




**Deep Learning for Computer Vision**  
**Professor. Vineeth N Balasubramaniam**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Hyderabad**  
**Lecture No. 47**  
**CNNs for Segmentation**

Having seen detection, we now move on to segmentation of images using CNN architectures.

(Refer Slide Time: 0:26)


Homework



**Exercises**

- 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?  $1/9$
- 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?  $19 \times 19 \times (5 \times 5 + 20) = 19 \times 19 \times 45$

$S \times S \times (5B + C)$



Vineeth N B (IIT-H) S7.3 CNNs for Segmentation 2 / 29

We will quickly review the exercise from the previous lecture. 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 overlapping region of  $1 \times 1$ . What is the IOU? This should be a simple one. So, you have an upper left, which is  $2 \times 2$  and then you have a lower right which is  $2 \times 3$  with a  $1 \times 1$  overlap. So, the total number of boxes here are 9 of which the intersection is one box. So, your overall IOU would be  $1/9$ .

And the second question, consider using YOLO on a  $19 \times 19$  grid, 20 classes, five anchor boxes, the output volume would be, remember we said it is  $S \times S \times (5B(\text{that is the no. of anchor boxes}) + C(\text{no. of classes}))$ . And that is the final answer you get. We assume here that  $C$ , the number of classes, also includes the background. If you want to have background as a separate class, then you will have to add  $C$  to be 21.

(Refer Slide Time: 1:47)



### Recall: Image Segmentation

- Image segmentation methods: Watershed, Graph Cut, Normalized Cut, Mean Shift, etc
- Classical segmentation methods inspired early versions of deep learning based methods for object detection; R-CNN used a version of min-cut segmentation method known as CPMC (Constrained Parametric Min Cuts) to generate region proposals for foreground segments
- Moving on to deep learning for segmentation now...



Vineeth N B (IIT-H)





§7.3 CNNs for Segmentation

3 / 29

So, let us recall image segmentation in the first place. Early on in the lectures, we talked about different image segmentation methods such as Watershed, Graph Cut, Normalized Cut, Mean Shift, so on and so forth. To an extent, these methods inspired early versions of CNN architectures for tasks such as detection and segmentation. In fact, R-CNN, which uses selective search to get region proposals, in fact used a min-cut segmentation method known as CPMC (Constrained Parametric Min Cuts) to generate the region proposals. But we will now move on to deep neural networks for segmentation.


(Refer Slide Time: 2:36)

**Semantic Segmentation**



- Task of grouping together similar (in semantic content) pixels in an image. How to formulate this using DNNs?  
**Cast as a pixel classification problem!**
- For each training image, each pixel is labeled with a semantic category. Quite annotation-intensive!
- Various architectures in recent years: FCN, SegNet, U-Net, PSP-Net, DeepLab and Mask R-CNN

Credit: Fei-Fei Li et al, CS231n, Stanford Univ



Vineeth N B (IIT-H) 87.3 CNNs for Segmentation 4 / 29

So, the task at hand for us to start with is going to be semantic segmentation, where we would like to assign a class label to every pixel in the image. How do you solve this problem using neural networks, in particular, convolutional neural networks? We are going to cast this as a pixel wise classification problem. While, so far we spoke about image level classification where we had only a label at the level of an image that we went to detection where we had labels at the levels of bounding boxes, where we not only gave class label, but also an offset to the bounding box to make a final prediction.

But in semantic segmentation, we are going to classify at the level of every pixel. So, for each image, each pixel has to be labelled to the semantic category. So, you can assume that creation of a dataset for semantic segmentation is very annotation intensive. There have been various architectures in recent years, such as FCN, SegNet, U-Net, PSP Net, DeepLab, Mask R-CNN. We will cover each of them in this lecture.

(Refer Slide Time: 3:58)



### Fully Convolutional Networks for Semantic Segmentation (FCN)<sup>1</sup>

- Adapts various classification networks (VGG net, GoogLeNet) into fully convolutional networks by converting FC layers into  $1 \times 1$  conv layers
- To obtain classification for each pixel, another  $1 \times 1$  conv layer is appended with channel dimension  $C + 1$  where  $C$  is number of classes. **Do you see any problem?**
- Image classification architectures perform downsampling as they go deeper  $\implies$  fully convolutional architecture will have lower resolution than input. What to do?
- Perform upsampling to get back to original resolution. How to do?
- Learnable upsampling done through **Transpose Convolution**

<sup>1</sup>Shelhamer et al, Fully Convolutional Networks for Semantic Segmentation, TPAMI 2016



Starting with FCNs or Fully Convolutional Networks for Semantic Segmentation, which was proposed in 2015 and 16, they adopted various classification networks such as VGG net, GoogleNet so on and so forth into fully convolutional networks by converting the FC layers into  $1 \times 1$  layers. We saw that with single shot detection methods, it is the same idea here because for semantic segmentation you still have to classify at the level of a pixel. So you do not want to move away from the convolutional region.

What do we do to obtain classification for each pixel? We can have  $1 \times 1$  convolutional layer at the end of any CNN. And have  $C+1$  channels in that final output volume where  $C$  is the number of classes and the  $+1$  is for, say, a background class. But do you see any problem with this approach? There is a problem when you use convolutional layers, which is, all standard CNNs keep down sampling as you go further down the layer, be it through convolution or through pooling, which means your output feature maps are not going to be the same size as your input.

But we finally want to give a pixel level label in the input resolution, in the input images resolution. So, what do we do? We already know the answer. We can upsample after a certain stage. So, you have convolutional layers and after a certain stage you upsample and bring the feature maps to the same dimension as your input image. How do we upsample? One of the methods that we have discussed before is transpose convolution.

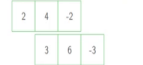
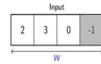
(Refer Slide Time: 6:00)



### Recall: Transpose Convolution

- Allows for learnable upsampling
- Also known as Deconvolution (bad) or Upconvolution
- Traditionally, we could achieve upsampling through interpolation or similar rules
- Why not allow the network to learn the rules by itself?
- Let us see a 1D example

Transposed convolution layer



Credit: Francois Fleuret



Vineeth N B (IIT-H)

§7.3 CNNs for Segmentation

6 / 29

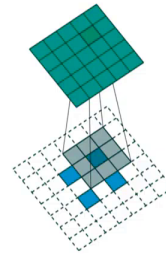
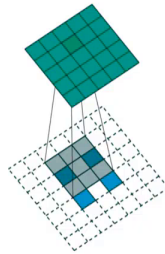
Let us quickly recall transpose convolution. So, this was the one dimensional example we spoke when we discussed convolutional neural networks. So, where your output can be larger than your input.

(Refer Slide Time: 6:15)



### Recall: Transpose Convolution

Dilated  
Atrous



Upsampling  $2 \times 2$  input to a  $5 \times 5$  output

Credit: Vincent Dumoulin



Vineeth N B (IIT-H)



§7.3 CNNs for Segmentation

7 / 29

And here is the illustration that we gave for transpose convolution where you have a dilation in your convolution when you take the filter and apply it on an input image. Because there is a dilation, transpose convolution is also known as dilated convolution, or it is also sometimes

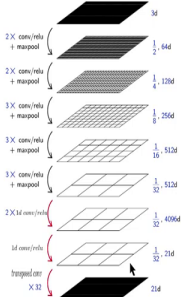
referred to as Atrous convolution, as we will see a little later because it dilates when you perform the convolution.


(Refer Slide Time: 6:51)

### FCN with VGG-16 Backbone

- Remove fully connected layers
- Replace three FC layers with 1D conv layers; last layer has  $C + 1$  filters which give class probabilities (note that spatial size is downsampled because of maxpool operations)
- One final upsampling layer to get back to original size
- Instead of one final upsampling layer, can have skip connections from earlier layers (we will see later how) for better boundaries





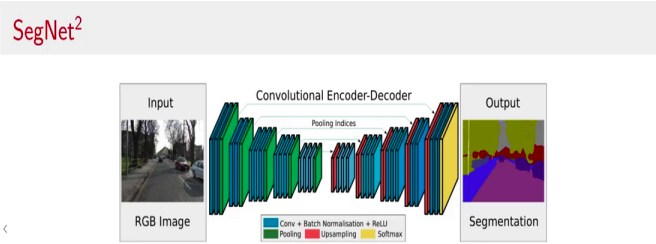
Vineeth N B. (IIT-H)
87.3 CNNs for Segmentation
8 / 29

*Credit: Francois Fleuret*

So, now in FCNs, this is the idea that is used to upsample after a certain level. Let us see an FCN architecture with a VGG backbone. So, if you have a VGG-16 backbone, the first thing that was done in FCN is to remove all the fully connected layers. So, what you have now are only the convolutional layers of VGG-16. So, you can see all the standard feature maps of VGG-16. The 3 FC layers are replaced with 1D convolutional layers. And the last 1D convolutional layers has  $C+1$  filters.

So, you can see here the last one has 21 dimensional filters. This was applied on a dataset that had 20 classes. So, that was 21 dimensional. However, you notice here that while you have 21 channels, the output is still smaller in size when compared to the input resolution. So, you finally have one upsampling layer which is done using transpose convolution to get back to the original size. This is what was done in fully convolutional networks. However, is transpose convolution the only way to obtain the upsampling? Not necessary. We can do several things, including skip connections, which we will see later.

(Refer Slide Time: 8:19)



SegNet<sup>2</sup>

- Fully convolutional encoder-decoder architecture
- Encoder is VGG-16 without FC layers
- Decoder maps low-resolution encoder feature maps to input resolution
- Decoder uses pooling indices of corresponding max-pooling steps to perform upsampling! These sparse upsampled maps are followed by conv layers to produce dense feature maps

<sup>2</sup>Badrinarayanan et al, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, TPAMI 2017

Vineeth N B (IIT-H) 87.3 CNNs for Segmentation 9 / 29

But before we go there, we will talk about another method that came around the same time as FCNs known as SegNets. SegNets are what you see here in the image. You have an input image, you have an output and the architecture is a fully convolutional encoder decoder architecture. How does, how is this different from FCNs? Very similar to FCNs. The encoder is VGG-16 without FC layers. The decoder maps the low resolution encoder feature maps to input resolution. Here, there is a difference from FCN.

FCN had only one upsampling layer towards the end, whereas SegNet has a sequence of decoder layers which mirror the architecture of your encoder. So, the encoder and decoder architectures are mirrors of each other. The important contribution in SegNet was to avoid transpose convolution in upsampling. So, how does SegNet do the upsampling in that case? It actually uses the pooling indices that were obtained in the encoder layers and uses them to be able to upsample.

(Refer Slide Time: 9:40)

**Upsampling in SegNet**

- Max pool indices: location of maximum feature value for each pooling window for each encoder feature map
- Storing maxpool indices is memory-efficient (instead of storing full feature map)
- In each stage of decoder, max pooling indices from corresponding encoder stage used to produce sparse upsampled feature maps
- What is the loss function in all these methods? **Sum of pixel-wise cross-entropy losses**

Vineeth N B (IIT-H) 87.3 CNNs for Segmentation 10 / 29

Let us see how this is done. So, if you had an output map from a particular decoder stage, we used a max pool indices from the original encoder. So, let us assume the maxpool indices were 00, 10, 10, 01. So, a would go to the 0, 0th location, which would be the top left. b would go to the 1 0th location, which would be the bottom left. c would go to the 1, 0th location, which would be the bottom left and d would be 0, 1 which would be top right. Now this becomes an upsampled version of a previous layers output map in the decoder. And one can continue to do this to get your final upsampled image.

What do we mean by max pool indices, in your encoder, whenever you had a pooling layer you keep track of the indices of which of those entries in  $2 \times 2$  patch or  $3 \times 3$  patch, depending on what pooling you are doing, you store the winning indices in a separate memory buffer and we are going to use that max pool indices to upsample back to higher resolutions. Why is this good? Storing max pool indices is more efficient than storing full feature maps of the encoder. And in each stage of the decoder, the max pooling indices of the corresponding encoder is used to produce the upsample feature maps.



So, if you see the complete image, the pooling indices of this encoder feature map goes to this decoder, this encoders feature map goes to this decoder and so on and so forth. What is the loss function? So, we have seen a couple of architectures now, FCNs and SegNets, what would one



use as a loss function to train such networks? Simple, it is the sum of pixel wise cross entropy losses. Remember, in both FCN and SegNet, your output is going to be a volume which would be the size of the input resolution, but the number of channels would be the number of classes+1 for background.

So, you can actually have for each pixel, a vector of probabilities, which is in the output value. Now you could compute the pixel level cross entropy for each pixel. You can sum all of them up and that becomes your loss function, which because is a sum of cross entropies will remain differentiable.

(Refer Slide Time: 12:23)




### U-Net<sup>3</sup>

- Fully convolutional encoder-decoder architecture with skip connections (an extension of FCN)
- Contracting path is an existing classification network with FC layers removed
- Each step in expanding path consists of upsampling of feature maps, concatenated with corresponding feature maps of contracting path followed by further conv layers
- Final layer is  $1 \times 1$  conv with  $C + 1$  channels,  $C$  being number of classes
- Upsampling performed by  $2 \times 2$  transpose conv with  $s = 2, p = 0$  with number of feature channels being halved each time (this operation doubles input map dimensions)
- In original architecture, unpadding convolutions are performed, making output segmentation map smaller than input by a constant border width

<sup>3</sup>Ronneberger et al, U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015

Vineeth N B (IIT-H) [§7.3 CNNs for Segmentation](#) 11 / 29





A third kind of architecture, which also came in the 2015-16 time frame, is known as U-net. U-Net is also a fully convolutional, encoder decoder architecture. But it introduces the concept of skip connections in a different way from SegNet. In the contracting path, that is the encoder path is an existing classification network with FC layers removed similar to SegNet and FCNs. But in the expanding path, in addition to upsampling of feature maps, the network also receives corresponding feature maps of the contracting path along with further convolutional layers.

Finally, at the end, you have a  $1 \times 1$  convolution with  $C+1$  channels where  $C$  being the number of classes. The Upsampling in U-Net is performed using transpose convolution,  $2 \times 2$  transpose convolution with stride 2 and padding 0. And the number of feature channels are halved in each

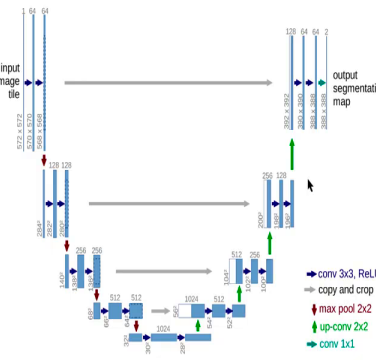
upsampling step. Which means the total input dimensions will double now with the  $2 \times 2$  transpose convolution and channels being halved, the total input map dimensions will double at this time. So, the original unit architecture unpadded convolutions are performed, which means the output segmentation is smaller than the input by a constant border width.


(Refer Slide Time: 14:00)

### U-Net Architecture

- One modification from FCN is: upsampling part has large number of feature channels
- Concatenation of feature maps from encoder gives localization information to decoder
- Other ways of concatenation include simply summing up feature maps





Vineeth N B. (IIT-H)
§7.3 CNNs for Segmentation
12 / 29

Let us look at the visual. So, here is the U-Net and the shape of it tells why it is called the U-Net architecture. You can once again see the contracting path is similar to any other architecture without FCN layers. And in the expanding path in addition to doing transpose convolution at each step, you also receive the corresponding feature maps of your contracting part, which is added to your expanding path. So, that gives a localization information from the encoder to each step of the decoder beyond just upsampling. So, you get additional information from the corresponding layer of the encoder rather than rely only on upsampling as in SegNet or FCNs.

(Refer Slide Time: 14:57)



### U-Net: Other Features and Extensions

- Designed for biomedical image segmentation; **How does this affect the model learning?**
- Very few training images available, hence excessive data augmentation is performed by applying elastic deformations to available training images
- Another challenge in medical domain: separation of touching objects of same class; hence, U-Net proposes a weighted loss to penalize pixels closer to edges
- Variants of U-Net: Same principles but with different kinds of blocks (dense blocks instead of conv blocks), depths, etc



Vineeth N B (IIT-H)

§7.3 CNNs for Segmentation

13 / 29

U-Net was originally proposed for biomedical image segmentation. It was, in fact, published in a medical conference, Mechai. How do you think this can affect the learning of the unit model? Biomedical image segmentation has some unique challenges, firstly, often you do not have training datasets which are of the order of 1 million or even tens of thousands. Very few training images may be available. So, this particular work employed excessive data augmentation by applying elastic deformations to available training images, which was one of the contributions.

Another challenge in the medical domain is there could be many objects of the same class touching each other because there could be objects which are similar to each other placed next to each other. So, U-Net also proposes a loss function which penalises pixels closer to edges more than pixels away from edges. Edges are extremely critical for medical image segmentation. Over the years, there have been several variants of U-Net, mostly by changing the architecture, by replacing the convolutional blocks with dense blocks and increasing the depth of the U-Net, so on and so forth.

(Refer Slide Time: 16:30)



### PSPNet: Pyramid Scene Parsing Network<sup>4</sup>

- **What challenges could arise when using FCN on complex scenes?**
  - FCN may not learn that some visual patterns are co-occurrent; E.g. cars are on roads, while boats are over rivers
  - FCN may predict parts of an object as different categories; E.g. parts of a skyscraper may be mis-classified as different buildings
  - FCN may fail to correctly segment small objects and big objects (relative to its receptive field)
- **Hypothesis:** Using global context information can lead to better segmentation; a network with suitable global scene-level information at different scales can improve segmentation

<sup>4</sup>Zhao et al, Pyramid Scene Parsing Network, CVPR 2017



Vineeth N B (IIT-H)

§7.3 CNNs for Segmentation

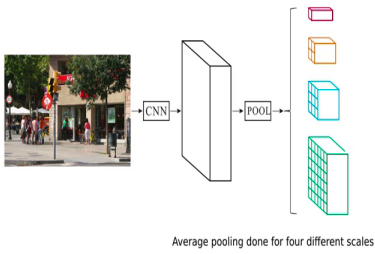
14 / 29

Another network used for segmentation is known as the PSPNet. And the PSPNet starts by asking the question, What challenges could you face if you applied FCNs on complex scenes? If you think deeper, you would notice that FCNs do not learn patterns that are co-occurrent. For example, we know that cars are on roads while boats are on rivers. And FCNs (may) can also predict parts of an object as different categories. For example, it can look at a skyscraper and think pixels belong to different buildings because of the sheer scale of that object.

Thirdly, FCNs could fail to detect large objects or small objects, depending on the architecture and the receptive field employed in the architecture. So, how can we address some of these constraints? PSPNet hypothesizes that one should use global context information to improve segmentation performance. So, how do you get global context information? Segment at multiple scales. So, when you segment at a lower resolution, you are actually trying to bring more global information into the segmentation result.

(Refer Slide Time: 18:15)

**PSPNet: Pyramid Pooling Module**



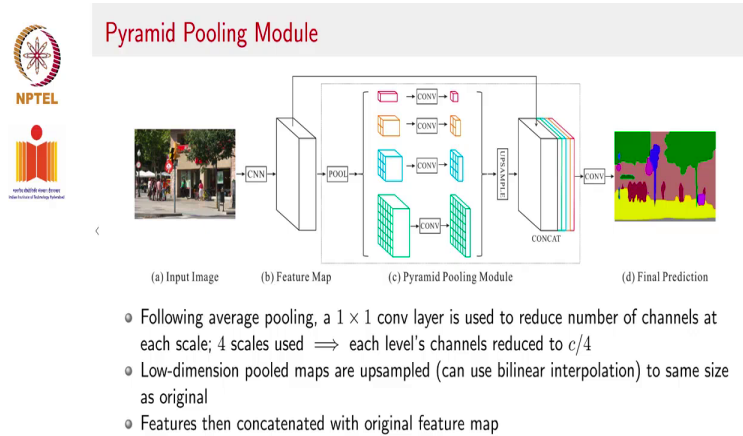
- Take final layer feature map of a deep network like ResNet
- Pyramid Pooling Module** combines features in four different scales:
  - Coarsest level is simply global average pooling
  - Each successive pooling level gives increased localization information
- Average pooled outputs are thus  $1 \times 1 \times c$ ,  $2 \times 2 \times c$ ,  $3 \times 3 \times c$ ,  $6 \times 6 \times c$  where  $c$  is number of input channels

Vineeth N B (IIT-H) 57.3 CNNs for Segmentation 15 / 29

So, PSPNet introduces a pyramid pooling module where, given an image, you forward propagate it through a CNN and then there is a pooling layer that pools at multiple scales. What are the different scales? There is a coarse scale which is simply a global average pooling. So, this will be  $1 \times 1 \times C$ , where  $C$  are the number of channels. And each successive pooling level gives increased localized information, localisation information. So, you would have your pooled outputs to be  $1 \times 1 \times C$ . That is the global average pooling. So, each of the  $C$  channels here in the convolutional output feature map, you would take that channel, global average pool and get one scalar.

And for each of these  $C$  channels you would get  $1 \times 1 \times C$ . Similarly  $2 \times 2 \times C$ ,  $3 \times 3 \times C$  and  $6 \times 6 \times C$ . And those are the 4 scales that PSPNet employs. Remember that when you do pooling, typically the number of channels do not reduce. You operate channel wise.

(Refer Slide Time: 19:39)



Vineeth N B (IIT-H)

97.3 CNNs for Segmentation

16 / 29

So, once these 4 scales pooling is done, the PSPNet now applies a convolutional  $1 \times 1$  convolution on each of these maps that came from the pooling operations to reduce the number of channels in the output. By how much does each of these  $1 \times 1$  convolutions reduce the channel size? By one fourth. There is a reason for that. The reason is, once these  $1 \times 1$  convolutions are performed, all of these are upsampled and concatenated, so each of these pooled maps are upsampled using simple bilinear interpolation and then they are concatenated. And this concatenated volume is going to have the same number of channels as the original feature map. Once this is done, this is then sent through a convolutional layer to get your final prediction of segmentation.

(Refer Slide Time: 20:46)

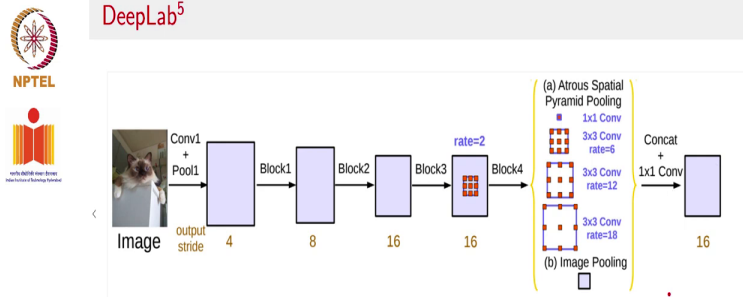


Figure 5. Parallel modules with atrous convolution (ASPP), augmented with image-level features.

Note that atrous convolution = dilated convolution = transpose convolution = fractionally strided convolution

<sup>5</sup>Chen et al, DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, TPAMI 2017



Vineeth N B (IIT-H)

§7.3 CNNs for Segmentation

17 / 29

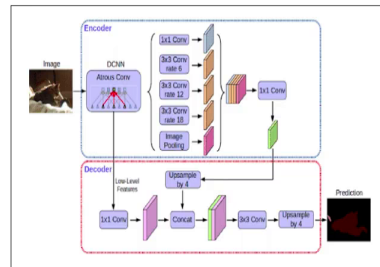
Another method is known as DeepLab which is a popular method for semantic segmentation which is similar to PSPNet. In this case, the image goes through several blocks of convolution. And then instead of having a pooling layer, which splits the image into 4 different scales, it performs Atrous Spatial Pyramid Pooling. So, this module is known as the ASPP module. Remember, Atrous convolution is the same as dilated convolution or transposed convolution, or sometimes even called fractionally strided convolution.

So, when your stride is less than one. So, then you get the same effect as performing pooling at multiple scales. So, Atrous spatial pyramid pooling, you could say, is a different way of implementing your pyramid pooling module in PSPNet. So, once you do your Atrous spatial pyramid pooling, you once again upsample, concatenate and you finally get your result on that concatenated volume.

(Refer Slide Time: 22:00)



## DeepLab V3<sup>6</sup>



Adds image-level features to ASPP, batch normalization for easier training, decoder module to refine segmentation results

Credit: Kevin Windholm, Novatec-GMBH

<sup>6</sup>Chen et al, Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation, ECCV 2018

Vineeth N B (IIT-H)

87.3 CNNs for Segmentation

18 / 29



DeepLab has gone through a few different versions over the years, and one of the more recent versions, known as DeepLab V3, adds a few modules to the basic DeepLab architecture. In this DeepLab V3 module, image level features are also passed on to the ASPP module. There is batch normalization which is used for easier training. And there is also an entire decoder architecture which is used to refine the segmentation results. So, you can see here that you have the ASPP module using which you perform  $1 \times 1$  convolution,  $3 \times 3$  convolution, so on and so forth.

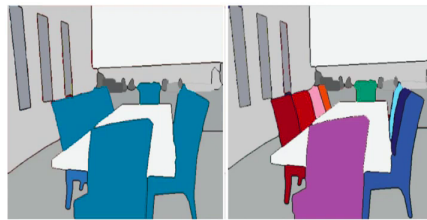
You get a set of feature maps which are concatenated.  $1 \times 1$  convolution is done to reduce the depth of those channels and that is then upsampled by 4 and passed on to the decoder along with the output of the ASPP module. These are then concatenated and you finally have one more convolution at the end and then upsample and make your predictions at the end. Remember that this predict, when we say prediction, it is going to be again an output volume, whose resolution is the same as the input image and the number of channels would be  $C+1$  where  $C$  is the number of classes, +1 for background.



(Refer Slide Time: 23:34)



### Instance Segmentation



Semantic Segmentation

Semantic Instance Segmentation

Credit: Soroush, StackOverflow



Vineeth N B (IIT-H)

§7.3 CNNs for Segmentation

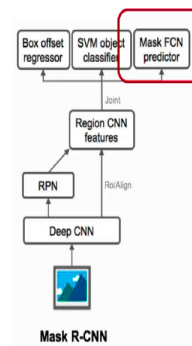
19 / 29

Moving on from semantic segmentation, instance segmentation is a slightly more challenging task than semantic segmentation.

(Refer Slide Time: 23:52)



### Mask R-CNN<sup>7</sup>



- Aims to tackle instance segmentation (where each pixel is given a class label, as well as an object ID)
- Mask R-CNN = Faster R-CNN with FCN on ROIs
- Adds a parallel head to predict masks along with existing branches for classification and localization

Credit: Lilian Weng, Github.io

<sup>7</sup>He, Mask R-CNN, ICCV 2017



Vineeth N B (IIT-H)

§7.3 CNNs for Segmentation



20 / 29

In semantic instance segmentation or what is popularly known as instant segmentation, Our goal is to not only give a class label for every pixel, but to also assign an object ID for each of the objects. So, if there are multiple people or multiple chairs or multiple dogs, you will want to also separate those people, the instances of the people and instances of the dog, which the basic

semantic segmentation task did not consider. In semantic segmentation, a dog would be a dog pixel whether there were 2 dogs or 3 dogs or 4 dogs, all of them would just be classified as dog. But in instance segmentation we want dog 1, dog 2, dog 3 and dog 4. And the challenge here is you may not know the number of dogs which is similar to the detection problem.


Which is why one of the most popular approaches, for instance segmentation known as Mask R-CNN actually tries to improve upon faster R-CNN to achieve instance segmentation. So, Mask R-CNN was proposed in ICCV of 2017. And it uses a faster R-CNN like architecture but adds a branch to also mask out and get instance level segmentation. So, you could look at mask R-CNN as faster R-CNN with FCN on the regions of interest, it adds a parallel head. Along with your SVM object classifier or simply your classification module and your bounding box offset regressor, you now add a mask predictor that is a third head.

(Refer Slide Time: 25:42)



### Mask R-CNN: RoIAlign

- An improvement of RoIPool operation (used in detection frameworks)
- There is loss of information due to quantization when moving from object proposals in image space to proposals in feature space; further loss in RoIPool operation. **Why does this matter?**
- This is significant because one pixel in feature space is equivalent to many pixels on image
- **RoIAlign** performs bilinear interpolation for the exact coordinate
- This preserves translation-equivariance of masks



Vineeth N B. (IIT-H) 87.3 CNNs for Segmentation 21 / 29

An important contribution in mask R-CNN to be able to achieve the mask accurately was a module known as RoIAlign. And since mask R-CNN operates in the framework of Faster R-CNN, the main limitation of Faster R-CNN that this module addresses is that when you map object proposals to feature space, recall that both in fast R-CNN and faster R-CNN. You had certain object proposals and you map them to a certain feature space.

And feature space could be of a different resolution than the resolution in which you obtained your object proposals. In both Fast R-CNN and faster R-CNN, we simply warp it to the size required and then bring your object proposals to that dimension. However, when you do something like this, you could face errors due to quantization. Why does this matter? This matters because one pixel in feature space could be equivalent to many pixels on an image because the feature space is smaller in later convolutional layers when compared to the initial input.

When you try to get a mask, these few pixels can bring a significant error in the final performance. So, how do we overcome this problem? RoI align performs bilinear interpolation to get the exact coordinate for the locations where the object proposals match to a given feature map. This preserves translation equivariance of masks.

(Refer Slide Time: 27:48)

Mask R-CNN: RoIAlign

target masks on Rols

Translation of object in Rol => Same translation of mask in Rol

Credit: Kaiming he, ICCV 2017 Tutorial

Vineeth N B. (IIT-H) §7.3 CNNs for Segmentation 22 / 29

What do we mean? We mean that given an image where an object could be located in this yellow box or red box, both could be valid detections. But the mask position in each of these bounding boxes is different. So, by translation equivariance, we mean that if the object has moved by a certain amount inside the bounding box, the mask also has to move by certain, by the same amount. And this can be achieved by the RoI align layer.

(Refer Slide Time: 28:23)

**Mask R-CNN: RoIAlign**

Credit: *Kaiming he, ICCV 2017 Tutorial*

Vineeth N B (IIT-H) 57.3 CNNs for Segmentation 22 / 29

So, the way RoI align does it is, when you convert your object proposals to locations on your feature map, if the location turns out to be in between a few pixels on your feature map, then you perform bilinear interpolation of the pixels around the feature map and then you get your specific pixel value at that specific point pointed by the object proposal. And that is what is then pooled to get your fixed dimensional representation that goes to later layers of your faster R-CNN architecture.

(Refer Slide Time: 29:08)

**Mask R-CNN: FCN Mask Head**

- FCN branch generates mask for each proposal
- Mask is  $28 \times 28$  in size during training
- Rescaled to bounding box size and overlaid on image during inference

Credit: *Kaiming he, ICCV 2017 Tutorial*

Vineeth N B (IIT-H) 57.3 CNNs for Segmentation 23 / 29

So, in addition to your standard faster R-CNN architecture, with the classification loss and a bounding box loss, we also have a mask head with the mask R-CNN. And what the mask R-CNN does is, it uses an FCN kind of a branch. So, which means it is fully convolutional. And for each RoI the mask size is 28 cross 28, as you can see here. And it is finally rescaled to the bounding box size and then overlaid on the image during inference. So, this mask as you can see here is obtained for every RoI and then finally overlaid back on input image at inference time.

(Refer Slide Time: 29:55)

The slide is titled "Panoptic Segmentation" and features three side-by-side images illustrating different segmentation techniques. The first image, labeled "Semantic Segmentation", shows a street scene with various objects colored in different shades of purple and blue. The second image, labeled "Instance segmentation", shows the same scene with individual objects highlighted in distinct colors and outlined. The third image, labeled "Panoptic segmentation", combines the elements of the first two, showing a fully segmented scene where each pixel is assigned a unique color representing a specific object instance. The slide also includes the NPTEL logo, a credit to Harshit Kumar on Github, and a footer with a speaker's photo, the text "Kirillov et al, Panoptic Segmentation, CVPR 2019", "Vineeth N B (IIT-H)", "57.3 CNNs for Segmentation", and "24 / 29".

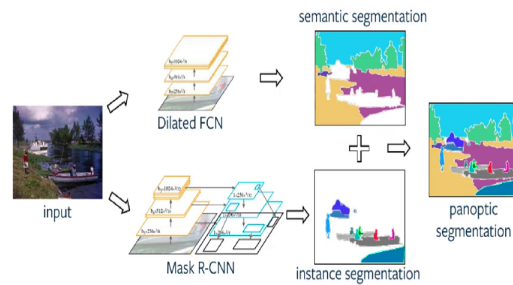
A last and more recent form of segmentation that is popular is known as panoptic segmentation. Panoptic segmentation combines semantic segmentation and instance segmentation. So, semantic segmentation, as you can see, is where you segment the pixels of all objects. But all instances remain the same. Instance segmentation is where you only care about the instances of the objects in your categories, but you label each instance separately.

And panoptic segmentation is the union of the two where you label every pixel as belonging to different objects. And you also ensure that each instance of an object gets a different label altogether.

(Refer Slide Time: 30:52)



Panoptic Segmentation: Possible Architectures



Credit: Harshit Kumar, Github.io



Vineeth N B (IIT-H)

§7.3 CNNs for Segmentation

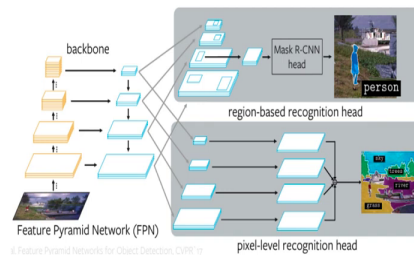
25 / 29

So, how do you perform panoptic segmentation? So, this work was first introduced in CVPR of 2019, about a year old. And one could achieve panoptic segmentations using a couple of architectures. So you could perform Mask R-CNN to get instance segmentation results. You could do a dilated FCN to get semantic segmentation results and then you merge the two to get panoptic segmentation. That is one approach. So, this particular work actually implements a couple of approaches and then studies how this needs to be evaluated differently.

(Refer Slide Time: 31:31)



Panoptic Segmentation: Possible Architectures



Loss Function?  $L = \lambda_i(L_{cls} + L_{bbox} + L_{mask}) + \lambda_s L_s$ , where:

- Instance segmentation branch has  $L_{cls}$  = classification loss;  $L_{bbox}$  = bounding-box loss;  $L_{mask}$  = mask loss
- Semantic segmentation branch has  $L_s$  = pixel-wise cross-entropy loss



Vineeth N B (IIT-H)


§7.3 CNNs for Segmentation

26 / 29

So, another approach is you could take a feature pyramid network instead of what we saw here. Instead of using a dilated FCN or a mask R-CNN, You could use an FPN, a feature pyramid network and then send those detections through a mask R-CNN head to get your instance level segmentations. And you could also send those different feature maps through a pixel level recognition head to your semantic segmentation. So, in both these approaches, there is an instance segmentation pathway and there is a semantic segmentation pathway which are combined to give your panoptic segmentation. What would be the loss function here? The loss function recommended is a combination of both semantic segmentation and (panoptic) and instance segmentation.

Remember that instance segmentation the way we saw with mask R-CNN has three losses classification, bounding box and the mask loss. And then for semantic segmentation, you have your standard pixel wise cross entropy loss.

(Refer Slide Time: 32:46)



### Panoptic Segmentation: New Evaluation Metric

Why do we need a new evaluation metric?

- In semantic segmentation, we use **IoU** and **per-pixel accuracy**
- In instance segmentation, we use **average precision over different IoU thresholds**
- Why can't we use these for panoptic segmentation?  
**Can cause asymmetry for classes with or without instance-level annotations**

Credit: Harshit Kumar, Github.io




One of the challenges of panoptic segmentation is that it cannot be evaluated using the metrics one may use for semantic segmentation or instance segmentation. Why is this so? In semantic segmentation, we use IoU and per pixel accuracy. So, you can try to see if you do a set of pixels that belong to a particular object, you could try to get the intersection over union with the ground truth of that object. You could also get a per pixel accuracy because you are going to predict

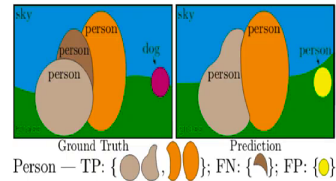
probabilities at each pixel level for all the classes that you have. You can also look at that accuracy as a metric.

On the other hand, for instance segmentation, which is similar to detection, you would have average precision over different IoU thresholds. Why cannot we use these for panoptic segmentation? That is because this could cause asymmetry for classes with or without instance level annotations. Remember that in panoptic segmentation you could have certain classes, where there are multiple instances. For example, people or dog, and there could be certain classes such as sky, road which may not have many instances. So, to combine these metrics without considering these asymmetries may not be wise.

(Refer Slide Time: 34:28)



### Panoptic Segmentation: PQ Metric



Person — TP: { }; FN: { }; FP: { }

For a ground truth segment  $g$ , and for predicted segment  $p$ , PQ is calculated as:

$$PQ = \frac{\sum_{(p,g) \in TP} \text{IoU}(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} = \underbrace{\frac{\sum_{(p,g) \in TP} \text{IoU}(p,g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition quality (RQ)}}$$

Credit: Harshit Kumar, Github.io
Vineeth N B (IIT-H)
§7.3 CNNs for Segmentation
28 / 29

So, this particular work recommends a new metric known as the PQ metric. How does the PQ metric work, you can look at this example here. Let us assume that this is your correct ground truth. So, there is sky, there is grass, there are three people. That is the instance segmentation part of it and then there is a dog. Let us assume now that whatever module we came up with predicted it this way.

The sky, it got it right. The grass, it got it right. The dog became person. And instead of three people, it only predicted two people where it combined two of these people. So, how do you now measure error of this kind of a system? So, the way this approach recommends is, you first write



out all your true positives, which would be your two people, which are true positives between your ground truth and prediction. There is a false negative, which is the person you missed and there is a false positive, which is the person you added instead of a dog.

So, for a ground truth segment  $g$  and for a predicted segment  $p$ , the P Q metric is computed as

$$\frac{\sum_{(p,g) \in TP} IoU(p,g)}{|TP|+0.5|FP|+0.5|FN|} = \frac{\sum_{(p,g) \in TP} IoU(p,g)}{|TP|} \times \frac{|TP|}{|TP|+0.5|FP|+0.5|FN|}$$

So, the first term here gives the segmentation quality. Among the true positives, What was the IOU for all of your segments? And the second part gives the recognition quality as to among all of your possible segments, how many of them did we get as true positive? So, you have a part of it that checks for segmentation quality and a part of it that checks for recognition quality and that is why this metric becomes useful for panoptic segmentation.

(Refer Slide Time: 37:07)

The screenshot shows a presentation slide with the following content:

- NPTEL** logo on the left.
- Homework** section header.
- Readings** section containing:
  - [Semantic Segmentation Overview by Nanonets](#)
  - [Overview of Deep Lab, AnalyticsVidhya](#)
  - [Introduction to Panoptic Segmentation](#)
- Footer: **Vineeth N B (IIT-H)**, **§7.3 CNNs for Segmentation**, and **29 / 29**.

For more details, please read a good overview of semantic segmentation by Nanonets, a good overview of Deep lab semantic segmentation method, which is a popular one at analytics Vidhya and also a good introduction to panoptic segmentation.

