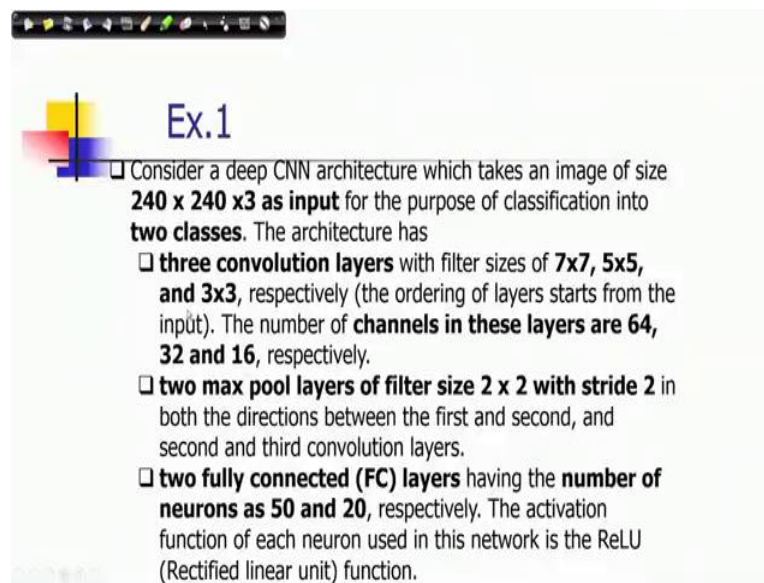


Computer Vision
Prof. Jayanta Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 58
Deep Neural Architecture and Applications Part – IV

We continue our discussion on Deep Neural Architecture and Applications. In the last lecture we have discussed about convolutional neural networks and we have also considered different architectures of convolutional neural network.

(Refer Slide Time: 00:35)



Ex.1

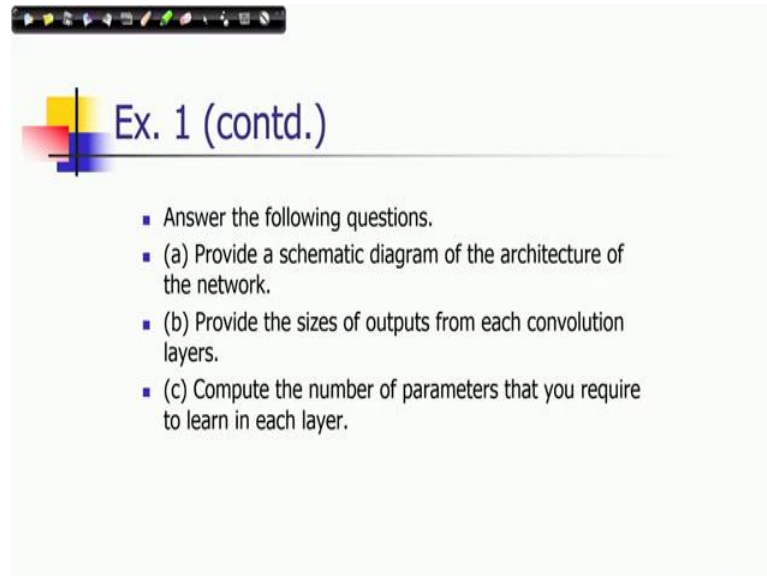
- Consider a deep CNN architecture which takes an image of size **240 x 240 x 3 as input** for the purpose of classification into **two classes**. The architecture has
 - **three convolution layers** with filter sizes of **7x7, 5x5, and 3x3**, respectively (the ordering of layers starts from the input). The number of **channels in these layers are 64, 32 and 16**, respectively.
 - **two max pool layers of filter size 2 x 2 with stride 2** in both the directions between the first and second, and second and third convolution layers.
 - **two fully connected (FC) layers** having the **number of neurons as 50 and 20**, respectively. The activation function of each neuron used in this network is the ReLU (Rectified linear unit) function.

Let us consider a problem at this stage to find out the number of parameters and also to determine the sizes of the output or input to a particular layer given a specification of convolutional neural network. So, consider this exercise, a deep CNN architecture which takes an image of size $240 \times 240 \times 3$ as input.

So, in the input there are 3 channel input and for the purpose of classification into two classes and the architecture has three convolution layers, where in each layer the filter sizes are of 7×7 , 5×5 and 3×3 and the number of channels in this layers are 64, 32 and 16. So, these layers they are specified from input to the output and then there are two max pool layers in between every convolutional layers.

So, which means two max pool layers of filter size 2×2 with stride 2 in both the directions between the first and second and second and third convolutional layers. And after the third convolutional layers we have two fully connected layers having the number of neurons as 50 and 20 respectively. And the activation function of each neuron used in this network is the ReLU or rectified linear unit function.

(Refer Slide Time: 02:09)



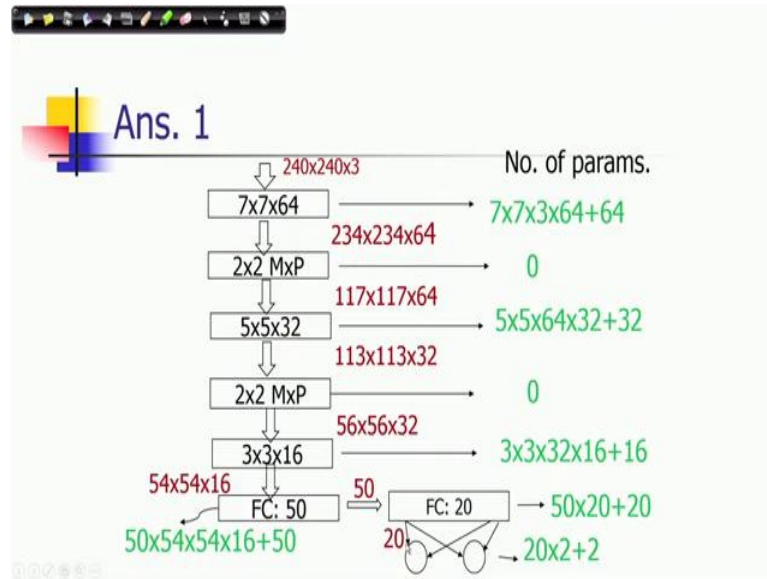
The slide is titled "Ex. 1 (contd.)" and contains a list of three questions. The slide has a light green background and a dark toolbar at the top. The title is in blue text with a small graphic to its left. The questions are listed with blue square bullet points.

Ex. 1 (contd.)

- Answer the following questions.
- (a) Provide a schematic diagram of the architecture of the network.
- (b) Provide the sizes of outputs from each convolution layers.
- (c) Compute the number of parameters that you require to learn in each layer.

So, the problem here is that we have to provide a schematic diagram of the architecture of the network, provide the sizes of outputs from each convolutional layers and compute the number of parameters that you require to learn in each layer. So, let us see how we can find or we can compute which have been asked here.

(Refer Slide Time: 02:33)



So, let us consider the architecture, a schematic diagram of the architecture itself. So, as you can see that there is an input layer and which is going to the first convolutional layer of filter size 7×7 . And since there are 64 channels so, it should be a filter size of $7 \times 7 \times 64$. Then it is followed by a max pool layer of 2×2 and stride is 2 there.

And, then again the output from the max pool layer is fed to the fed to a another convolution layers whose filter size is $5 \times 5 \times 32$, because there are 32 channels output from that convolution layer. And, then again it is followed by a max pool layer of 2×2 size, this is the third convolution layer or final convolution layer of our architecture.

So, output from this max pool layer it is fed to that convolution layer where filter size is of 3×3 and also the there are 16 channel output. So, it is $3 \times 3 \times 16$ that is what will be considering, that is not exactly filter size it is 6; the last 16 is showing it is the 16 channel outputs. Filter size will be 3×3 will see that thing. And, then this is being fed to the fully convolutional layers where there are 50 neurons.

So, the output from the 16 channel output from the third convolution layer, they are all flattened and they are considered as a say linear vector and or in the 1 dimensional feature vector. And, then that is fed to the fully connected layers of 50 neurons, this is also this output from this layer also is fed to another fully connected layer of 20 neurons and then those are again connected to output neurons. So, this is the output layer where there are 2 neurons and it is used for the classification because there are 2 class

classification problems; so, there are 2 neurons at the output. So, this is a schematic diagram of the architecture.

Let us now determine what should be the sizes of outputs from each layer. So, the input here as I has been specified that we have it will be handling and a size of $240 \times 240 \times 3$ this is the input size. So, the output if I consider 7×7 filter and as you know that it is only the output should be from those pixels of the input where this filter is fully embedded. So, filter size is $7 \times 7 \times 3$; that means, it is the number of channels of the input that would determine the depth of the filter and 7×7 is its width and height.

So, filter size is 7 cross 7 not 64, if I have mentioned wrongly earlier 64 is the number of channel here in this architecture output channel. So, that $7 \times 7 \times 3$ filter is embedded and then fully embedded on those pixels only from there the output has been provided. So, you have to leave around those boundary pixels where, it could not be fully embedded.

So, you have to basically subtract 6 pixels from width and height and only 1 plane will be coming out of this process. So, you have 234×234 and the since the output has 64 channels; so, there are 64 such filters. So, the output is $234 \times 234 \times 64$. So, this is our output size from the first convolution layer, then this goes to the max pool layer of size 2×2 and stride 2 which means they fill the input size should be half of the output of max pool layer should be half of the input size and number of channel will remain same which is $117 \times 117 \times 64$.

Then it goes to the next convolution layer where the filter is a 5×5 and since there are 64 channel input; so, filter size would be $5 \times 5 \times 64$ that is the depth of the filter. Once again you get only know 1 plane output 1 channel output from each filter there are 32 such filters.

So, you have to leave aside those boundary pixels where they are not fully embedded. So, 117 you have to subtract 4 from there so, it would be 113×113 and since there are 32 channels; so, it is $113 \times 113 \times 32$. This will again go to the next max pool layer of 2×2 and it has to be divided by 2 and since it requires full embedding of max pool layer; so, it would be only the lower ceiling or of the $\frac{113}{2}$.

So, it is $56 \times 56 \times 32$ channel remains same. Once again $3 \times 3 \times 16$ filter there is a third convolution layer, actually size is $3 \times 3 \times 32$ and there are 16 just filters. So, you have to subtract 2 from 56 and the size will be $54 \times 54 \times 16$ because there are 16 channels. Now, the all these 2 dimensional array output, it will be flatted into a 1 dimensional vector of size $54 \times 54 \times 16$ and that would be input to the fully connected layer of 50 neurons.

So, each neuron will have that many number of weights. So, the total number of weight will be coming to the total number of weight whatever. So, output of 50 neurons from the fully connected layer would be 50 only. And then output of 20 neurons from the fully connected layer it will be 20 only and then you have 2 class outputs. So, there will be from there will be 2 class output.

So, which is not written here, but there are even 2 outputs from this layer. So, this is the sizes of different outputs from different layers and next we are going to count the number of parameters. So, I have indicated by arrows by showing that we will be counting the parameters of those layers which is associated with those layers. So, consider the first layer so, we will be considering number of parameters. Consider the first convolution layer as the filter size is $7 \times 7 \times 3$ because input is 3 channel. So, there 1 filter size would be $7 \times 7 \times 3$, but there are 64 such filters.

So, it would be $64 \times 7 \times 7 \times 3$ that many number of weights and each 64 filters will have also bias; so, plus 64 bias. So, in the green color I am showing this count of parameters. So, this is answer that the number of parameters in the first convolution layers will be this.

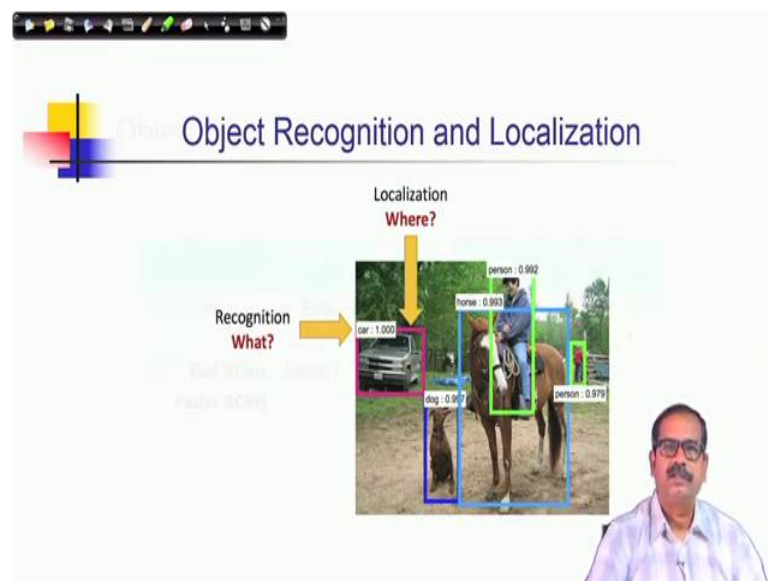
Similarly, for the max pool layer since we know that max pool layer it does not require any weight, it is already specified by its sizes itself. So, the number of parameters would be 0 there. So, that is what is max pool layer, next layer 5×5 then the filter size is 5×5 is 64 depth is 64. And how many filters are there? There are 32. So, it should be $5 \times 5 \times 64 \times 32$, that many weights plus there are 32 filters; so, there are 32 bias.

So, this should be the number of parameters from this layer. Again there is a max pool layer so, we will have 0 here. In the next it will be fed to the 3×3 filters and depth would be 32 because, the input to that layer it has 32 channels. So, it is $3 \times 3 \times 32$ and there are 16 filters. So, $3 \times 3 \times 32 \times 16$ plus for each 16 filters there are 16 biases. So, we

will have $(3 \times 3 \times 32 \times 16) + 16$ Next it goes to the know fully connected layer and as I mentioned that the size of the input here is $54 \times 54 \times 16$ each input will have a weight to the fully connected there and there will be 50 such neurons; so, there will be 50 such bias. So, it should be $(50 \times 54 \times 54 \times 16) + 50$ so, that is what here and here the size is 50.

So, it will be 50 weights should be connected to every neuron of 20 neurons and there are 20 biases. So, it is $(50 \times 20) + 20$ and again since there are know still you require more weights because this is the output layer. So, output here its size is 20; so, each neuron will have 20 weights and 1 bias so, it should be $(20 \times 2) + 2$ So, this is the total number of parameters those are involved in this architecture, what I have specified in the problem statement. Understand that this is how that parameters and sizes could be determined given the specifications of a particular CNN.

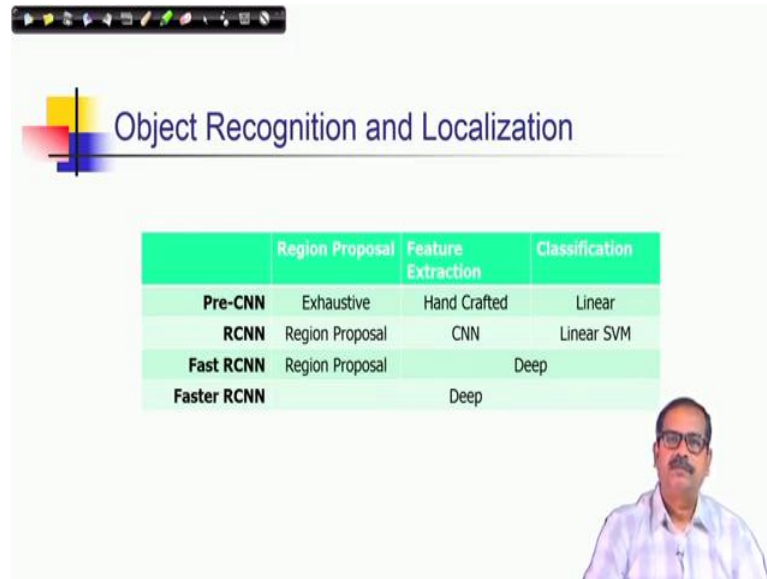
(Refer Slide Time: 12:22)



So, let us now consider another kind of problem which can be solved once again using convolutional neural network. This is this problem is called object recognition and localization. So, in this problem the task here is that given an image not only you have to identify the level of the object, but also you have to identify which portion of the image where this object lies, in the form of a rectangular block.

So, any rectangular block can be defined by the two corners of the block in pixel coordinates. So, that is how the specification of rectangular block will be there.

(Refer Slide Time: 13:05)



The slide features a title 'Object Recognition and Localization' with a decorative graphic to its left. Below the title is a table with four columns: 'Region Proposal', 'Feature Extraction', and 'Classification'. The table compares four methods: Pre-CNN, RCNN, Fast RCNN, and Faster RCNN. A small video inset in the bottom right corner shows a man with glasses speaking.

	Region Proposal	Feature Extraction	Classification
Pre-CNN	Exhaustive	Hand Crafted	Linear
RCNN	Region Proposal	CNN	Linear SVM
Fast RCNN	Region Proposal		Deep
Faster RCNN		Deep	

So, there are different techniques, this is a very classical problem and even before the deep neural architecture there are methods by which people have tried to solve it. So, this is a pre-CNN and the fact is that the object while it is which the region of the object in the image that needs to be identified. And, we call that task is as a region proposal and which means you would like to find out a probable region which contains an object and then you try to find out that what is the object in that region.

So, this is a kind of approach; there are stages like first you propose the region and then you describe that region by features. So, you have to extract features and then use those features to classify and if you could find some objects in your target in your classes. Then how you decide about that, otherwise you may say that region does not contain any object. So, there are various methods which have been reported.

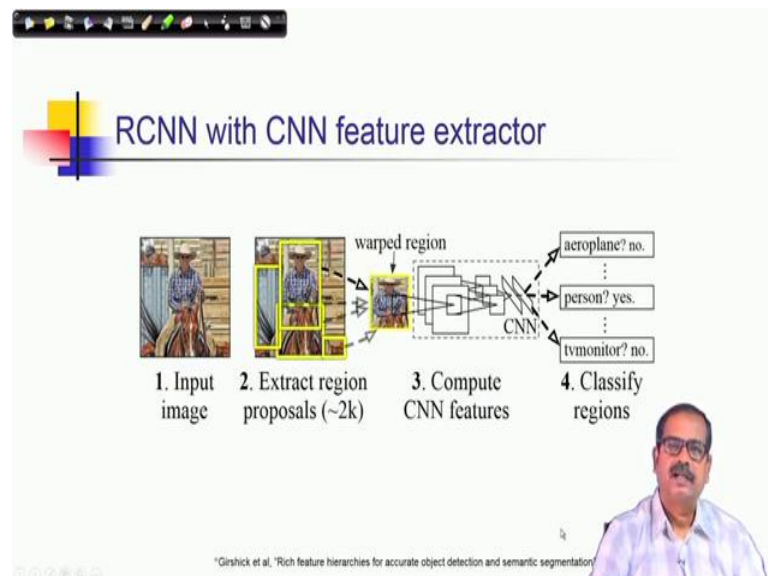
So, one method was very exhaustive means you are searching every rectangular type of rectangular block which is a combinatorially a hard problem, you need to use some heuristics there. So, that is what you should do it. And there are methods also to propose you certain heuristics to propose those regions. And then for feature extraction you can use some handcrafted features. Handcrafted features I mentioned what is used considered as a handcrafted features by considering the classes of objects, you predetermine that what kind of features will be computing.

For example, you can use shift operator, you can use the descriptor of shape or other kind of key point detectors and key point descriptors those. And, then combine them aggregate them to define a feature. And, then there are different classification algorithms you can use like linear classification algorithms. And, for deep neural architecture based technique there are three such variations. So, very first a technique called RCNN has been proposed.

So, RCNN what it does? It considers the task of region proposal should come from the conventional methods and then the feature extraction part is only should be done using CNN. So, that you do not have to use any handcrafted design. So, as per your problem features could be extracted learnt and extracted and then use a classifier to find out the object.

For example, linear SVM that is a classical classifier a rather another improvement or variations over on top of RCNN, where the both feature extraction and classification is done by a deep neural network. Whereas, the region proposal is still carried out using the conventional methods and the third variation that whole thing can be performed can be done is in a deep neural architecture.

(Refer Slide Time: 16:29)

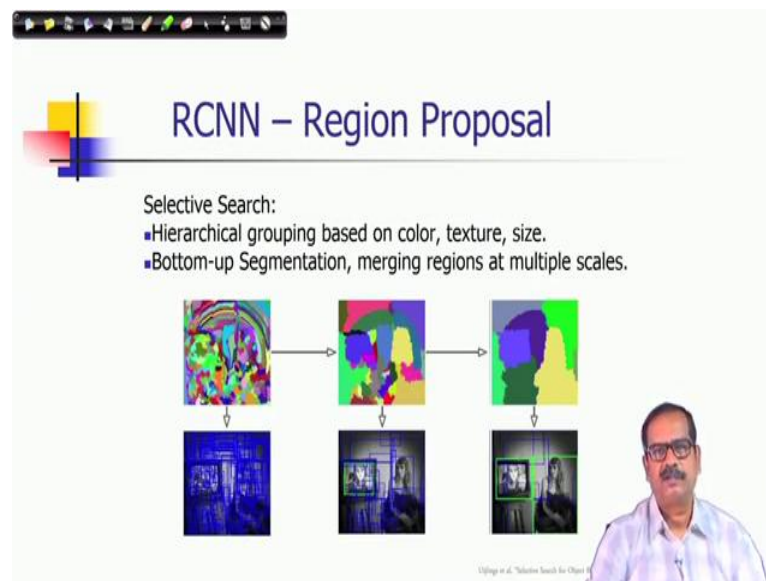


So, this diagram it is explaining this flow of this computation. So, you have an input image and then the regions are proposed. So, these yellow boxes are proposed regions by conventional methods, then you extract them one by one and then fed it to the CNN, as

CNN takes a fixed input size. So, you need to work those regions and then you know compute the features.

So, last layer of the CNN will give you a feature descriptor and which is used in a classifier. Once again this classifier is the conventional classifier and then you decide what kind of class it is based on those target classes of this classifier.

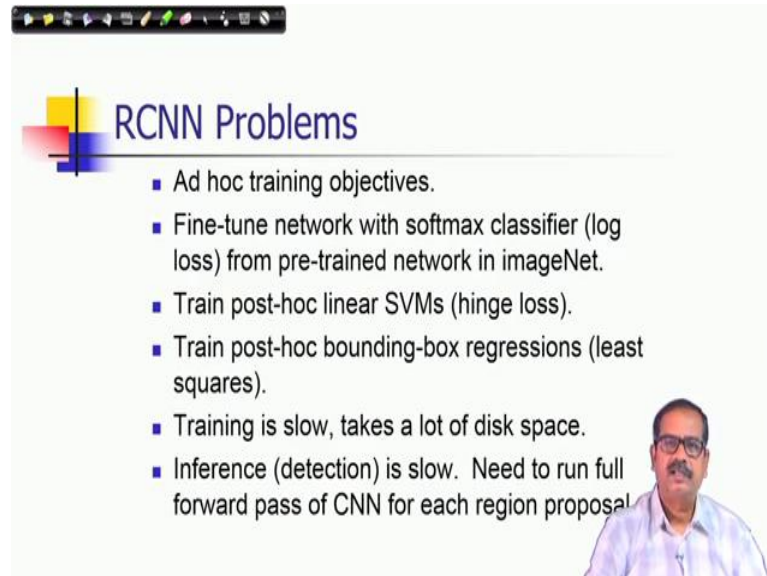
(Refer Slide Time: 17:13)



For this particular method as I mentioned that it uses conventional method for proposing regions. So, there is a method based on selective search which uses hierarchical grouping based on color texture size and it is a bottom up method. So, it performs bottom up segmentation then it merge regions at multiple skills and then based on the regions property it decides whether it contains any object or not. So, this figure tries to summarize this process.

So, it considers very small segments at the bottom layer, then it merges it and the bigger segments and it tries to identify out of them some of them are considered that they contain the object. And then you use the rectangular box enclosing those particular segments.

(Refer Slide Time: 18:12)



The slide is titled "RCNN Problems" and features a list of six bullet points. To the right of the text is a small video inset showing a man with glasses and a mustache, wearing a light blue shirt, speaking. The slide also has a decorative graphic on the left consisting of overlapping colored squares (yellow, red, blue) and a black crosshair.

- Ad hoc training objectives.
- Fine-tune network with softmax classifier (log loss) from pre-trained network in imageNet.
- Train post-hoc linear SVMs (hinge loss).
- Train post-hoc bounding-box regressions (least squares).
- Training is slow, takes a lot of disk space.
- Inference (detection) is slow. Need to run full forward pass of CNN for each region proposal

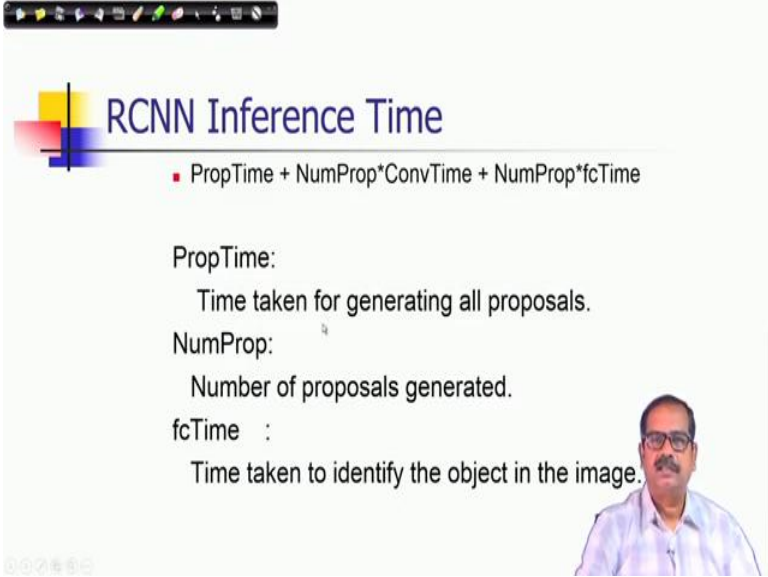
Their problem in this method is that the training objectives are not very clearly mentioned here. As you can see that the CNN is a feature extractor stage, but know what is what should be the objective function of that CNN. So, far we have discussed about training of CNN based on some classification task. So, even the features are learned based on the end objective of classification, but in this case since the classifier is a conventional classifiers; so, classifier is independent, independently designed.

So, this is a problem; so, you have to consider some intermediate objectives to learn the CNN. So, kind of adhoc is remains there. So, you can use some pre trained network, but now again you need to do fine tuning based on your domain, based on your problem domain. So, that is required or and also we require to once again train the SVMs, Support Vector Machines based on those features and usually feature descriptors are quite large; so, this will take lot of time. So, and also another training you have to do because you have to find out the regions.

So, though the regions are proposed, but you need to do some kind of fine tuning. So, some kind of regressions based on those proposed regions you need to do; so, that also requires some training. So, there are so, many different kinds of trainings are involved that is why it is very slow and also it takes a lot of disk space. So, inference is also slow, it need to run full forward process of CNN for each region proposal.

So, that is what because for each region proposal you are extracting features independently and you will need it so, many times you need to iterate this process.

(Refer Slide Time: 20:05)



RCNN Inference Time

- $\text{PropTime} + \text{NumProp} \cdot \text{ConvTime} + \text{NumProp} \cdot \text{fcTime}$

PropTime:
Time taken for generating all proposals.

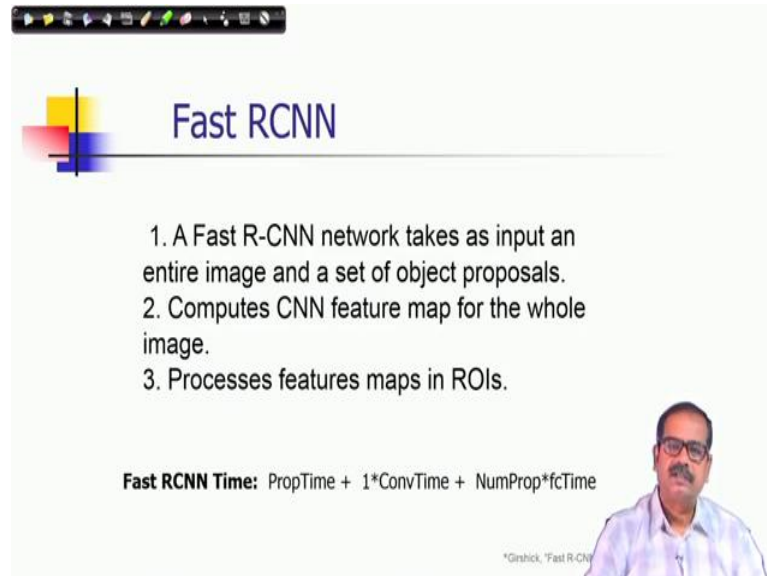
NumProp:
Number of proposals generated.

fcTime :
Time taken to identify the object in the image.

So, that is why its inference time is very large, it takes quite a bit of time. So, what are the times components here? You can see there is a time called prop time or proposal time. So, it is time taken for generating all proposals, then number of depending upon number of proposals that is generated at this stage. So, each proposal or each region has to be fed to the CNN.

So, it extracts the feature. So, it is number of proposals into the time taken for computing features using CNN and finally, each one once again has to be classified. So, that there is a classification time and time taken to identify the object in the image. So, this is a total inference time, if I consider for RCNN.

(Refer Slide Time: 21:06)



Fast RCNN

1. A Fast R-CNN network takes as input an entire image and a set of object proposals.
2. Computes CNN feature map for the whole image.
3. Processes features maps in ROIs.

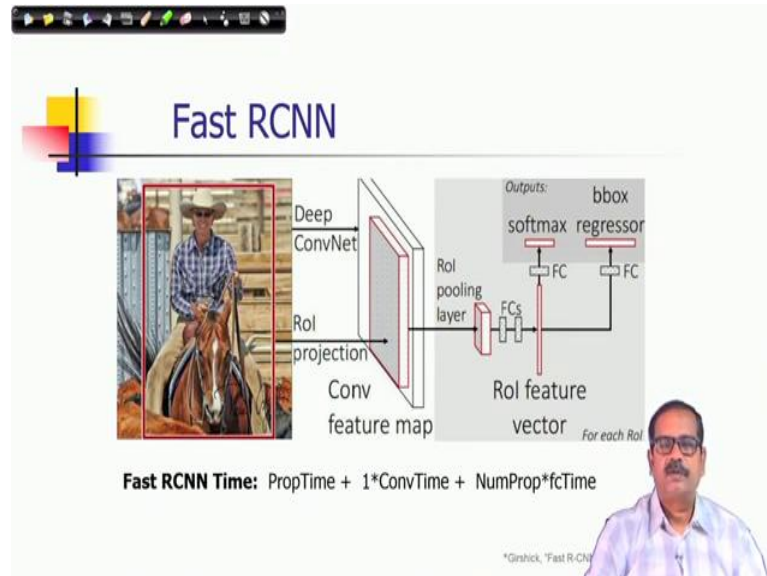
Fast RCNN Time: $\text{PropTime} + 1 * \text{ConvTime} + \text{NumProp} * \text{fcTime}$

*Girshick, 'Fast R-CNN'

So, to an improvement over this particular approach is that you can use a first RCNN network. And here the idea is that instead of using an SVM, we can use also the whole thing classification and feature extraction can be carried out at the same go. That means, for every region it will be passing through both feature extraction and classification in this case.

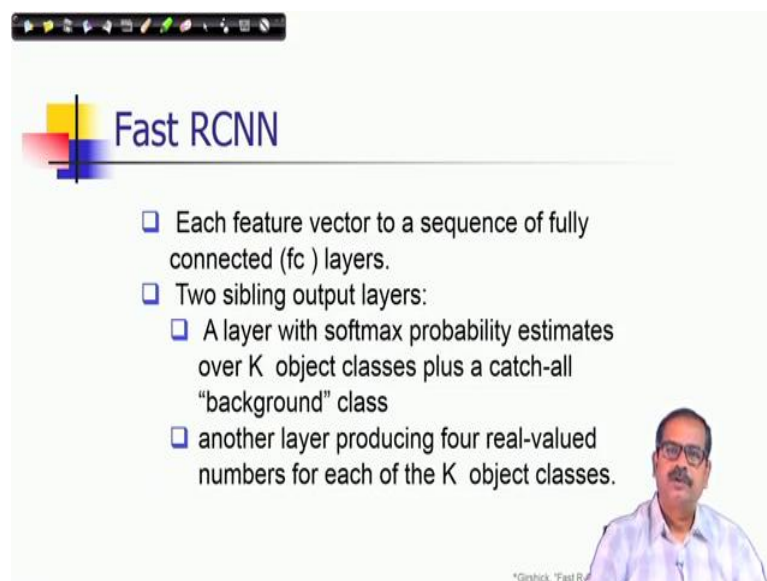
So, it computes CNN feature a map of the for the whole image. So, and also it is not required to compute the feature map of the whole image. You can just process the feature maps in ROIs and you know here only the thing is that it pools the ROIs and then again it passes to the classification stage. So, it is not that deep neural network a deep architecture is not there at the classification stage it is a conventional classification.

(Refer Slide Time: 22:12)



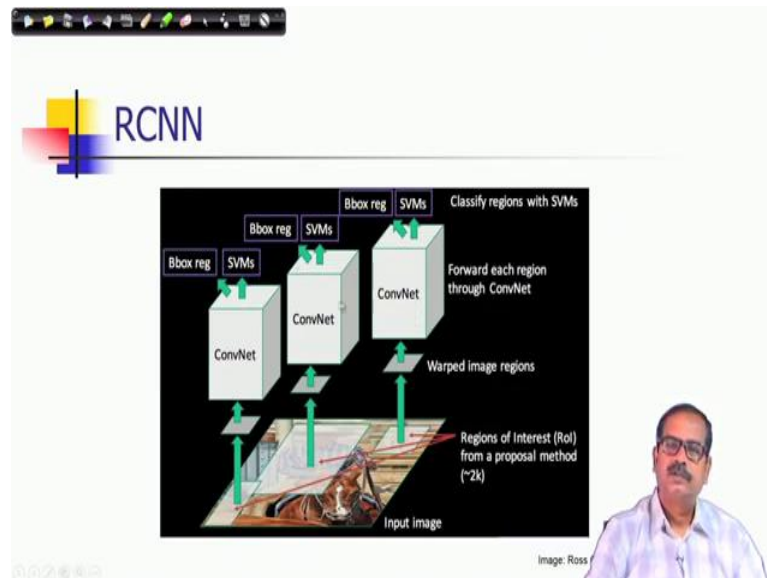
So, this is what it is doing it, it only uses the same conventional neural network. So, given the image it computes the whole feature set, then using the region proposal it pools those features and use a classifier to compute to classify it and also the objects. So, the time here is the proposal time it remains proposal time and then since it is only computing 1 time all the features, it is only 1 time convolution network is used. So, it is $1 \times$ that conv time, but every region every proposal is separately classified. So, it is number of proposals \times fc time.

(Refer Slide Time: 22:59)



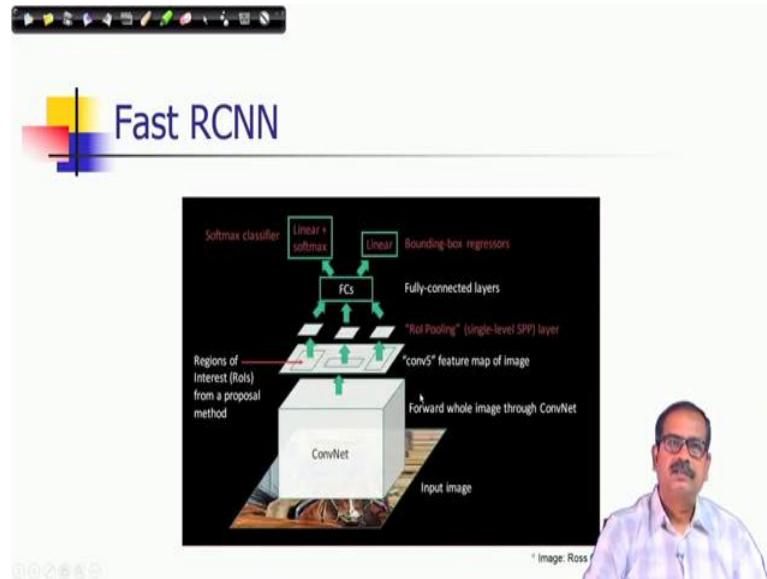
So, first RCNN the features are that each which are vector to sequence a fully connected layers and two sibling output layers. So, a layer with the softmax probability estimates of our K object classes plus a catch all background class. So, you are considering that and another layer producing four real valued number for each of the K object classes, this is actually regression of the regions.

(Refer Slide Time: 23:25)



So, it just shows the architectures of RCNN, you can see that every we proposed region is passed through separately through the convolution layers. And, then those are used for classifying and regression of the regions using regression module and also the SVMs for classifications; every region every regions are processed in this way.

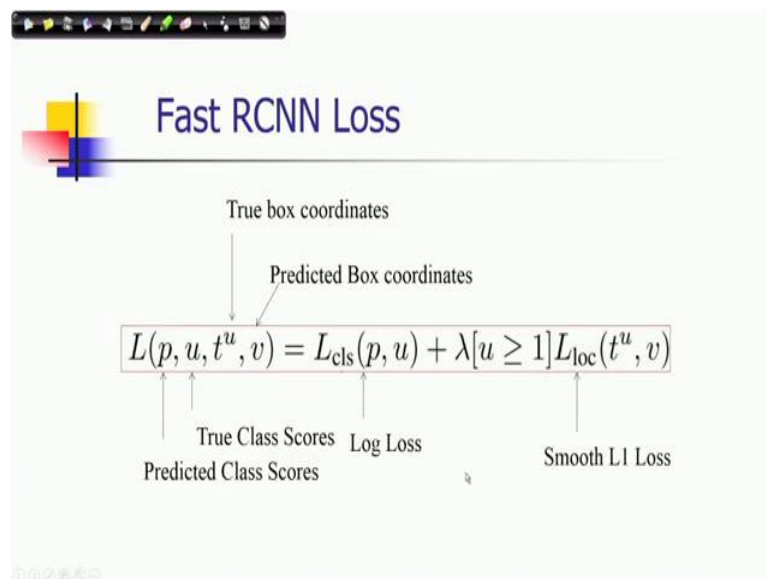
(Refer Slide Time: 23:55)



Whereas, in fast RCNN you have the only one time the convolutional neural network is used to generate all the features and also it has propose the regions. So, use those regions in the feature map those are used for classification; so, those are fed again to the fully connected layers and then.

So, it is a classifier which could be also neural network classifier like ANN classifier and those are used for classifying the object and also the finding out the regions; that means, fine tuning of estimates of the regions using a regression box.

(Refer Slide Time: 24:41)

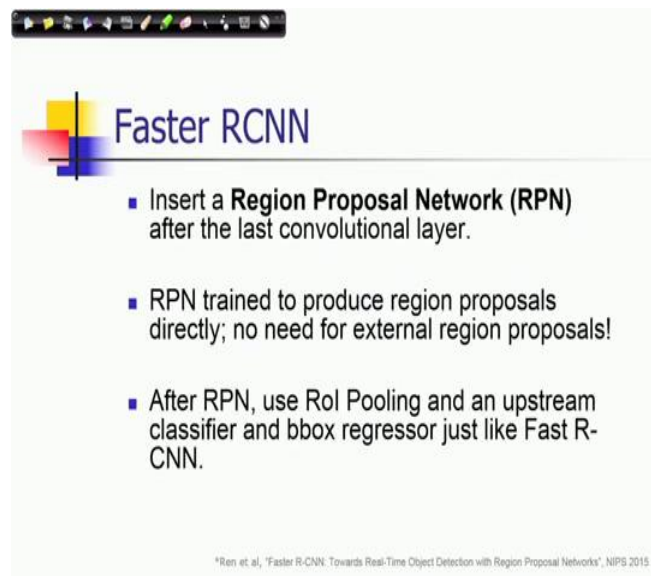


The loss what is used in the stages classification stage as I have shown where there is a fully connected layer which is simultaneously classifying the object and also detecting the box or a rectangular area by the object is contained. So, there are two components of this loss; so, one component is due to classification and the other component is due to the regression.

So, and we are only considering those samples those classes which are object classes. So, u is the ground truth, p is the predicted class. So, if only the it is a object class then only we are considering the loss due to the regression and this is a linear regression λ is any parameter. So, these this is the thing this is the \log loss which is cross entropy loss.

This is predicted class course, p is the predicted class course and u is the true class scores. And, this is the smooth L1 loss, L1 is the function which is used for computing the regression the box, the error deviation of the localizations or deviation of the rectangles from the true rectangle to the predicted rectangle. So, t'' is the true box coordinates and v is the predicted box coordinates. So, this is how the loss function is defined.

(Refer Slide Time: 26:10)



Faster RCNN

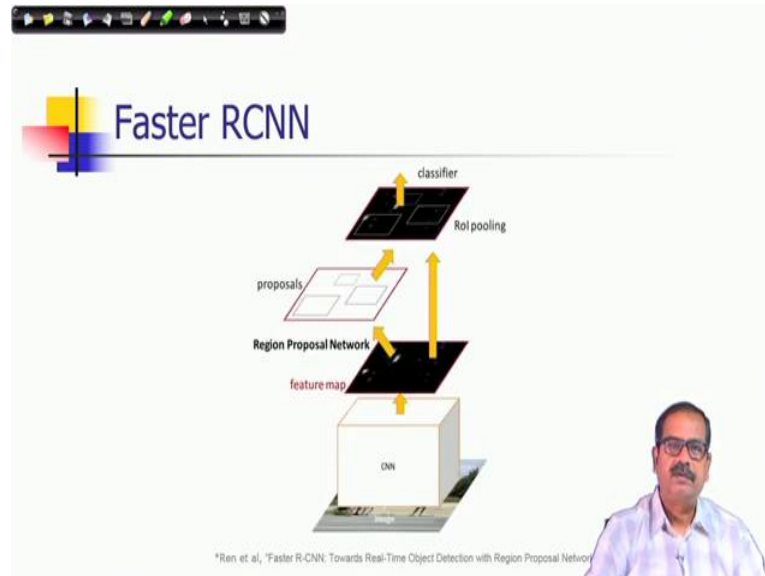
- Insert a **Region Proposal Network (RPN)** after the last convolutional layer.
- RPN trained to produce region proposals directly; no need for external region proposals!
- After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN.

*Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

So, then improvement over even faster RCNN is faster RCNN, where the whole task has been considered you doing it in a convolutional network; even the region proposal also is done using a convolutional network. So, we call it as a region proposal network. So, what we can do that using the features what is generated in by the convolutional layers, the same features are used for generating region proposals.

And, from the region proposals you pool those features which are there in that re-proposed region and used it for classification.

(Refer Slide Time: 26:53)



So, that is why it becomes so fast and you can see that the first stage is that you generate a feature map. So, use this feature map to propose regions and also use this feature map to classify and also regress the rectangular boxes.

(Refer Slide Time: 27:07)

Region Proposal Network

- Slide a small window on the feature map
- Build a small network for:
 - classifying object or not-object, and
 - regressing bbox locations
- Initial localization:
 - Position of the sliding window w.r.t image.
- Finer localization:
 - Box regression performs finer localization w.r.t this sliding window.

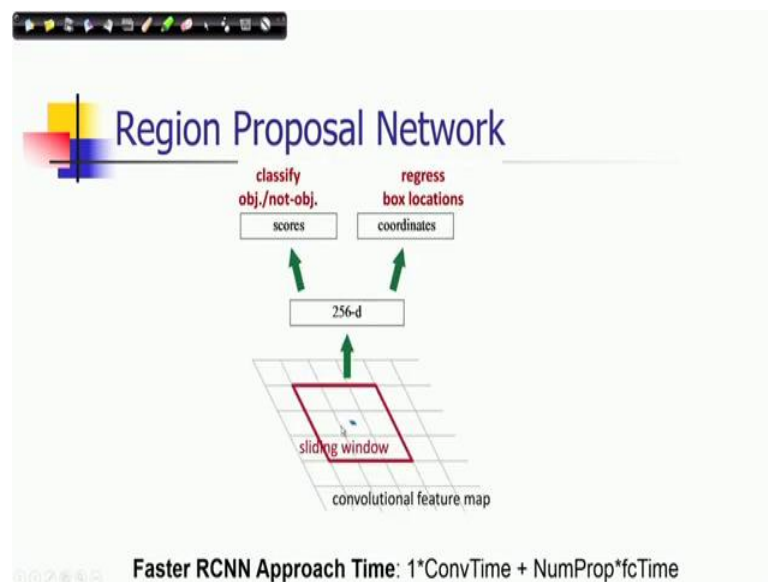
Faster RCNN Approach Time: $1 * \text{ConvTime} + \text{NumProp} * \text{fcTime}$

So, this is what is faster RCNN and in the region proposal network in the task is that we can slide a small window on the feature map and then we can build a small network for

classifying object or non-object. And so, here the task is that you have to classify the object or non-object and also regress the bounding box locations and this initial local this is the initial localization.

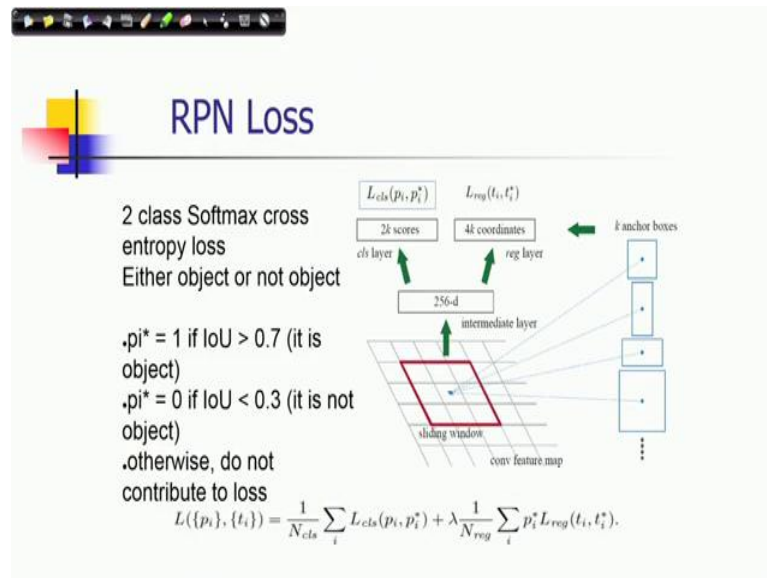
So, these positions are used and final localization is considered at the final stage of classification and regression, where box regression performs final localization. So, and the faster RCNN as you can see that it requires only there is no proposal time separately, it is 1 time convolutional time and then number of proposals into the time required for classification.

(Refer Slide Time: 27:57)



So, this is describing the region proposal network. So, you take a sliding window in the feature map and it is a 256 dimensional feature descriptor. Then you are classifying as object or non-object or regressing the box location. So, this will give you the initial box locations of the object, again in the final in the last stage when you are classifying the objects you can make a fine tune of that locations.

(Refer Slide Time: 28:25)



So, for region proposal network loss what it considers that in this sliding window there are different shapes of windows are also considered, which may contain a box or object. So, there are k anchor boxes; so, each one is tested whether it contains a object and also this box itself becomes the coordinates those are output of this region. So, it contains there are two components: one is once again classification loss, once again it is a regression loss. So, these two components are used in the loss function.

In the 2 class loss function standard cross entropy loss is used and because now you are deciding whether it contains object or not. So, if your predicted object is intersecting about 0.7 of the true object 70 percent of the true object, then we call this is 1 otherwise it is 0. So, it is intersection of union.

So, it is the ratio of intersection of predicted object and predicted region and true region divided by union of predicted region and true region. So, if this ratio is more than 0.7 then we call it is 1, if it is less than it is less than 0.3 it is 0 and otherwise it do not contribute to loss. So, with this let me take a break here and we will continues this discussion in the next lecture.

Thank you very much for listening.