



Deep Learning for Computer Vision
Professor. Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Lecture No. 45
CNNs for Object Detection-I Part 2

(Refer Slide Time: 0:15)




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) §7.1 CNNs for Detection 25 / 44

Having seen how object detection was performed in the pre deep learning era as well as how basic CNNs can be adapted to object detection, let us now move to how object detection is done today in terms of contemporary methods. Existing methods that obviously are based on top of CNNs can be broadly divided into what are known as the Region Proposal based methods and Dense Sampling based methods. So we will cover region proposal based methods in the rest of the structure and take up dense sampling methods in the next lecture.

In region proposal based methods, the framework is composed of two stages, in the first stage there is a network or a module that proposes a different regions that could contain objects and in the second stage a classifier, in our case a CNN takes those regions and classifies them as belonging to an object or could even belong to the background, this is more robust in performance but is slow because of the two-stage approach.

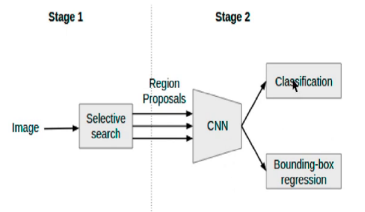
On the other hand, dense sampling approaches are a single-stage approach where the proposal of regions and the classification or detection of those regions into objects is all done in one shot through a dense sampling procedure. This is simpler and faster but sometimes can fall short of the accuracy when compared to region proposal methods.

(Refer Slide Time: 2:05)



R-CNN (Region-based Convolutional Neural Networks)⁶

- Region proposal-based object detection network
- Uses **Selective Search** as region proposal algorithm to identify potential objects
- Each region is then evaluated by a CNN that performs classification and bounding box regression



⁶Girshick et al, Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014



Vineeth N B (IIT-H)

§7.1 CNNs for Detection

26 / 44

Let us now discuss region proposal methods. So, the earliest one was known as the R-CNN or the region based CNN which was proposed in CVPR of 2014 by Ross Girshick,. This approach has a first stage where it uses a method known as selective search which is an image processing method, it does not use a neural network for that but using a selective search approach it proposes several regions that could have objects. Each of these regions is then evaluated by a CNN to perform classification as well as do bounding box regression to get the correct localisation inside those region proposals.

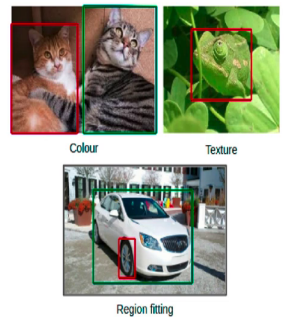
Remember when we talked about OverFeat, we said that we could divide these into two heads, we will continue to use that for all detection methods that we will see.

(Refer Slide Time: 3:02)



R-CNN: Selective Search

- Uses graph-based image segmentation or mean shift method for initial set of region hypotheses
- Hypotheses are hierarchically grouped/combined based on region similarity measures like color, texture, size and region-filling



Vineeth N B (IIT-H)

§7.1 CNNs for Detection

27 / 44

How do you do that first stage selective search? So, the selective search module uses a graph based image segmentation or a mean shift based image segmentation to get an initial set of region hypotheses. So, these hypotheses are then hierarchically grouped based on region similarity measures like colour, texture, size, region-fillings so on and so forth. So, if you had, if you used colour for grouping up regions, you may end up getting an object such as this, if you used texture for grouping up regions, you may get a region such as this or if you used region-filling, you may end up getting a region such as this.

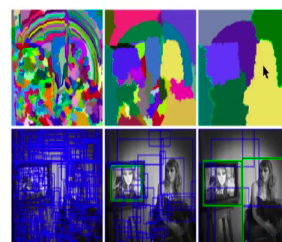
(Refer Slide Time: 3:50)



R-CNN: Selective Search



Initial proposals



Regions coalesce as we travel up the hierarchy



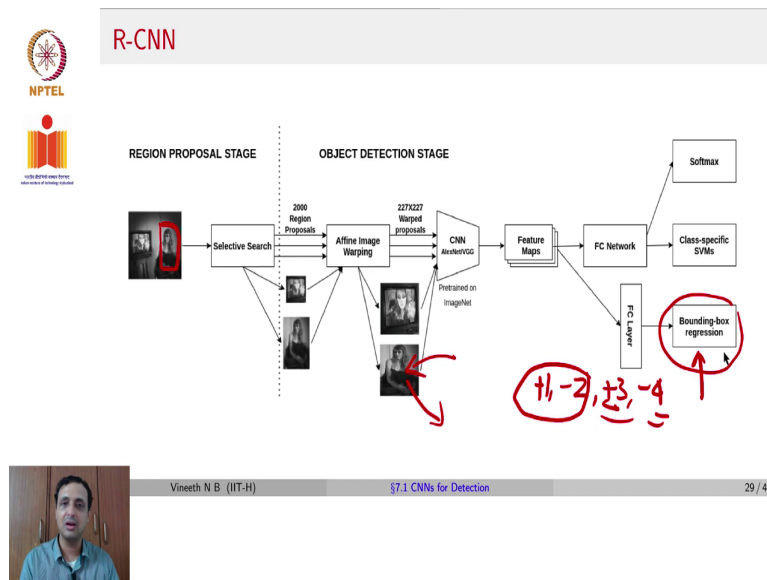
Vineeth N B (IIT-H)

§7.1 CNNs for Detection

28 / 44

In practice, you have an image, you use an existing segmentation method, we covered this in the initial weeks, you will get various different segments that you see here, then you combine these segments using agglomerative clustering, if you recall, using either colour cues, texture cues or region filling cues or any similar cues and this gives us regions of a larger scale.

(Refer Slide Time: 4:24)



Now how do we use these regions? Once you get these region proposals, this particular method selects 2000 such region proposals by combining and qualifying different subregions. You get somewhere around 2,000 region proposals. Each of these region proposals are warped to the same size. Why do we need that? Because we are going to give that as input to the CNN and remember, the CNN can only take a fixed input size as its input dimension, you cannot give a 20 cross 20 image and a 100 cross 100 image.

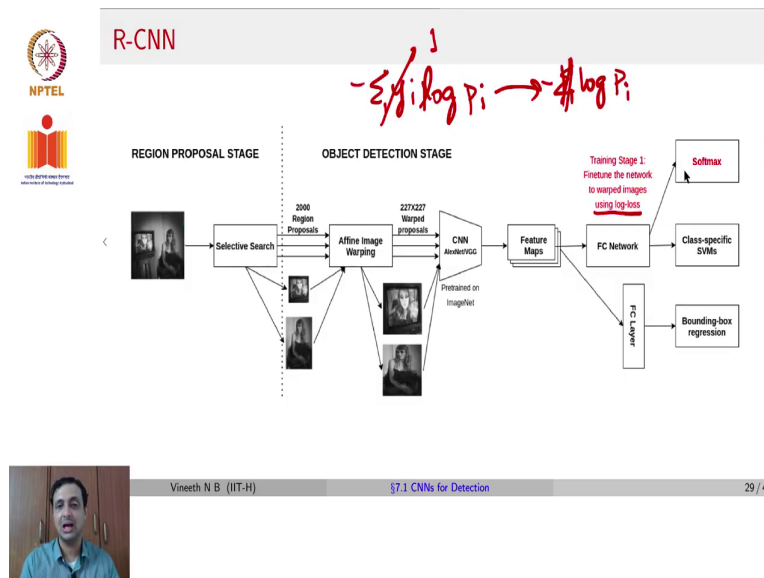
A CNN can only accept one size that is the nature of the architecture that we use. So, whatever be the size of the region proposals, they are all warped to make it the same size so there could be some artefacts of stretching because of this. And these proposals are then given to a CNN so in this particular work, the CNN was like an Alexnet or a VGG, then you get feature maps out of the CNN that goes through a fully connected network.

And then there are 3 components, you have a softmax component that does classification of each of these regions into one of the known classes that you are looking for while performing detection. It also has an SVM which is actually used for classification, we will explain that in a moment and finally it also takes the feature maps through a different fully connected layer

to perform bounding box regression. I will repeat that this bounding box regression step that you see here does not you already have a region given to you, remember that this region is a part of the original image, so what the bounding box regression module does here is to give you the offset from this region in the original image to the correct bounding box.

So, if this region was in a specific location of the original image, the bounding box regression module may give value such as +1, -2 and maybe -3 -4, which means that the top left corner should be moved by one pixel further on the x-axis, moved by two pixels to the left on the y-axis and the height and width must be adjusted by +3 and -4 so on and so forth. So, this bounding box module regression module predicts the offsets from the current location of the box that you have.

(Refer Slide Time: 7:19)

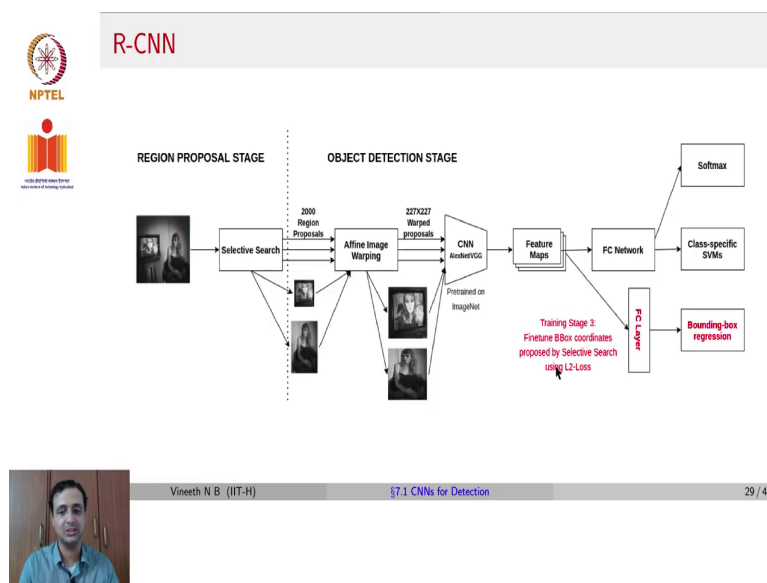


Now let us see the entire training procedure. So in the first step, you consider all of these modules, the CNN, the feature maps, the fully connected network and you fine tune this fully connected network to the warped images using a softmax and a cross entropy loss. We generally refer to cross entropy loss also as log loss because when you have multiple classes your cross entropy loss turns out to be simply the log of the probability of the ground truth class, why so? Because you would have your cross entropy loss to be given by $y_i \log p_i$ summation over i $y_i \log p_i$ this is your cross entropy loss.

$$- \sum y_i \log(p_i) \rightarrow - \log(p_i)$$

Remember that your ground truth is going to be one only for the correct class and is going to be 0 for all other classes, so you will be left with just $-\log(p_i)$ no summation and that is why we sometimes call cross entropy loss as just log loss because when you use it for a multi class classification setting that is what it turns out to be. So, that is the first step to tune the CNNs, you already initialised the CNN with weights of an Alexnet model or a VGG model. Now we will further refine those weights by using a log loss and the softmax in the last layer through the fully connected network and the feature maps.

(Refer Slide Time: 8:50)



Vineeth N B (IIT-H)

§7.1 CNNs for Detection

29 / 44

In the second step, you now take the outputs of the fully connected neural network and use them as representations of each of these regions and then train a class specific SVM depending on the number of classes in your detection problem you could be detecting say up to 20 different kinds of objects or even further, you will then learn 20 different SVMs based on the feature that you got as output of the fully connected network and then the SVM is then used to make the final decision of the class in that region.

And the stage 3 is training the fully connected layer for performing bounding box regression where one would use L2 loss or the mean square error loss, remember they both are synonymous.

(Refer Slide Time: 9:41)



R-CNN: Limitations

- Multi-stage pipeline of training
- Feature extraction of region proposals requires a lot of time and space; e.g. PASCAL VOC07 dataset with 5k images requires 2.5 GPU-days and several hundred GBs
- Object detection takes 47s/image at test time



Vineeth N B (IIT-H)

§7.1 CNNs for Detection

30 / 44

What are the limitations of R-CNN? Firstly, it is a multi-stage pipeline of training as we just saw, you have to first train one head then you have to take representations and train 20 different SVMs or as many different SVMs as classes and then you also have to train a bounding box regression head, can we make this simpler? We will see this in a minute but before that the extraction of region proposal initially could require a lot of time and space. Remember, we have to do segmentation, qualify them and then finally pass each region proposal through the CNN and get your classification bounding box offset so on.

A dataset such as Pascal VOC which has 5000 images approximately required about two and half GPU days and several hundred GBs of storage which can turn out to become impractical. And using R-CNN at test time the object detection took about 47 seconds which is not really real time.

(Refer Slide Time: 10:57)

Fast R-CNN⁷

- Instead of passing each region proposal through CNN, extract their features directly from feature maps
- Use multi-task loss to integrate classification and bounding box regression training stages (no SVMs)

⁷Girshick, Fast R-CNN, ICCV 2015
Vineeth N B (IIT-H) §7.1 CNNs for Detection 31 / 44

This led to the development of the fast R-CNN approach by the same authors in fact, and it was published in ICCV of 2015. In this case there were a couple of different innovations that were introduced. One idea was, while we still do selective search to get the region proposals, we would map these region proposal locations directly to the feature maps after sending the full image through the CNN. If you go back and see how it was done earlier, earlier each region proposal was fed individually to the CNN.

(Refer Slide Time: 11:44)

Fast R-CNN⁷

- Instead of passing each region proposal through CNN, extract their features directly from feature maps
- Use multi-task loss to integrate classification and bounding box regression training stages (no SVMs)

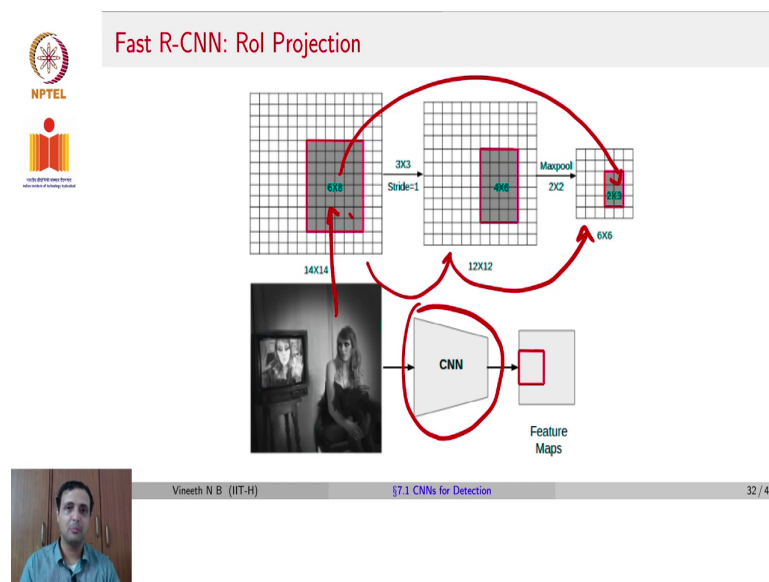
⁷Girshick, Fast R-CNN, ICCV 2015
Vineeth N B (IIT-H) §7.1 CNNs for Detection 31 / 44

But now we are saying that we will feed only the entire image through the CNN. This helps us because otherwise we have to do 2000 forward propagations where 2000 are the number

of region proposals through the CNN. Now we just need to do one forward prop through the CNN which is the entire image. Through selective search, we would get a lot of region proposals. We now map because the region proposals have certain coordinates in the original image. You can map those coordinates to the feature maps that you get as the output of the CNN.

And once you get this feature maps you have two heads, one for classification where log loss is used, one for bounding box regression where L2 loss is used, no more need for SVMs which seems obvious but that is how this method was improved.

(Refer Slide Time: 12:40)



So now, how is the region of interest projected? So, we said that you take these region proposals that are outputs of the selective search approach and map them onto the feature maps, how would you do that? If this was one of your region proposals, let us assume that the 6×8 patch was a region proposal in your original image, then remember that your CNN follows a set of steps where the image gets in most likelihood downsized as you go through each iteration because if you perform convolution the image size reduces, if you perform pooling it further reduces and so on.

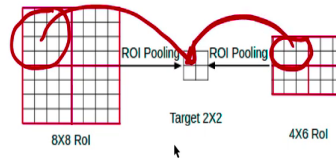
So, you take the same region proposal's dimensions, keep forward propagating it through a similar set of operations and you will notice that this 6×8 patch matches to this 2×3 patch on the feature maps.

(Refer Slide Time: 13:45)



Fast R-CNN: RoI Pooling

- Rols can be of different scales
- We need a scale-invariant way of pooling
- Given an Rol of size $h \times w$, **RoI Pooling** converts this region into $H \times W$ grid of subwindows with each subwindow of size approximately equal to $h/H, w/W$



Vineeth N B (IIT-H)

§7.1 CNNs for Detection

33 / 44

As a next step, after projecting the region proposals on the feature maps, we need to bring all of these region proposals to the same size for the same reason that we said that the next layers can only take input of the same size. To do this, fast R-CNN proposes a module known as RoI pooling. RoI stands for region of interest again, the RoIs can be of different scales, so to get a scale invariant way of pooling, what RoI pooling does is, given any region which is say of size $h \times w$, it converts it into region of $H \times W$ grid of subwindows with each subwindow of size $h \times H$ and $w \times W$.

So for example, if you had an 8×8 region and your target size was 2×2 , you would accordingly divide this 8×8 into a 2×2 grid and do pooling inside each of these 2×2 regions and get your output in the target location. So, your pooling here would happen in this location you would do say a max pooling and only the max value would go here so on and so forth. So, if you had a 4×6 region of interest, then you would still have to do a 2×2 , you still want a 2×2 output.

So, you divide it this way now and you pool across all of these 6 pixels and that value goes here. So, that is how RoI pooling works.

(Refer Slide Time: 15:19)



Fast R-CNN: Multi-Task Loss

- Let u be true class, and \mathbf{v} be ground truth bounding box regression targets
- Let p be predicted class probability, and $\mathbf{t}^u = (t_x^u, t_y^u, t_w^u, t_h^u)$ be bounding box regression offsets for true class
- Loss for each ROI is then given by:

$$L(p, u, \mathbf{t}^u, \mathbf{v}) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(\mathbf{t}^u, \mathbf{v})$$

where:

$$L_{cls}(p, u) = -\log p_u$$

$$\text{and Iverson bracket indicator function } [u \geq 1] = \begin{cases} 1 & \text{if } u \geq 1 \\ 0 & \text{otherwise} \end{cases}$$



Vineeth N B (IIT-H)

§7.1 CNNs for Detection



34 / 44

And once ROI pooling is done, you then send it through further layers, fully connected layers and at the end of which you have two heads with two different loss functions. So you have a multi task loss, one task for classification and one task for regression. Say, if we assume that u is the true class and \mathbf{v} are the ground truth bounding box regression targets, if p is the predicted class probability and \mathbf{t}^u are the bounding box regression offsets for the true class, then the loss for each region of interest would be the classification loss plus your localisation loss. By localisation loss, we mean the bounding box regression loss.

The classification loss as we just said is given by $-\log(p_u)$. The cross entropy loss simplifies into a log loss and the localisation loss is prefixed by an indicator function which states that we are going to compute this only for the true class, for the other classes we do not need to compute this kind of a localisation, that is what the indicator function dictates. What is the localisation loss in fast R-CNN?

(Refer Slide Time: 16:49)


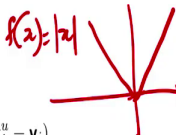
Fast R-CNN: Multi-task Loss




- Localization loss L_{loc} given by:
$$L_{loc}(\mathbf{t}^u, \mathbf{v}) = \sum_{i \in \{x, y, w, h\}} \text{smooth}_{L1}(t_i^u - v_i)$$

where $\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$

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



Vineeth N B (IIT-H) §7.1 CNNs for Detection 35 / 44



Earlier, we said we typically use the L2 loss but in fast R-CNN the method recommends the use of a new loss known as a smooth L1 loss. Note that L1 loss is the sum of the pairwise differences between dimensions and their corresponding absolute values and this is used instead of the L2 loss and a smooth version of the L1 loss, remember that L1 loss, any absolute value function looks somewhat like this, that is the f of x is equal to absolute value function.

$$f(x) = |x|$$

But this function can become non-differentiable at 0, so to avoid that you have a smooth variant of the L1 loss known as smooth L1 loss which is given by $0.5x^2$ if $|x|$ is less than 1, otherwise $|x| - 0.5$ otherwise. So, this would give you slightly smooth, when it comes close to the 0 it becomes quadratic, see you would have a slightly smooth function but otherwise its absolute value or L1 everywhere else. Why smooth L1 loss? It turns out that smooth L1 loss is less sensitive to outliers than L2 loss, why is this so? Is homework for you and we will discuss that in the next lecture.

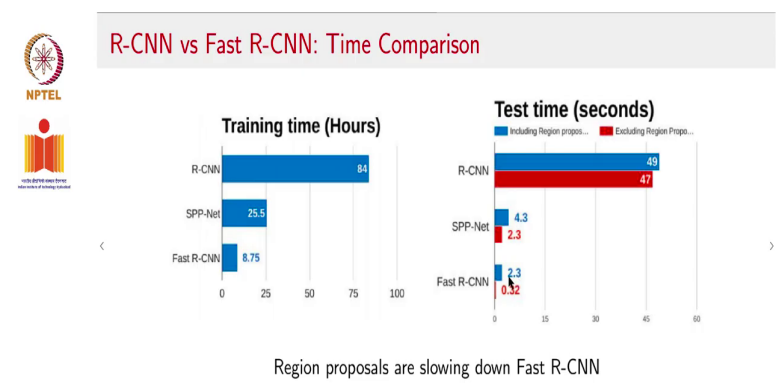
(Refer Slide Time: 18:13)

Fast R-CNN: Summary

Vineeth N B (IIT-H) [§7.1 CNNs for Detection](#) 36 / 44

So, the summary of fast R-CNN is given an image, you still have selective search, you also forward propagate the full image through the CNN, say in Alexnet or VGG whose output gives you a depth of feature maps like the output of any convolutional layer. Now you project all the region proposals that you got through selective search onto this feature maps, this avoided as we said multiple forward propagation through the convolution layer, then you perform RoI pooling to get all those region proposals to the same size, you then send this through a fully connected network which has a softmax head or a classification head and a bounding box regression head.


(Refer Slide Time: 19:06)



With these improvements, fast R-CNN significantly outperformed R-CNN in training time and test time. Fast R-CNN took about 8.75 hours for training while R-CNN took 84 hours for training, that is the 10x reduction. Importantly at test time, while R-CNN took 47 or 49 seconds for doing detection on each image this came down to 2.3 seconds for fast R-CNN and just 0.3 seconds if you have to avoid the region proposal finding step.

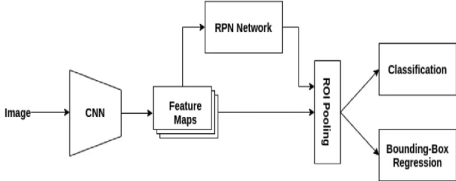
So, that gives us a clue on what needs to be done next, which is we have to find a way of avoiding the process of getting region proposals using selective search because among the 2.3 seconds it looks like 2 seconds is required just for that step.

(Refer Slide Time: 20:06)



Faster R-CNN⁸

- Replaces Selective Search with a **Region Proposal Network (RPN)** to reduce time to compute regions
- Uses **anchor boxes** of different scales and aspect ratios to identify multiple objects present in the same window



⁸Ren et al, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NeurIPS 2015

Vineth N B. (IIT-H) [§7.1 CNNs for Detection](#) 38 / 44


This leads us to faster R-CNN, faster R-CNN replaces selective search with a region proposal network. So, instead of getting region proposals through segmentation methods that are offline, we are going to try to see if this can also be done as part of the neural network itself, this was proposed in NeurIPS of 2015. And let us see how this is done, so given an image you forward propagate the image through a CNN like Alexnet or VGG, you get a set of feature maps.

Now these feature maps are provided to the region proposal network whose job is to tell us which regions are likely to have an object, it does not worry about classifying the object it only worries about objectness of a region and these regions are then provided through an ROI pooling layer, that part of it is very similar to fast R-CNN. And finally you have classification and bounding box regression just like how we had it with fast R-CNN. How does the region

proposal network work? It uses what are known as anchor boxes of different scales and aspect ratios to identify multiple objects in a given window.

(Refer Slide Time: 21:33)

Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, cs231n, Stanford Univ



Vineeth N B (IIT-H)

§7.1 CNNs for Detection

39 / 44

Let us see how that works. So, given a particular window, let us say any particular window in an image. So, it takes a different predefined anchor boxes, so you can define an anchor box which is defined given a particular location you can define an anchor box that is given by this rectangle here, you can define an anchor box that is given by this rectangle here and you can define anchor box that is given by different this rectangle here.

So, these are a set of predefined rectangles which, given the centre, can define several kinds of regions around that centre. What does the region proposal network learn? It has its own loss function to learn that part of the network, recall InceptionNet, recall that in InceptionNet we had certain loss functions at different stages of the network, you could consider this to be similar to that.

So here, the output expected of the region proposal network is, is any of these anchor boxes an object, so you are only looking for object not an object, so there are no further classes, it is a binary classification problem and if you had k anchor boxes which as I just mentioned are predefined, you have to define them when you start. And if you had your depth to be, if you had $512 \times 20 \times 15$ to be your image features, then you are going to have $k \times 20 \times 15$ as the output.

And you would have $4k \times 20 \times 15$ as the bounding box offsets for each of these anchor regions. So, this part of the network does do a bounding box offset recognition or fine tuning but it also checks whether a region is an object or not an object, let us see then what happens?

(Refer Slide Time: 23:38)

The slide shows a diagram of the Region Proposal Network (RPN) architecture. It starts with an 'Image' input that goes into a 'CNN' block. The output is 'Feature Maps'. These feature maps are then processed by the 'RP Network', which contains a 'Fully Convolutional layer'. This layer branches into two outputs: 'Objectness Classification' and 'Bounding-box regression (offsets)'. The slide also includes the NPTEL logo and a small video inset of the presenter.

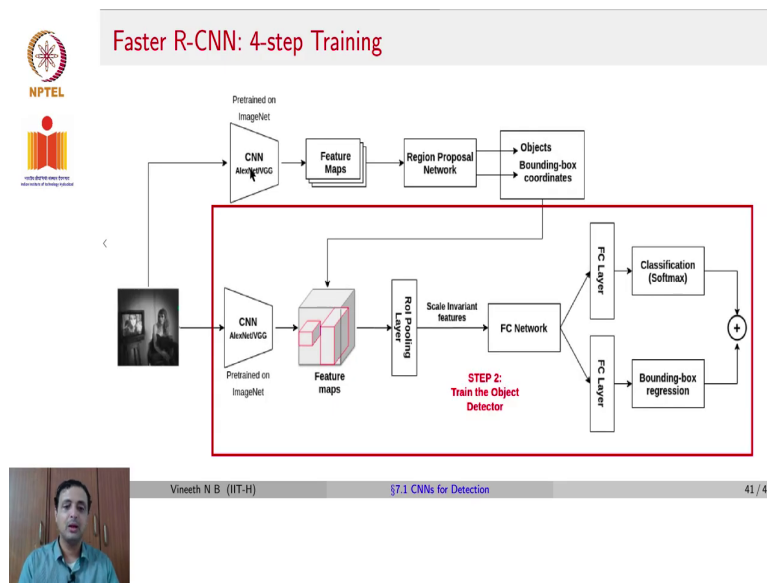
So, you are going to have a fully convolutional layer which does objectness classification and bounding box regression, so in this case we are talking about say 9 anchor boxes which is what was k on the previous slide but this could be different based on what one chooses to do in a given setting.

(Refer Slide Time: 23:56)

The slide illustrates the 4-step training process for Faster R-CNN. It shows a 'Pretrained on ImageNet' CNN (AlexNet/VGG) producing 'Feature Maps'. These are fed into the 'Region Proposal Network' (RPN), which is highlighted as 'STEP 1: Train the RPN Network'. The RPN outputs 'Objects' and 'Bounding-box coordinates'. The main pipeline shows an input image going through a 'CNN AlexNet/VGG' to produce 'Feature maps'. These are then processed by an 'ROI Pooling' layer to produce 'Scale Invariant Features'. These features go into an 'FC Network' (Fully Connected Network), which has two parallel paths: one for 'Classification (Softmax)' and another for 'Bounding-box regression'. The outputs of these two paths are combined at a summation node (+).

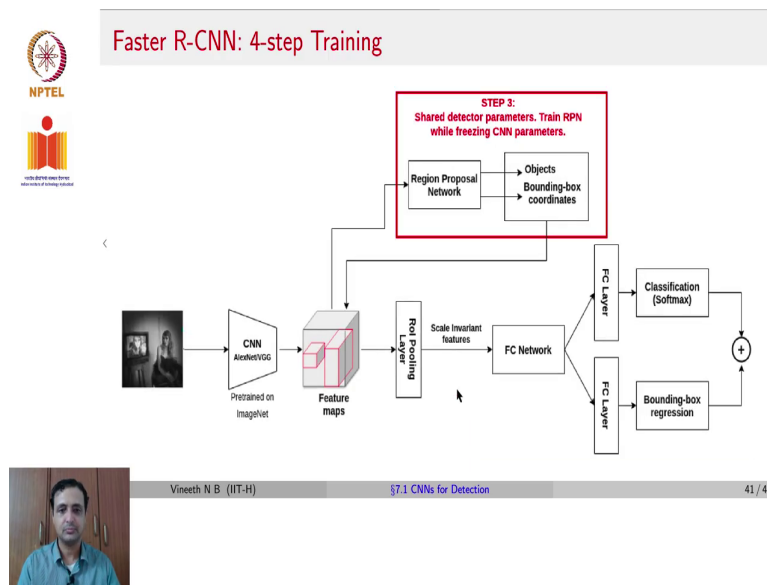
So in the first, you take the input image, you take a CNN pre trained on Alexnet, use the feature maps, send it through a region proposal networks which is a further set of convolutional layers and check for objectness and bounding box coordinates and train this region proposal part of the network, you can fine tune the previous weights too. This is step 1.

(Refer Slide Time: 24:24)



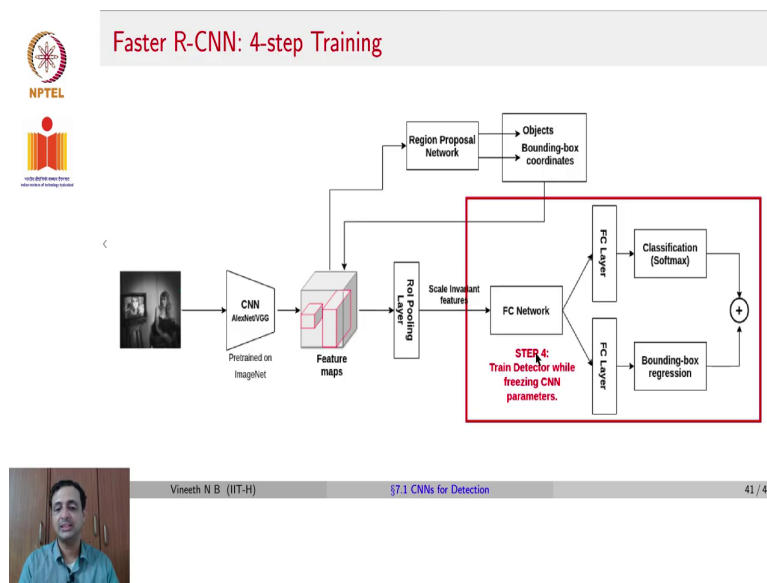
In the second step you take these region proposals obtained from the region proposal network, feed that onto the feature maps. Those are going to be your regions, now the rest of it stays very similar to fast R-CNN and you train all these weights using your classification loss and bounding box regression loss where now the classification loss tries to categorise region into any of say, 20 objects or how many ever objects you have in a given problem setting. So, this is the second step which is very similar to what is done in fast R-CNN.

(Refer Slide Time: 25:08)



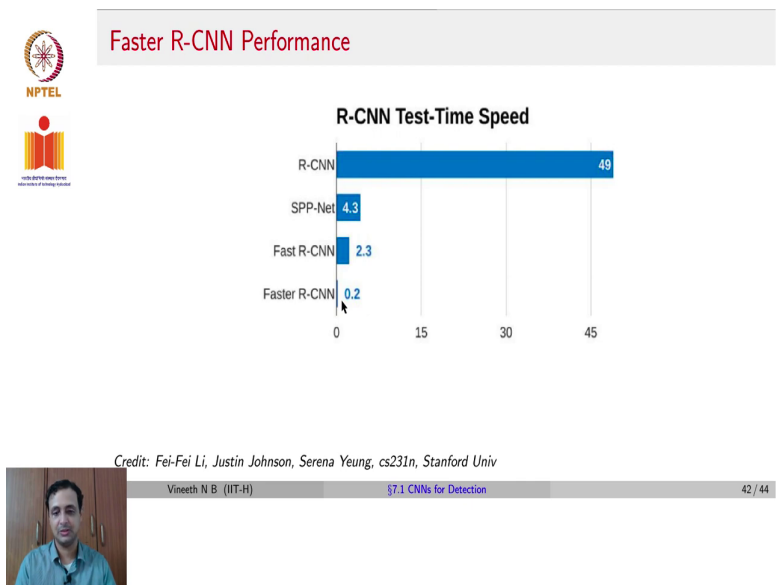
In step 3, the CNN is made common for your entire network as well as the region proposal network, so you share those parameters and you train your RPN while freezing your CNN parameters to help improve the RPNs performance.

(Refer Slide Time: 25:30)



And finally in step 4, you freeze everything else in the network and only fine tune your FC networks to get your final classification output and bounding box offset regression outputs, this is the four step training procedure proposed by faster R-CNN.

(Refer Slide Time: 25:50)



And using this approach the test time speed reduces from 2.3 seconds to 0.2 seconds which becomes closer to real time for every image.

(Refer Slide Time: 26:04)

Homework

Readings

- [Viola-Jones Object Detection Framework](#)
- [Object Detection for Dummies \(Parts 1-3\)](#)
- [Understanding OverFeat](#)

Video Series

- [Evolution of Object Detection Models \(Chapter 5-8\)](#)

Exercise

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

Vineeth N B (IIT-H) §7.1 CNNs for Detection 43 / 44

The homework for this lecture would be to go through the Viola Jones object detection framework and a very nice write up called object detection for dummies linked here and if you like to understand OverFeat here is a link. There is also a nice video series on evolution of object detection models if you like and here is the exercise that we left for you, smooth L1 loss is less sensitive to outliers than L2 loss as was proposed in fast R-CNN, can you try to

find out why? The next lecture, we will talk about dense sampling methods for object detection.

(Refer Slide Time: 26:47)



References

- Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. Vol. 1. IEEE. 2001, pp. 1-1.
- Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 886-893.
- Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580-587.
- Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440-1448.
- Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91-99.



Vineeth N B (IIT-H)

§7.1 CNNs for Detection

44 / 44