

**Deep Learning for Computer Vision**  
**Professor. Vineeth N Balasubramaniam**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Hyderabad**  
**Lecture No. 46**  
**CNNs for Object Detection II**

We will continue now with dense sampling methods for object detection.

(Refer Slide Time: 0:23)



**Exercise: Smooth L1 Loss**

Why is Smooth L1 loss less sensitive to outliers than L2 loss?

If the deviation of predicted output from ground truth is very high, squaring the difference explodes the gradient. This can happen in L2 loss for outliers, and is mitigated in the Smooth L1 loss.



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

2 / 33

Before we go there, the exercise that we left behind from the last lecture. Why is smooth L1 loss less sensitive to outliers than L2 loss? When the deviation of the predicted output from the ground truth is high, which happens when you have outliers, the squared loss or the L2 loss exaggerates the deviation which also results in the gradient exaggerating it which could cause an exploding gradient problem. This issue gets mitigated within L1 loss because L1 loss is not differentiable at 0 we use a smooth version of it known as a smooth L1 loss which mitigates the problems caused due to outliers.

(Refer Slide Time: 1:10)



## Recall: Contemporary Object Detection Methods

### Region Proposal-based:

- Two-stage detection framework
- In the first stage, potential object regions are proposed (through methods such as Selective Search or Region Proposal Network, which we will see soon)
- In the second stage, a classifier processes the candidate regions
- More robust in performance but slower

### Dense Sampling-based:

- One-stage detection framework
- Integrates region proposals and detection by acting on a dense sampling of possible locations
- Simple and fast but performance not as good as Region Proposal-based methods



Vineeth N B. (IIT-H)

§7.2 CNNs for Detection - II

3 / 33

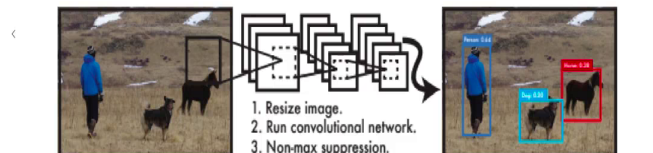
So, we said that contemporary methods can broadly be divided into Region Proposal methods and Dense Sampling Methods. So, we will now go into Dense Sampling Methods which are single stage detection frameworks.

(Refer Slide Time: 1:27)



## You Only Look Once (YOLO): v1<sup>1</sup>

- Single-stage detector based on OverFeat
- Speed with good performance the main aim



<sup>1</sup>Redmon et al, You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

Vineeth N B. (IIT-H)

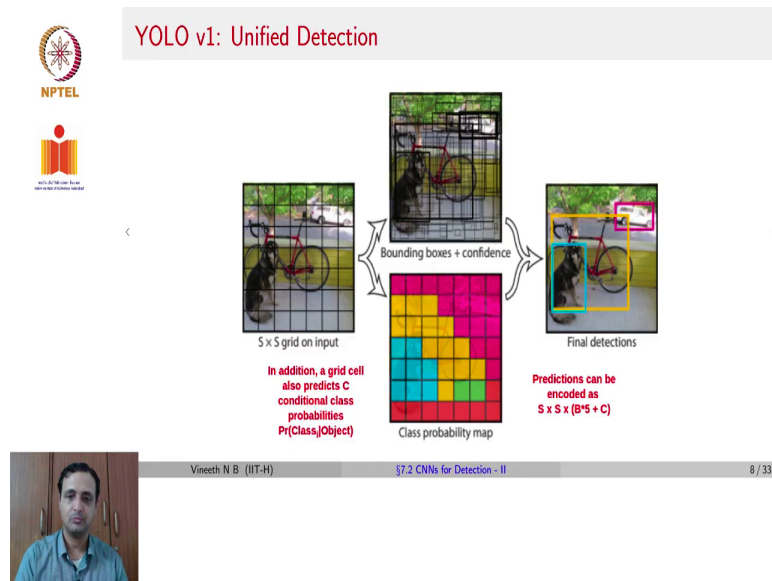
§7.2 CNNs for Detection - II

4 / 33

The most popular that is still used today is known as YOLO or You Only Look Once. The first version of YOLO was developed in CVPR of 2016. It is a single stage detector, you could say that all these single stage detectors are loosely based on OverFeat, they all use only

convolutional layers, no fully connected layers and speed along with good performance is the main aim. So, the high level pipeline is you resize an image, you run a convolutional network, you get a set of outputs then you do non-max suppression to identify your final bounding boxes. Let us see it in more detail.

(Refer Slide Time: 2:16)



So, here is an overall flow, given an input image you first divide the image into an  $S$  by  $S$  grid and each grid cell is responsible for the object with its coinciding center. So for example, if you took this particular grid cell here that would be responsible for this object dog here and has to predict those bounding box offsets. So, each grid cell predicts  $B$  bounding boxes,  $B$  is another hyper parameter and confidence scores for each of these boxes.

So, you can see here that each grid cell can predict multiple bounding boxes and a confidence for each of those bounding boxes and each grid cell predicts just one probability per class. So, total of  $C$  classes would mean  $C$  total probabilities. You could consider that this is computing a quantity which is given by probability of class  $I$  given an object. So that is a conditional probability. The final predictions can be encoded as you have an  $S$  cross  $S$  grid then in each grid location you predict  $B$  bounding boxes. For each of those  $B$  bounding boxes you have 5 values. What are those 5 values?

4 coordinates that could be the center of the object and the width and the height and a confidence for each bounding box so that amounts to 5 and then you have  $C$  class probabilities per grid cell and that is how you get  $S \times S \times (B \times 5 + C)$  that is the total number of outputs that you would have in your output layer before applying non-max suppression.

(Refer Slide Time: 4:14)



## YOLO v1: Bounding Boxes and Confidence Scores

### Bounding Boxes:

- Each bounding box gives 4 coordinates  $x, y, w, h$
- $(x, y)$  = Coordinates representing center of the object **relative to the grid cell**
- $(w, h)$  = Width and height of the object **relative to the whole image**

### Confidence Score:

- Reflects how confident the model is that the box contains an object and also how accurate it thinks the box is
- Formally,

$$\text{Confidence} = Pr(\text{Object}) * IOU_{pred}^{truth}$$



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

9 / 33

So, each bounding box gives 4 coordinates  $x, y, w, h$ ;  $x, y$  are the coordinates representing the center of the object related to the grid cell and  $w, h$  are the width and height of the object relative to the whole image and the confidence score given for each bounding box reflects how confident the model is that that bounding box contains an object and also how accurate the boxes. So, you could view the confidence to be the probability of an object, any object at this point you do not know which class it belongs to, so it is just the  $P(\text{object}) \times IoU(\text{ground truth}, \text{pred})$ . So, confidence takes into account both of these quantities.

(Refer Slide Time: 5:10)



### YOLO v1: Conditional Class Probabilities

- Regardless of number of boxes  $B$ , we only predict one set of class probabilities per grid cell
- At test time, class-specific confidence scores for each box are given by:

$$Pr(Class_i|Object) * Pr(Object) * IOU_{pred}^{truth} = Pr(Class_i) * IOU_{pred}^{truth}$$



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

10 / 33

And you have the conditional class probabilities for each class in each grid cell regardless of the number of boxes  $B$  predicted by each grid cell. YOLO predicts only  $C$  class probabilities for one grid cell. So, you could assume that those  $C$  class probabilities are these conditionals here given by probability of class  $i$  given an object. So, if you multiply these confidence scores, these class conditionals with your confidences, you would get probability of class  $i$  given an object which is the conditional class probability and the confidence which is given by product of probability of object and the IOU between ground truth and predicted values which amounts to the  $P(class\ i) \times IoU(Ground\ Truth\ \&\ pred)$

So, for each cell we are saying what is the probability that this class occurs in this grid cell and for the predicted box, how much IOU does it have with the ground truth.

(Refer Slide Time: 6:19)



## YOLO v1: Loss Function

Loss function used to train YOLO v1 given by:

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

where:

- $\mathbb{1}_i^{\text{obj}}$  denotes if object appears in cell  $i$
- $\mathbb{1}_{ij}^{\text{obj}}$  denotes that  $j^{\text{th}}$  bounding box predictor in cell  $i$  is 'responsible' for that prediction



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

11 / 33

What loss function do you use to train YOLO? The loss function effectively combines the several components that you need to ensure that each of these quantities that is being predicted is close to the ground truth. So you see here there are multiple terms, so the first term, so let me first explain these indicators here. One  $\mathbb{1}_{ij}^{\text{obj}}$  indicates if the  $j^{\text{th}}$  bounding box predictor in cell  $i$  is responsible for that prediction. Remember, every cell predicts  $b$  bounding boxes, so  $\mathbb{1}_{ij}$  here corresponds to the  $j^{\text{th}}$  bounding box predicted by  $i^{\text{th}}$  cell and one  $\mathbb{1}_i$  denotes if object appears in cell  $i$ .

Now let us look at what each of these quantities are doing. So, if you are looking at a particular bounding box predicted by a particular cell, you are first trying to ensure that the  $x_i, y_i$  predicted by the network is close to the expected  $\hat{x}_i, \hat{y}_i$ , then you are ensuring that the width and the height predicted by the network is close to the original width and the height, these are taken a square root because you have a width and a height in two dimensions so you take a square root here and the  $C$  and  $\hat{C}_i$  are the confidence values.

So, you also try to ensure that if the confidence is low with respect to what the confidence should have been which would have been 1 if an object was there, you also try to ensure that that is minimized and all these summations as you can see is done for across all of your grid cells and across all of the bounding boxes predicted by each of these grid cells and you do the same even

when an object is not present you would want the confidence to match the expected confidence which in this case would be 0. So, that is what these two terms in sense are complementary for positive and negative classes.

And finally, the last term here denotes the conditional class Probabilities matching the expected conditional class probabilities. So, these are the different terms in the loss function used to train the network.

(Refer Slide Time: 8:44)



### YOLO v1: Limitations

- Detects only a small number of objects
- Cannot detect objects that are small or close due to strong spatial constraints
- High localization error
- Relatively low recall



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

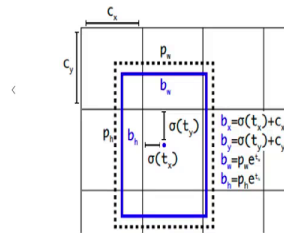
12 / 33

What about, what are the limitations of such an approach? YOLO v1 had a few limitations. Firstly it detects only a small number of objects. It misses objects that are small or close to each other because of the methodology itself because of the spatial constraints of the methodology and how many objects you can present that could be overlapping. It ended up having a high localization error and a relatively low recall.

(Refer Slide Time: 9:21)



## YOLO v2: Anchor Boxes



- Predicts 5 coordinates per anchor box:  
 $t_x, t_y, t_h, t_w, t_o$
- If cell is offset from top left corner of image by  $(c_x, c_y)$  and bounding box has width and height  $p_w, p_h$ , then predictions correspond to:

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

13 / 33

To overcome these issues YOLO v2 which was proposed as an extension of YOLO v1 introduced the idea of anchor boxes into the YOLO framework, these anchor boxes are similar to what you saw with faster R- CNN. So, let us see how these anchor boxes work. So there are 5 coordinates predicted per anchor box  $t_x, t_y, t_h, t_w$  and  $t_o$ . Let us see each of them. So, if the cell is offset from the top left corner by  $c_x, c_y$ , top left corner of the image and the bounding box has width and height  $p_w$  and  $p_h$ , then the predictions corresponding to the anchor box are given by  $b_x = \sigma(t_x) + c_x$ .

So, you can see here that  $\sigma(t_x)$  is right here similarly  $b_y = \sigma(t_y) + c_y$  so that is the other coordinate of the same location added to  $c_y$  and  $b_w$  and  $b_h$  are written in terms of scaling multiples over  $p_w$  and  $p_h$ . So,  $b_w = p_w \times e^{t_w}$  and  $b_h = p_h \times e^{t_h}$ . So,  $t_w$  is a predicted quantity so  $b_w$  is given by what scaling factor should you change the width or the height? It is similar to predicting an offset but the offset is a multiplicative factor for width and height and finally, the  $p(\text{object}) \times IoU$  which is the quantity that we just saw for YOLO v1 is  $\sigma(t_o)$  that is the conditional class probability.

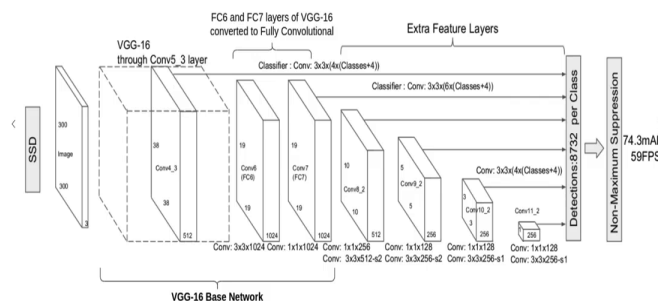


So, those are the different quantities that are predicted by the network and which relate to the actual bounding box that that grid location is trying to point to. Remember that each anchor box predicts only these 5 values, this is, the  $b_x$ ,  $b_y$ ,  $b_w$ ,  $b_h$  are the actual bounding box corresponding to these values given by the anchor box.

(Refer Slide Time: 11:42)



## Single Shot MultiBox Detector (SSD)<sup>2</sup>



<sup>2</sup>Liu et al, SSD: Single Shot MultiBox Detector, ECCV 2016

Vineeth N.B. (IIT-H)

§7.2 CNNs for Detection - II

14 / 33

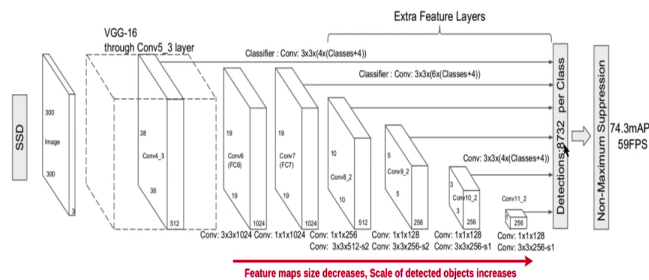
Moving on, there was another single stage detection method known as the single shot multibox detector known as SSD. SSD again uses an OverFeat like methodology. You can see here that given an input image the initial part of the network is a VGG like network then on there are convolution layers that keep reducing the size and you can see there are some skip connections that take you from a convolutional layer directly to the classifier. Let us see them in a bit more detail.

(Refer Slide Time: 12:18)



## SSD: Model

### Multi-scale Feature maps for Detection



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

15 / 33

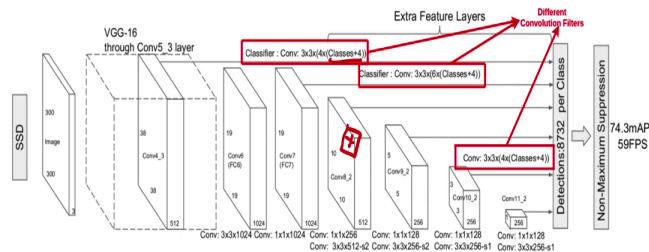
So you have multi-scale feature maps for detection because you are sending these convolutional layers directly to the output, these convolutional feature maps directly to the output, these ones directly to the output. So, the output layer receives the feature maps from convolutional layers of different scales that is why multi-scale feature maps.

(Refer Slide Time: 12:42)



## SSD: Model

### Exclusive convolutional predictors for each feature map



A feature layer of size  $m \times n$  with  $c$  channels gives  $m \times n$  locations (grid cells); bounding box offsets output values relative to grid cell location (like in Faster R-CNN)



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II


16 / 33

Then we also see that for each of these convolutional layers there are a different set of convolutional filters that connect it to the output layer. So, this initial layer here goes through a  $3 \times 3 \times 4 \times (classes + 4)$ , that is the number of outputs that it has, that is the number of,

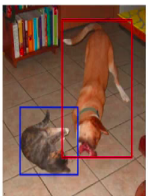
that is the size of the convolutional filters that you have rather which gives the corresponding output in the output layer and so on and so forth for each of these intermediate convolutional layers.

So, if you took anyone of these feature maps, let us take a feature map layer, a layer with a feature map of size  $m \times n$  with  $c$  channels, then that would give  $m \times n$  locations. So, each of those pixel locations in one of those feature maps could be a center of an object for instance, So, each of them is like a grid cell if you could compare this to YOLO and the bounding box offset values are relative to that grid cell location. So, you have, remember you see here that each of these convolutional maps predict  $classes + 4$ , that is the number of values that they would predict. So a class probability for each class plus 4 values for the bounding box offset corresponding to each pixel location in these convolutional feature maps.

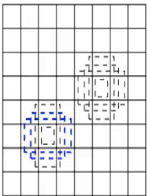
(Refer Slide Time: 14:12)



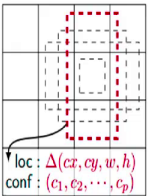
### SSD: Anchor Boxes and Aspect Ratios



(a) Image with GT boxes




(b)  $8 \times 8$  feature map



(c)  $4 \times 4$  feature map

loc :  $\Delta(cx, cy, w, h)$   
 conf :  $(c_1, c_2, \dots, c_p)$

- For each of  $k$  default boxes with different aspect ratios, SSD predicts  $c$  class-specific scores and 4 anchor box offsets
- For an  $m \times n$  feature map, there are  $(c + 4)kmn$  outputs



Vineeth N B (IIT-H)
§7.2 CNNs for Detection - II
17 / 33

So, as feature maps get smaller and smaller when you go through later parts of the network so you can see here if this was your ground truth boxes in your original image and you can see here an  $8 \times 8$  feature map followed by a  $4 \times 4$  feature map. So in an  $8 \times 8$  feature map, if you looked at the anchor boxes for each of these grid cell locations, each of these anchor boxes would predict a set of values the way we talked about it the previous slide. So, if you had  $k$  anchor boxes with different aspect ratios as you can see here, SSD would predict  $c$  class specific scores plus 4 anchor box of offsets. That is what we saw on the earlier slide as  $c + 4$  is the

number of channels that you had for each of your convolutional layers when you connected them to the output layer.

So, for an  $m \times n$  feature map, you would totally have  $(c + 4)k \text{ anchor boxes} \times mn \text{ outputs}$  because you are assuming now that each pixel can be the center of an object and you have  $k$  anchor boxes around each of those pixels and each anchor box predicts  $c$  class probabilities and 4 bounding box offsets. So, that is why for each  $m \times n$  feature map, you would have these many outputs in the SSD framework.

(Refer Slide Time: 15:45)



## SSD: Loss Function

Weighted sum of localization loss (loc) and confidence loss (conf):

$$L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

where:

- $x_{ij}^p$  is  $\{1, 0\}$  if  $i^{\text{th}}$  default box matches  $j^{\text{th}}$  ground truth box of category  $p$
- $c$  = class probabilities
- $l, g$  = predicted and ground truth box parameters



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

18 / 33

What is the loss function you use here? Very similar to the loss functions we have seen so far, SSD implements a localization loss function and a confidence loss function. So, the confidence loss function compares the confidences of  $x$  and  $c$ ,  $c$  are the predicted class probabilities and  $x$  are the ground truth and the way the ground truth is written is you have  $x_{ij}^p$  is given by 1 or 0 depending on whether there is an object or not an object, if the  $i_{th}$  default box or the anchor box, default box and anchor box are the same synonymous here, the  $i$ th default box matches the  $j$ th ground truth box of category  $p$ . That is the notation for these  $x$  i's and  $l$  and  $g$  are the predicted and ground truth box parameters. Let us see each of these loss functions in a bit more detail.

(Refer Slide Time: 16:45)



## SSD: Loss Function

- Localization loss  $L_{loc}$  given by:

$$L_{loc}(x, l, g) = \sum_{i \in Pos} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k \text{smooth}_{L1}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx}) / d_i^w \quad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy}) / d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \quad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

Regress offsets for center  $(cx, cy)$  of anchor box  $(d)$  and its width  $(w)$  and height  $(h)$

- Confidence loss  $L_{conf}$  is softmax loss of class confidences (probabilities):

$$L_{conf}(x, c) = - \sum_{i \in Pos} x_{ij}^p \log(c_i^p) - \sum_{i \in Neg} \log(c_i^0) \quad \text{where} \quad c_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

19 / 33

The localization loss for SSD is given by, once again a smooth L1 loss. It also has a factor  $x_{ij}^k$  in the beginning to say whether that is a class or not because you only want to evaluate this quantity when a class is involved that is when this would turn out to be 1 and this 1s are the predicted offsets and  $\hat{g}_j$  are the ground truth offsets and these ground truth offsets are given by  $\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx}) / d_i^w$  which is the center of the current anchor box. You are just ensuring that your  $g_j$  represents the correct offset with respect to the anchor box under question and you are normalizing it with respect to the width and the height for the x and y dimensions.

And this quantity here is the exponential factor that we were using to scale the width and the height. Instead of an additive factor we said that YOLO uses a multiplicative factor for the width and the height and because that had an exponential term there, you are using a log here to reverse the operation, so that is going to be your ground truth scaling factor that you would want your network to predict and when you take an exponent of that  $l_j$ , you would get rid of the log here and you would get your correct expected width and height. The confidence loss which is the other loss with SSD is a soft max loss of class confidence probabilities.

So, it is given by the first term for all your positive bounding boxes which is your standard cross entropy loss and the second term for your negative bounding boxes where there is a class label

corresponding to a background class which you would want to maximize. So,  $c_i^0$  here corresponds to a class label known as background which is considered as one of the classes you would like to predict and  $\hat{c}_i$  is your standard Softmax activation function.

(Refer Slide Time: 19:07)



## SSD: Practical Implementation

### Hard Negative Mining:

- Similar to Faster R-CNN, most anchor boxes are negative
- To counter it, select negative examples that have highest confidence loss such that ratio between negatives and positives is  $\sim 3 : 1$

### Data Augmentation: Training samples are obtained as below:

- Use original image
- Randomly sample a patch, such that minimum IoU with objects is in  $\{0.1, 0.3, 0.5, 0.7, 0.9\}$



Vineeth N B (IIT-H)

[7.2 CNNs for Detection - II]

20 / 33

In its practical implementation, both YOLO and SSD have this problem is that, most anchor boxes are likely to be negative when you compare with something like a faster R-CNN. So, to counter this, you select negative examples that have the highest confidence loss such that you maintain a ratio of negative to positive to be about 3:1 because otherwise remember that even if you had 100 objects in a single image, the number of anchor boxes and the grid cells that you have if you take SSD, it is in fact going to be per pixel, you will have  $k$  different anchor boxes and that can be a huge number in terms of the number of boxes that are negative and have no object in them.

And then learning can get affected and which is the reason you do this hard negative mining which when you train, you only select some of those boxes that have a very high confidence loss and only use them in the loss function that we talked about. Remember the loss function considered those negative boxes also. SSD also used a data augmentation strategy where given an original image the original image was also used. It also randomly sampled patches from images trying to ensure that the minimum IOU with the actual object is in a predefined set of

ranges to ensure that the network gets exposed to different kinds of patches from images and different kinds of objects and their overlaps.

(Refer Slide Time: 20:52)



### SSD: Performance

mean Avg Precision @ IoU

IoU=0.5

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

21 / 33

With this approach, SSD could outperform YOLO and faster R-CNN. All detection methods are measured using an evaluation metric known as MAP or Mean Average Precision. Precision refers to the standard precision metric used in machine learning methods, average precision talks about the average precision obtained across all of your classes and the mean is across all of your bounding boxes. So, one typically measures mAP at a particular IOU. So, how do you confirm whether you have predicted a bounding box or not?

So, you pre-specify a particular IOU such as 0.5 and say that as long as my predicted box has at least an IOU of 0.5 with my ground truth box, I am going to consider my prediction correct. So, that is how correctness is defined to get your precision and then you take the average across the classes and the boxes that you have predicted. So, you can see here that SSD matched faster R-CNN in its mAP but at a significantly higher FPS, at a significantly higher frames per second rate which was the main objective to make the single stage methods much faster in practice.

You can see that (this was also) the number of output boxes are significantly higher obviously with SSD because you do it for every pixel and they also showed that this works reasonably well with different input resolutions.

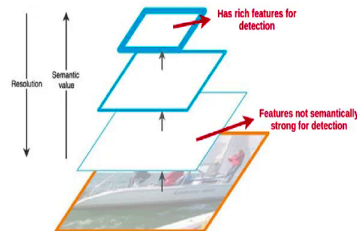


(Refer Slide Time: 22:37)



### Feature Pyramid Network (FPN)<sup>3</sup>

- Feature maps from initial layers (which are high resolution) cannot be used for detection
- FPN provides a top-down pathway to construct higher resolution layers from a semantically rich layer



<sup>3</sup>Lin et al, Feature Pyramid Networks for Object Detection, CVPR 2017

Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

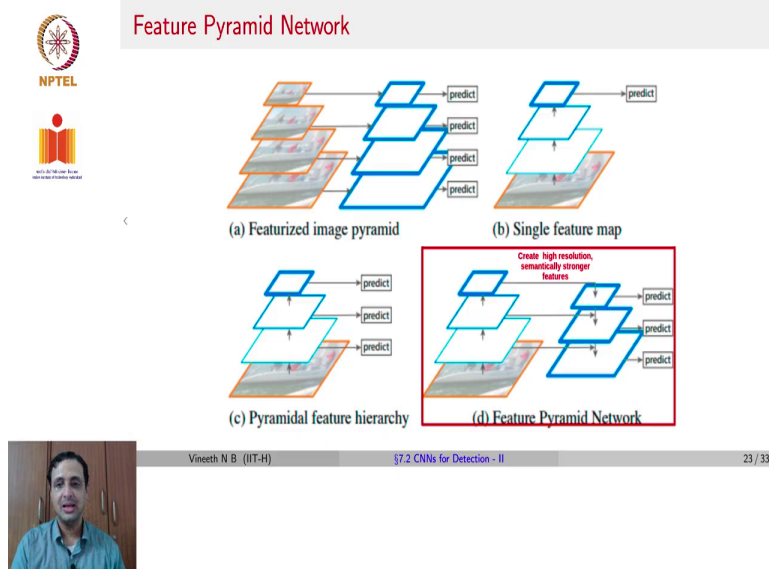
22 / 33



A third single stage detector approach is known as the Feature Pyramid Network or FPN. FPN uses the idea that feature layers from feature maps from initial layers may not really be suitable for detection, they are high resolution. When you go through a convolutional network, the initial layers are high resolution and then as you go deeper, the resolution gets lower and lower and lower but the initial layers although they are high resolution, we have seen from our visualizations of CNNs that they may not really be capturing semantics of objects in those initial feature maps but they are the higher resolution ones.

So, we are caught with this dilemma where the lower resolution feature maps have more richer features for detection whereas the higher resolution is in this initial feature maps. How do we bridge this gap is what feature pyramid network tries to do.

(Refer Slide Time: 23:41)



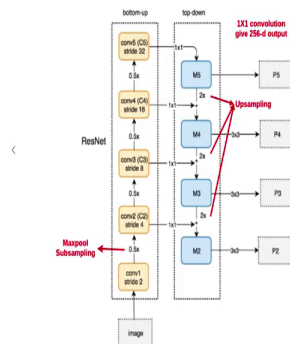
So, here, here is a visualization of how a Feature Pyramid Network attempts to bridge this gap. So, if you were to use an image pyramid to be able to do detection, so you, one thing you could do is you could subsample your images and for each resolution of the image, construct a feature map and predict for each of those feature maps or you could take your input image, construct feature maps at lower and lower resolutions and finally predict at the least resolution or given an input image construct feature maps at different resolutions that as you build many convolutional layers and predict at each of these resolutions.

And what feature map, feature pyramid network suggests is you do construct feature maps at different resolutions as you go through a convolutional network but now you upsample and get back feature maps at higher resolutions and now make predictions. So, this way you try to get your semantics at your, at your least resolution but upsample back to transfer the semantics to a higher resolution.

(Refer Slide Time: 24:59)



## Feature Pyramid Network: Methodology



- All convolutional feature maps are treated with a  $1 \times 1$  convolution with 256 channels
- M5 is upsampled by a factor of 2 (Maxpool at  $1/2 \times 1/2$ )
- M5 and C4 signals are element-wise added to give M4; similarly followed in the downward direction
- $3 \times 3$  convolution is used to reduce aliasing effect of Ms
- Finally, Ps are individually fed into exclusive object detectors

Credit: Jonathan Hui, Medium.com



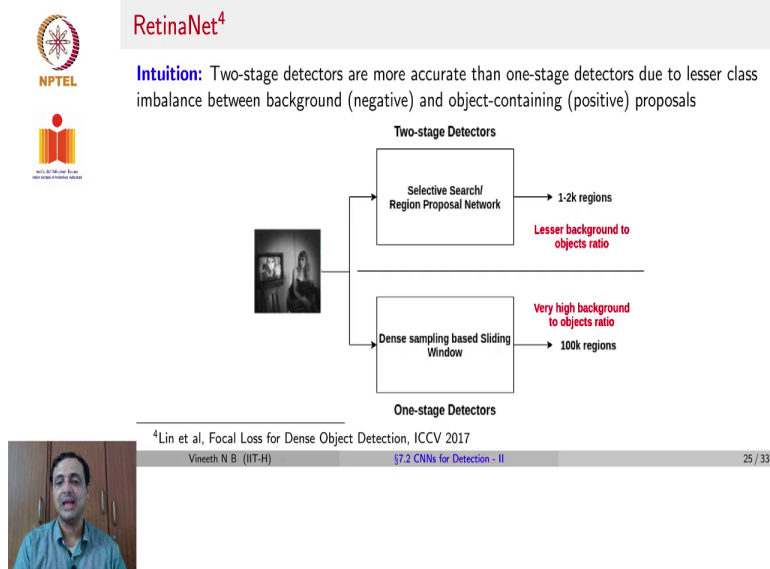
Vineeth N B (IIT-H)

Let us see how this is done in the architecture. So this is the overall architecture. So you have an image, you have a conv layer stride 2, then  $0.5 \times$  denotes a sub sampling max pool layer,  $2 \times 2$  max pool, then you have a conv 2 with a stride 4, a max pool, a conv 3 with a stride 8, a max pool, a conv 4 with a stride 16, a max pool and a conv 5 with a stride 32. So, you can consider, that is like a ResNet that is used. In the FPN, all convolutional feature maps C1 to C5 are treated with a  $1 \times 1$  convolution which you see on these arrows here with 256 channels and M5 is up sampled by a factor of 2 to get the next M4.

But before you get M4, you get the signals from C4 after applying your  $1 \times 1$  convolution and combine these outputs of C4 with  $1 \times 1$  convolution and M5 to get M4 and you similarly continue to do this to get M3 and M2. Once you do this a  $3 \times 3$  convolution is applied on M4, M3 and M2 and this is done to reduce the aliasing effect of Ms. Remember, that we are up sampling when we go from M5 to M4, M4 to M3 and M3 to M2 and up sampling, recall, we said could result in aliasing.

So, to smoothen out those aliasing factors we use a  $3 \times 3$  convolution which takes us from M4 to P4, P3, P2 so on and so forth. So finally, you are left with all of these Ps here which are provided to individual object detectors to get your final predictions.

(Refer Slide Time: 27:03)



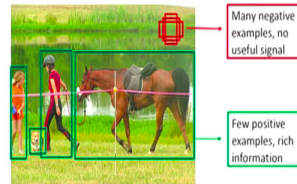
A more recent approach to object detection focused on the loss function that is used to train these object detectors. So, this was known as RetinaNet and the loss function that was proposed is known as the focal loss which was proposed in ICCV of 2017 and this relies on the intuition that two stage detectors were known to be more accurate than one stage detectors. One stage detectors were obviously faster and it surmised that two stage detectors are more accurate because this lesser class imbalance between background or negative classes and object containing or positive proposals.

Remember, when you do selective search we restricted ourselves to only 2000 region proposals whereas in one stage detectors you could be dealing with 100,000 regions because you could be getting multiple anchor boxes around each grid cell, in SSD it is on each pixel of the feature map but even with YOLO you may be predicting  $b$  bounding boxes for each of those  $s \times s$  grid cells which could be a very high quantity. So, how do we address this imbalance between negative or background classes and the actual positive classes?

(Refer Slide Time: 28:33)



### Class Imbalance in Object Detection: The Problems



- Training is inefficient as easy negatives contribute no useful signal
- Loss due to easy negatives overwhelms loss due to positives and thereby, training process can lead to degenerate models; hard negative training alleviates it to some extent

Credit: Sik-Ho Tsang, TowardsDataScience.com



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

26 / 33

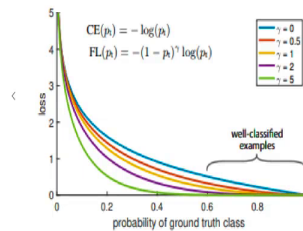
Whenever you have such an issue of where there are many negative examples which really do not help the the model and there are only a few positive examples, certain things that could happen are training could become inefficient because the easy negatives are not really giving any useful signal there could be so many kinds of easy negatives that it is not really going to help the model learn how to distinguish the positives from those negatives because that class of negatives or the background will be extremely vast in detection, so the background could be the sky, could be grass, could be buildings, it could be any of those and all of them are still a background .

Secondly, the loss could get overwhelmed because of the negatives instead of the positives and this could degenerate the training process lead to degenerate models. To some extent the hard negative mining that we spoke about in SSD where we try to ensure that the final loss only uses where there is a significant confidence loss and ensures that the ratio of 3:1 between negative and positive. That does alleviate these issues but the issue still remains.

(Refer Slide Time: 29:53)



## CE Loss is Bad<sup>5</sup>



<sup>5</sup>Lin et al, Focal Loss for Dense Object Detection, ICCV 2017

Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

27 / 33



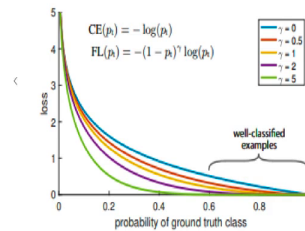
So, what this particular paper proposes is that cross entropy loss for using for the classification branch of detection could be inherently bad. Let us try to see why. Remember that the cross entropy loss is defined by  $-\log(p)$ . In the multi-class setting, it just turns out to be  $-\log(p_t)$ , log loss as we mentioned and over their empirical studies they observed this graph here where if you notice even when you have, if you observe the gamma is equal to 0 remember when gamma is equal to 0, this loss introduced by them known as the focal loss would turn out to be this coefficient will turn out to be 1 when gamma is 0.

And the loss would just become your standard log loss or cross entropy loss so when gamma is equal to 0, blue is your standard cross entropy loss and you can see here that even when the network predicts a high probability for the ground truth class the loss value is fairly non-trivial, you get a fairly high loss even when the model is predicting a high probability for the correct class and this can defeat the purpose of learning.

(Refer Slide Time: 31:17)



## RetinaNet: Balanced Cross Entropy and Focal Loss



### Balanced Cross Entropy:

- Introduces a weighting factor  $\alpha \in [0, 1]$  which can be inverse class frequency or a hyperparameter

- $\alpha$ -balanced CE loss is given by:

$$CE(p_t) = -\alpha \log(p_t)$$

### Focal Loss:

- Adds a modulating factor  $(1 - p_t)^\gamma$  to CE loss, with tunable focusing parameter  $\gamma \geq 0$ .

- Focal loss is given by:

$$FL(p_t) = -(1 - p_t)^\gamma \log(p_t)$$



Vineeth N B (IIT-H)

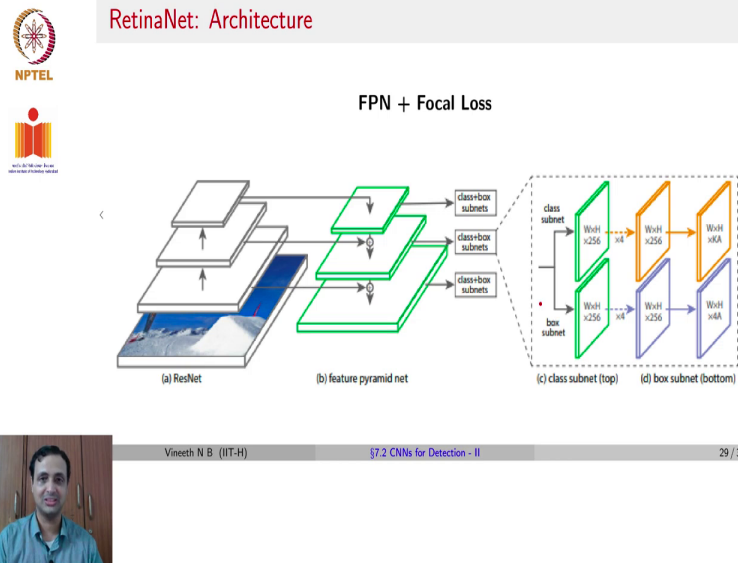
§7.2 CNNs for Detection - II

28 / 33

So, what RetinaNet with focal loss proposes is, one, you could do a balanced cross entropy where the  $-\log(p_t)$ , the log loss is weighted by some quantity  $\alpha$  and  $\alpha$  can be given as some inverse class frequency. That is one way to do this, but this RetinaNet Method also proposes a focal loss which considers the predicted probability itself to fine tune the loss. So you weight your log loss with  $(1 - p_t)^\gamma$  where  $\gamma$  is a tunable focusing parameter so  $\gamma$  is a hyper parameter that you have to provide while training the network and the final focal loss gives this quantity.

If you observe here, let us assume  $\gamma$  is a certain values such as 5, so when  $p_t$  is high, this quantity is going to become low and you are bringing down the overall loss because when  $p_t$  is high you want the loss to be low and when  $p_t$  is low let us say it is 0.1 because  $\gamma$  is 5 you would still have this quantity to be a reasonable quantity and the log loss would be maintained at a high level when your probability, predicted probability  $p_t$  for the ground truth class is low. That is the main idea of the Focal loss.

(Refer Slide Time: 32:48)



The RetinaNet architecture otherwise uses of FPN the feature pyramid network that we spoke about along with the focal loss where you can see here the first part of it is the feature pyramid network itself then for each of these scales you have a classification subnetwork and a bounding box regression subnetwork and the classification sub network uses the focal loss to learn.

(Refer Slide Time: 33:17)

**Detectron**

- Framework developed by Facebook AI Research (FAIR) to implement state-of-the-art object detection algorithms
- Highly flexible providing support across various algorithms and backbone networks
- See [Detectron](#) and [Detectron2](#) for details



Vineeth N B (IIT-H) §7.2 CNNs for Detection - II 30 / 33

There are all implementations of all of these contemporary detection methods both Dense Sampling Methods and the Region Based Proposal methods in a popular library known as



Detectron. Detectron was provided by Facebook AI Research especially to promote usage of object detection algorithms. So, if you are interested in implementing any of these object detection algorithms for any of your projects, you can look at Detectron or Detectron2 for further details.


(Refer Slide Time: 33:50)



### Homework

#### Readings

- Object Detection for Dummies, Part 4
- YOLO Family: All you want to know
- Understanding SSD
- Understanding FPN
- Understanding RetinaNet



Vineeth N B (IIT-H) §7.2 CNNs for Detection - II 31 / 33

So, your readings are a continuation of Object Detection for Dummies, this time Part 4 for the dense sampling methods. Here is a tutorial of the entire YOLO family of methods and tutorials on understanding SSD, FPN and RetinaNet.

(Refer Slide Time: 34:11)



## Homework

### Exercises

- YOLO9000 and YOLOv3 were follow-ups of YOLOv2. What was different in these extensions? Find out! (Hint: see the [YOLO Family: All you want to know](#) link)
- Given two bounding boxes in an image: an upper-left box which is  $2 \times 2$ , and a lower-right box which is  $2 \times 3$  and an overlapping region of  $1 \times 1$ , what is the IoU between the two boxes?
- Consider using YOLO object detector on a  $19 \times 19$  grid, on a detection problem with 20 classes, and with 5 anchor boxes. During training, for each image, you will need to construct an output volume  $y$  as the target value for the neural network; this corresponds to the last layer of the neural network. ( $y$  may include background). What is the dimension of this output volume?



Vineeth N B (IIT-H)

§7.2 CNNs for Detection - II

32 / 33






A few exercises to leave behind, we only covered YOLOv1 and YOLOv2 in this lecture. YOLO also had a YOLO9000 which talked about scaling YOLO to 9000 categories and a YOLOv3 which was very close to YOLO9000 in its ideas. How were these different from YOLO v2 is going to be a homework for you. Please do read the link that was given in the reading section in the previous slide for understanding YOLO and a couple of more simple problems, given two bounding boxes in an image an upper left box which is  $2 \times 2$  and a lower right box which is  $2 \times 3$  and an overlapping region of  $1 \times 1$ . What is the IOU between the two boxes?

To understand YOLO better, consider using YOLO on a  $19 \times 19$  grid on a detection problem with 20 classes and 5 anchor boxes. During training you have to construct an output volume as the target value for the neural network. What would be the dimension of this output volume? Remember, that in YOLO we said  $S \times S \times (5B + C)$ . Try to use that formula here to find out what should be the output volume for this particular YOLO object detector. Please do these exercises and we will continue the discussion in the next lecture.

(Refer Slide Time: 35:52)



## References

-  Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.
-  Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.
-  Tsung-Yi Lin et al. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.
-  Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.
-  Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.



Vineeth N B (IIT-H)