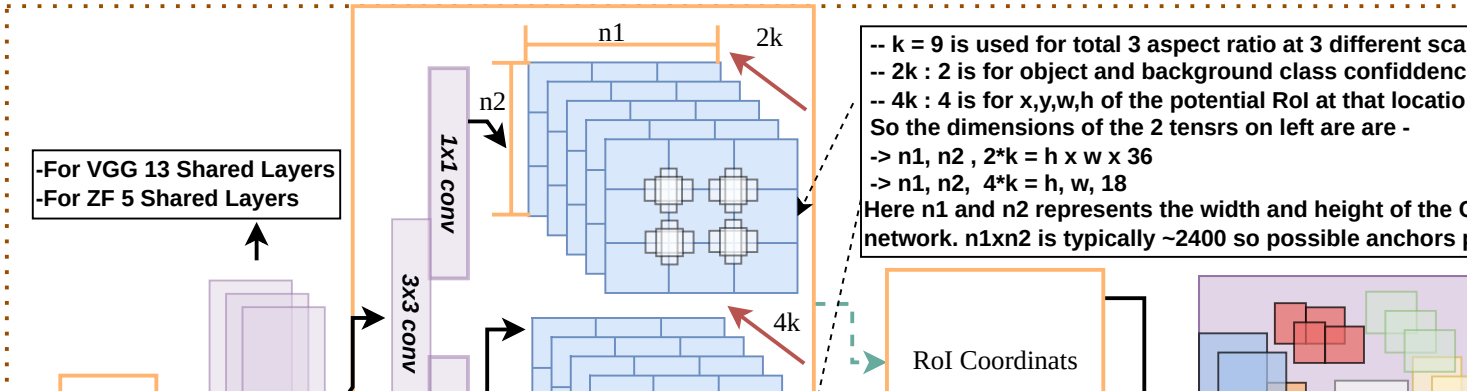
-> Pre training with ImageNet data on VGG16 (and two other smaller nets) images with 200 classes lr 0.01

-> Fine tuning with lr 0.001 with wrapped region proposal with N + 1 class. N is no of object class 1 is background

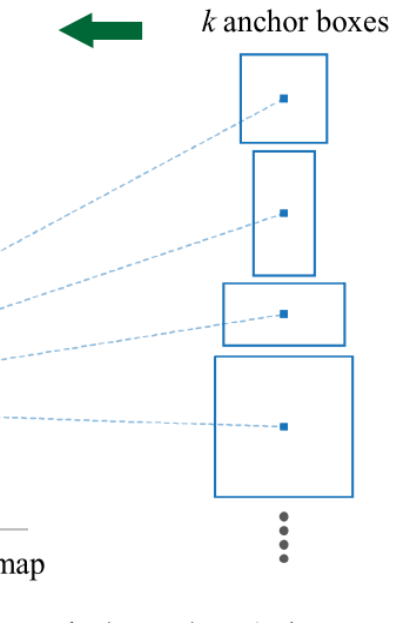
-> Oversampling positive classes 32 positive sample + 96 neg sample per mini-batch

3.[B] The last max pool layer is replaced with a RoI Pooling layer for extracting fixed sized feature maps for every Fmap RoI so that it ca
into the first dense layer (H=7, W=7 for VGG 16)

4. [C] The last fully connected layer is replaced with 2 sibling dense layer (i) one for classificatiob (ii) anther for bbox regression.



an fit into the first dense

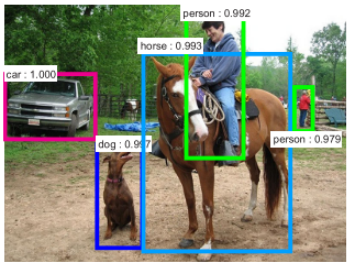


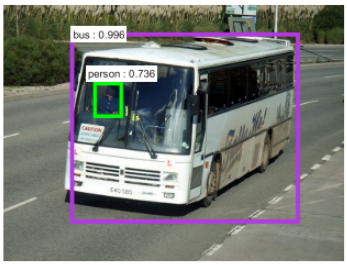
map

manner.

e

an fit





les.
e score
n.

CNN output of Region Proposal
per image 9*2400

Same as

Image Input

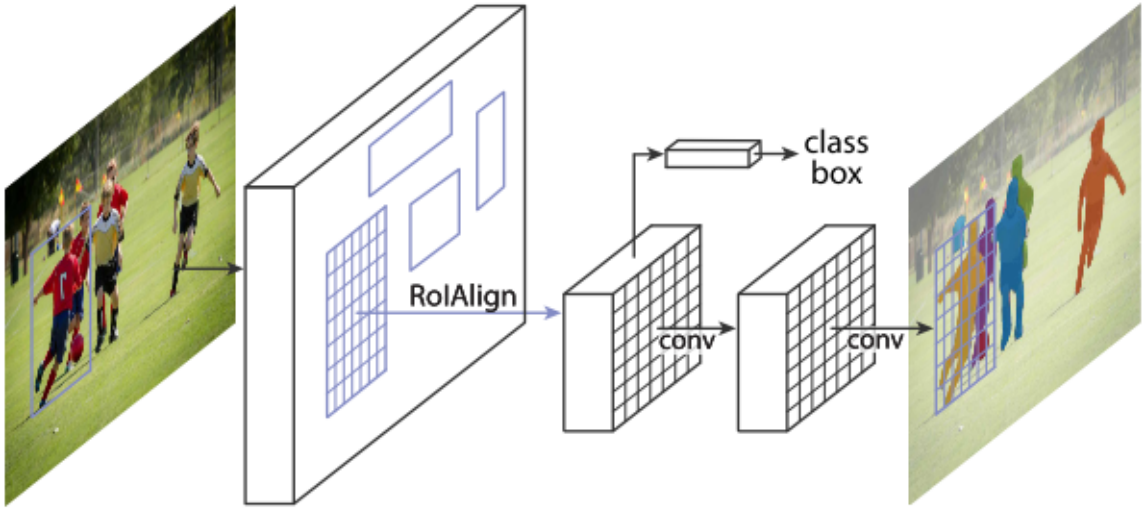
Feature Extractor CNN Layers

1x1 conv

Fmap RoIs

Feature Map

Since the RPN works as a sliding window manner at each pixel locations it is implemented naturally by using conv layers.



9

Anchor was us
The used 800 a
same 9 boxes a
capable of pro
-> For Multine
fully connec
-> For Faster R

Since the RPN works as a sliding window manner at each pixel locations it is implemented naturally by using conv layers.

-For VGG 13 Shared Layers
-For ZF 5 Shared Layers

(no of obj + 1) * k

1x1 conv

3x3 conv

(x, y, w, z) * k

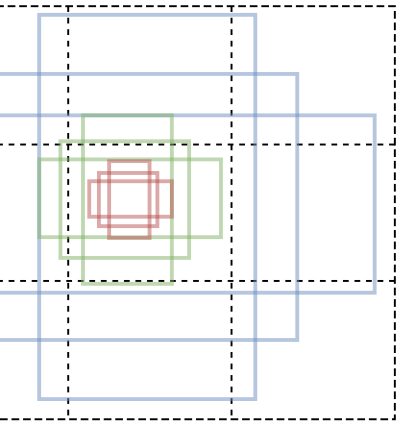
1x1 conv

At each pixel location k (here k=3) windows of different scales an aspect ratio is predicted

Feature Extractor CNN Layers

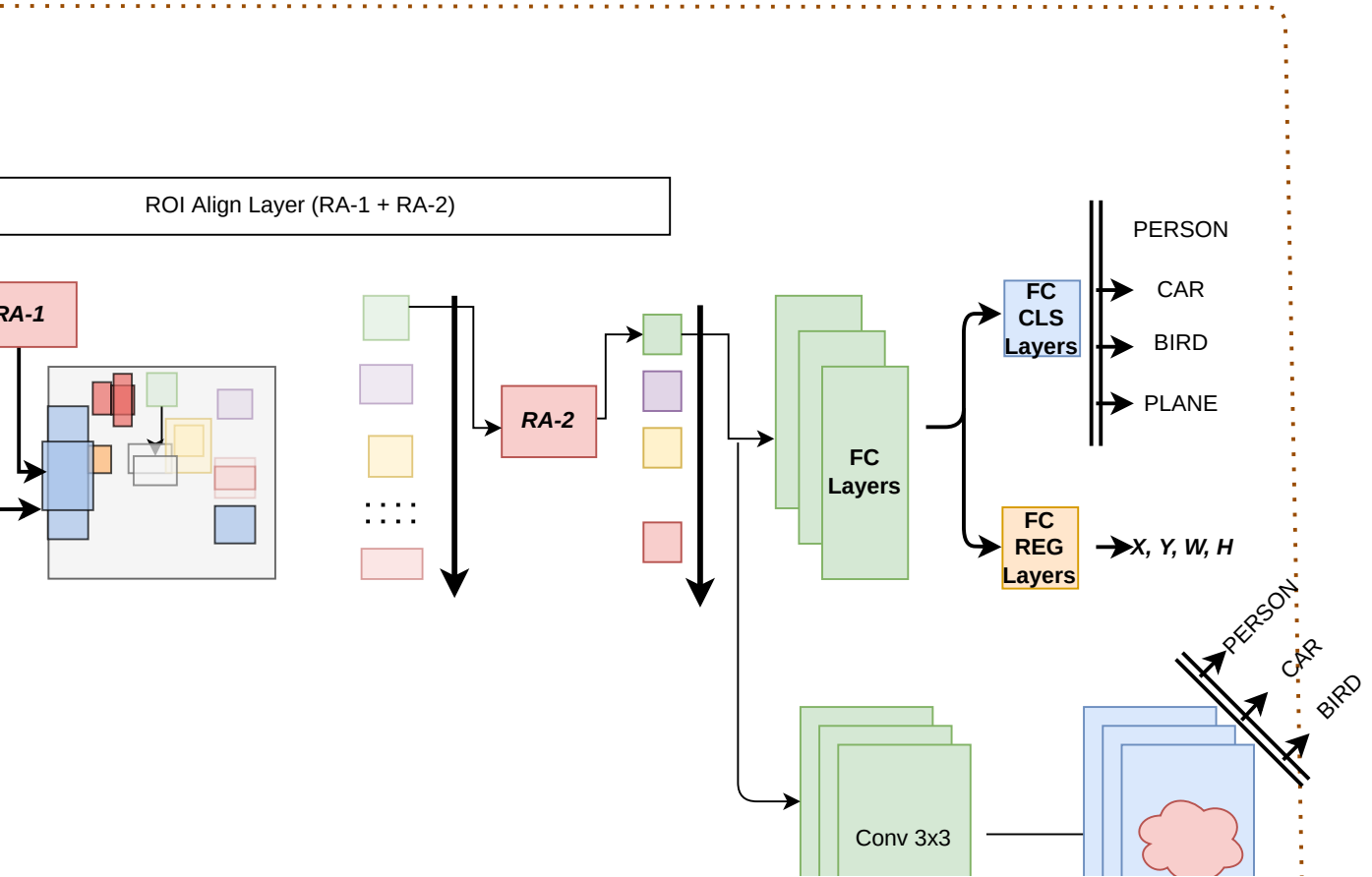
Feature Map

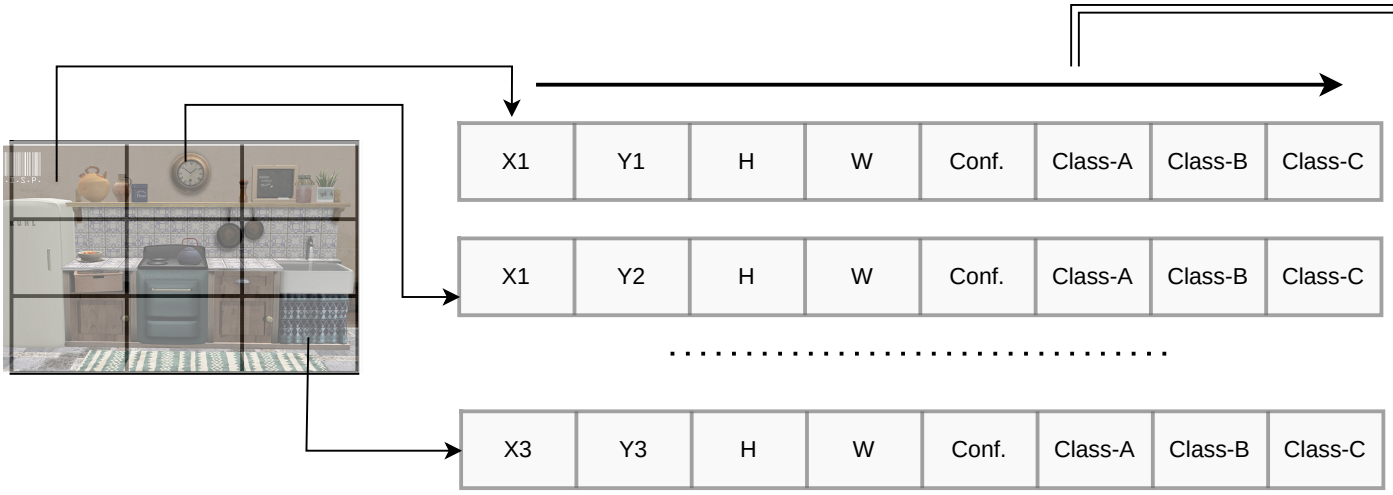
Fast
RCNN

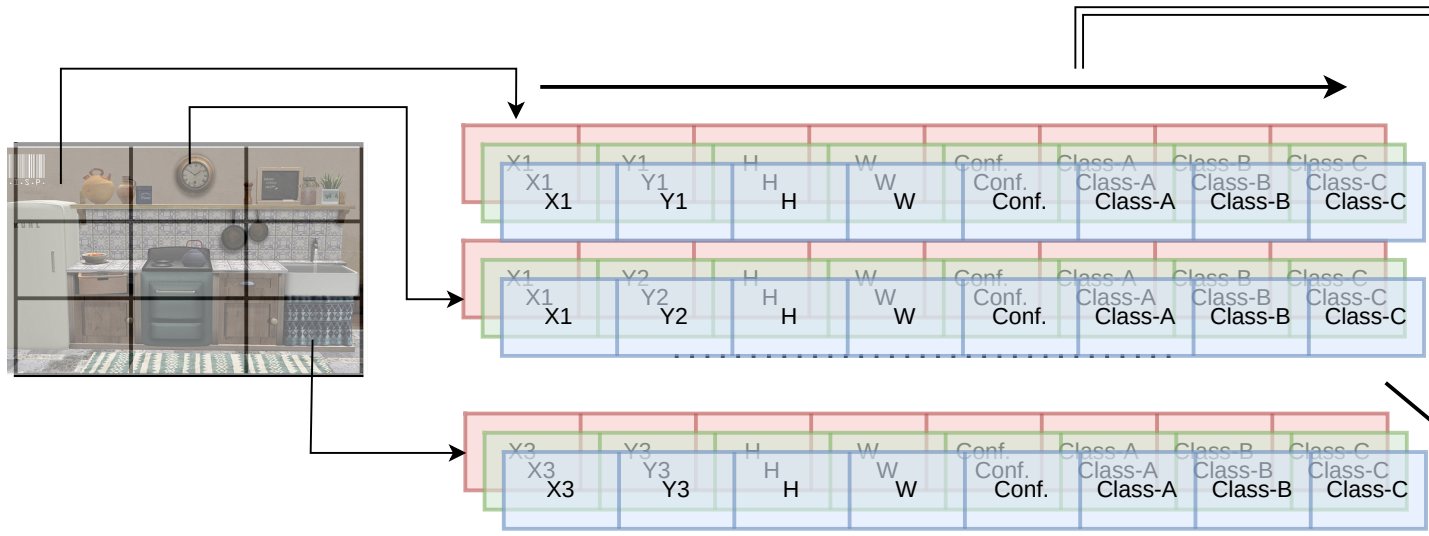


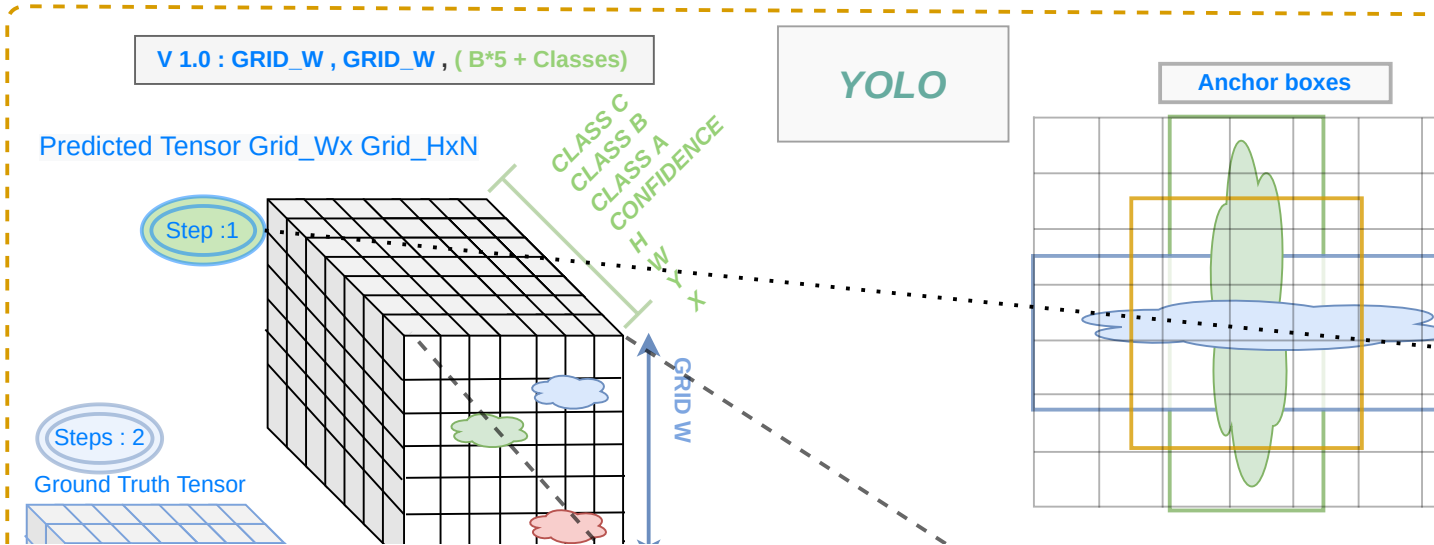
possible anchor boxe at one pixel locations

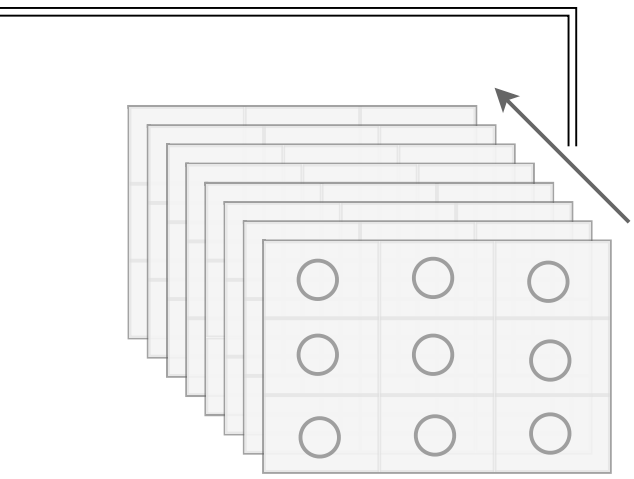
ed by Multibox[] but they were not tanslation invariant.
nchor boxes different locations of the image, faster rcnn used
t every locations
ducing (w*h=~2400) ~2400 x 9 RoI per image at maximum.
et the shape of tensor for RoI prediction was 4+1* 800 dim.
cted layer
RCNN it reduced to 4+2*9 convolutional output layer

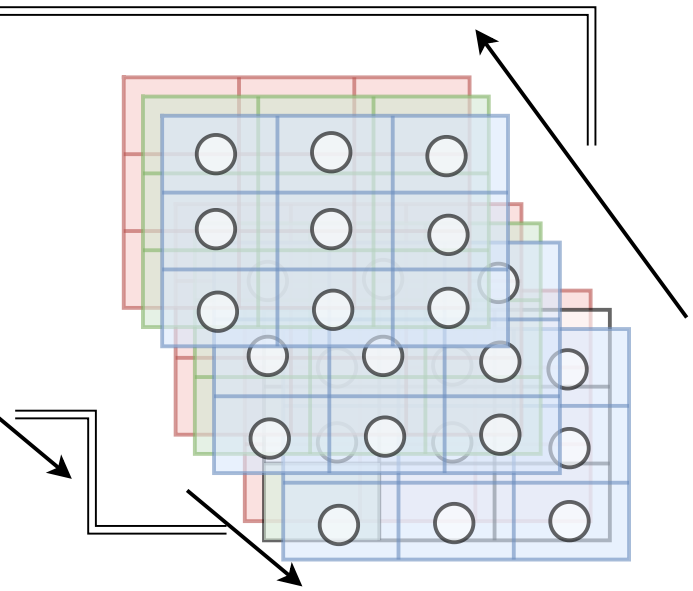


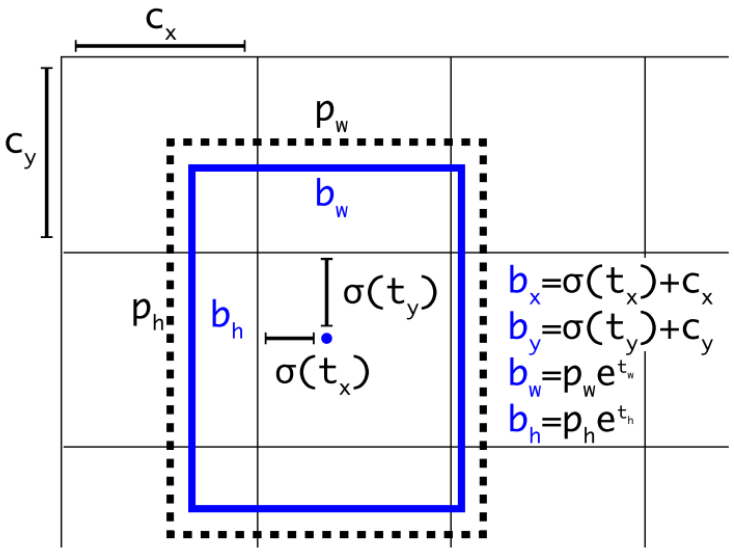

X1 Y1 H W Conf. Class-A Class-B Class-C
X1 Y2 H W Conf. Class-A Class-B Class-C
X3 Y3 H W Conf. Class-A Class-B Class-C


X1 Y1 H W Conf. Class-A Class-B Class-C
X1 Y2 H W Conf. Class-A Class-B Class-C
X3 Y3 H W Conf. Class-A Class-B Class-C


V 1.0 : GRID_W , GRID_W , (B*5 + Classes)
YOLO
Anchor boxes
Predicted Tensor Grid_Wx Grid_HxN
CLASS C
CLASS B
CLASS A
CONFIDENCE
H
W
Y
X
Step :1
GRID W
Steps : 2
Ground Truth Tensor







V 2.0 & 3.0 :GRID_W , GRID_W, BOX , (5 + Classes)

Predicted Tensor Grid_W x Grid_H x Box x N

GRID H

GRID W

Anchor BOX =2

P(Class C| Object)

P(Class B | Object)

P(Class A | Object)

P(Class C)

P(Class B)

P(Class A)

Step :3

*P(Class) = P(Class|Object) * P(Object)*

At each grid cen
of the anc box fi
max probability

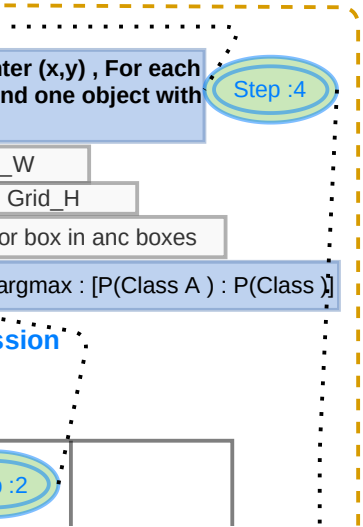
For x in Grid

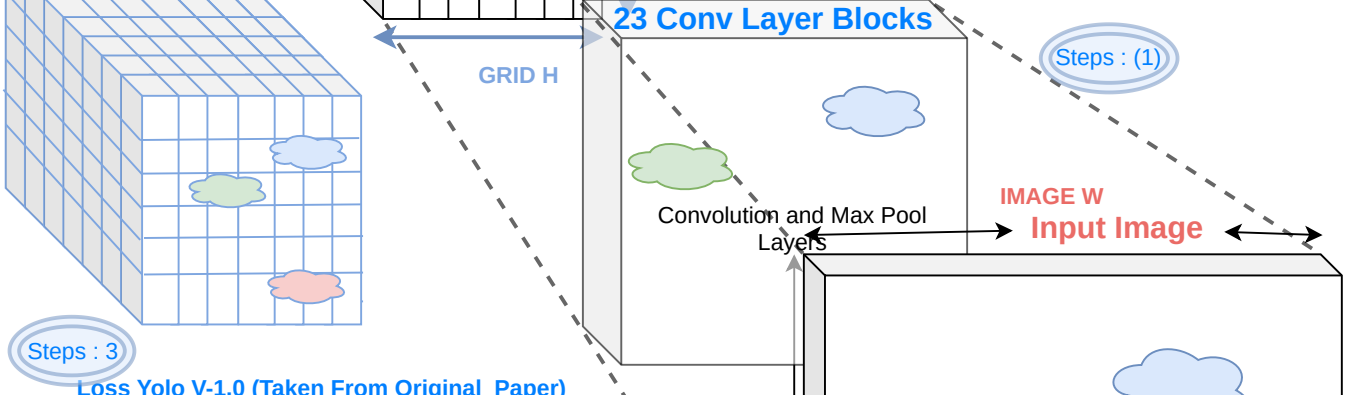
For y in

a Non Max Supres

b Removing low confidence box

Step


ter (x,y) , For each
nd one object with
Step :4
_W
Grid_H
or box in anc boxes
argmax : [P(Class A) : P(Class)]
ssion
:2



Loss Yolo V-1.0 (Taken From Original Paper)

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

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

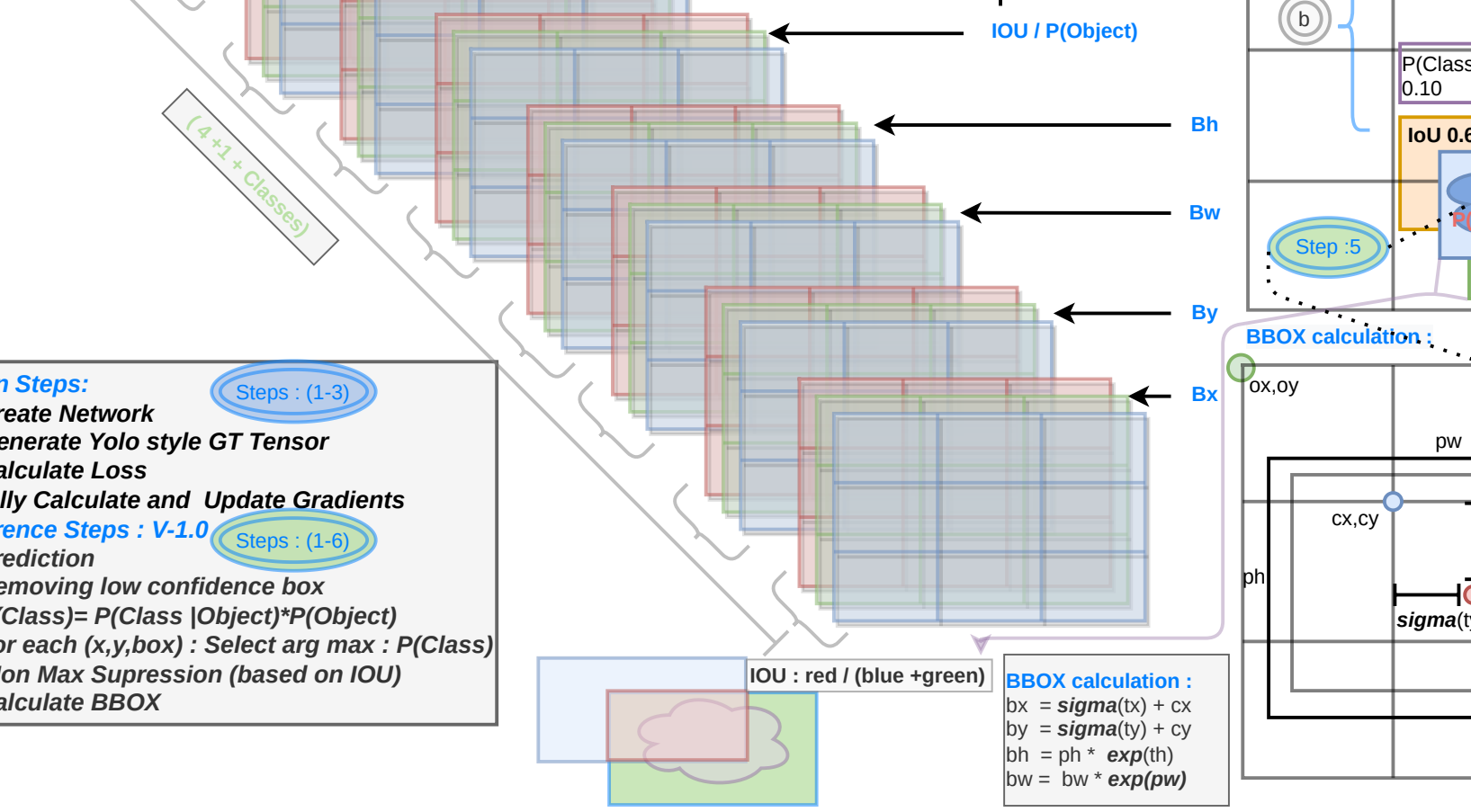
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

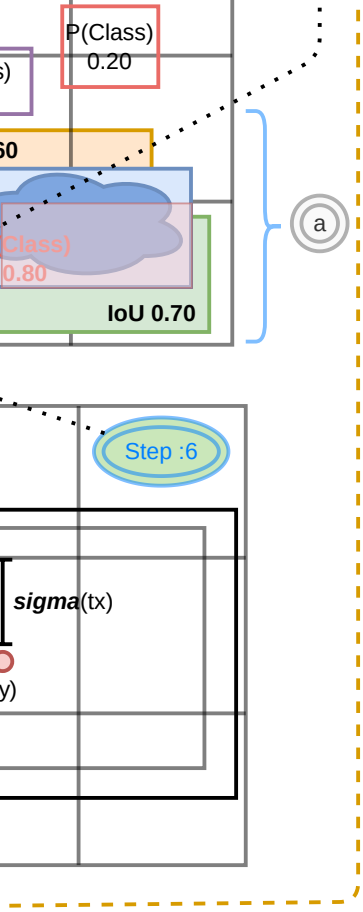
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

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

Train
1. Cr
2. Ge
3. Ca
Fina
Infer
1. Pr
2. Re
3. P(
4. Fo
5. N
6. Ca

Designed By - Irfan Mohammad Al Hasib
email - irfanhasib.me@gmail.com
Some figures are inspired from the original YOLO papers


IOU / P(Object)
Bh
Bw
By
Bx
(4 +1 + Classes)
Steps : (1-3)
Steps : (1-6)
n Steps:
reate Network
enerate Yolo style GT Tensor
alculate Loss
lly Calculate and Update Gradients
rence Steps : V-1.0
rediction
emoving low confidence box
(Class)= P(Class |Object)*P(Object)
or each (x,y,box) : Select arg max : P(Class)
lon Max Supression (based on IOU)
alculate BBOX
IOU : red / (blue +green)
BBOX calculation :
bx = sigma(tx) + cx
by = sigma(ty) + cy
bh = ph * exp(th)
bw = bw * exp(pw)
b
P(Class
0.10
IoU 0.6
Step :5
BBOX calculation :
ox,oy
pw
cx,cy
ph
sigma(t


P(Class)
0.20
IoU 0.70
a
Step :6
sigma(tx)