

دانشگاه

خواجه نصیرالدین طوسی

K. N. Toosi University
of Technology



Computer Vision

Lecture 15: Image Segmentation

Dr. Esmaeil Najafi

MSc. Javad Khoramdel

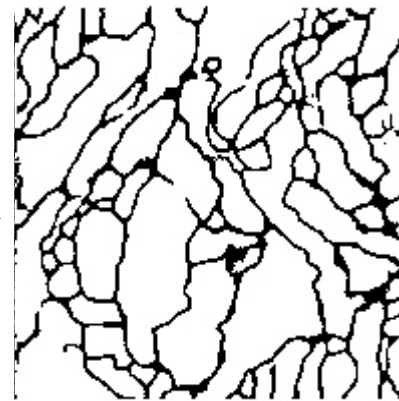
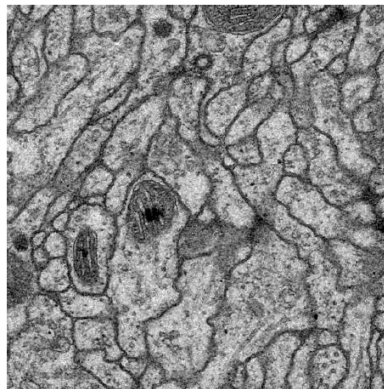


Image segmentation

- In computer vision, image segmentation is breaking down the image into subgroups called “image segments”.
- The areas in a subgroup are similar to each other and dissimilar to the areas from other segments.
- Image segmentation has lots of applications specifically in medical image processing.



(SEMANTIC IMAGE SEGMENTATION WITH DEEP CONVOLUTIONALNETS AND FULLY CONNECTED CRFS, Liang-Chieh Chen et al., (2016))



(Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images, Dan C. Cireşan et al., (2012))

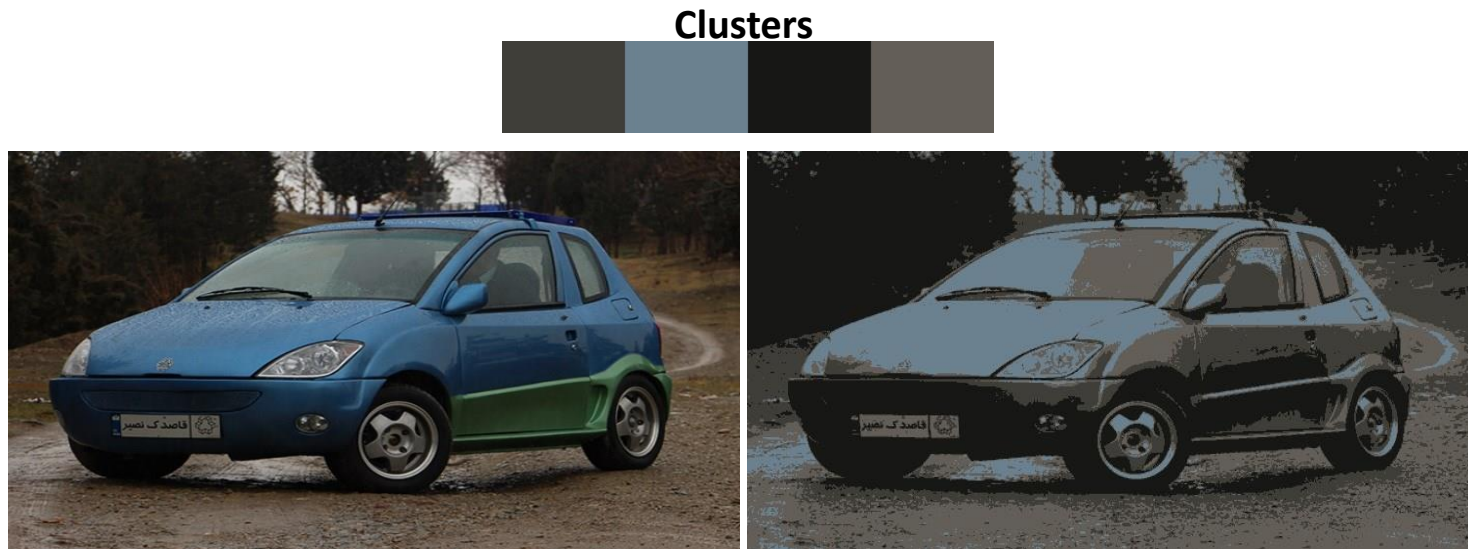
Image segmentation

- Image segmentation can be discussed from different points of views.
- If someone ask you to segment the image below into four subgroups, someone might segment the image by colors: white, brown, green, and blue.
- Another person might segment the image by the objects and things in the image: Car, trees, ground, and sky.
 - This kind of segmentation is called “**semantic segmentation**”.



Image segmentation

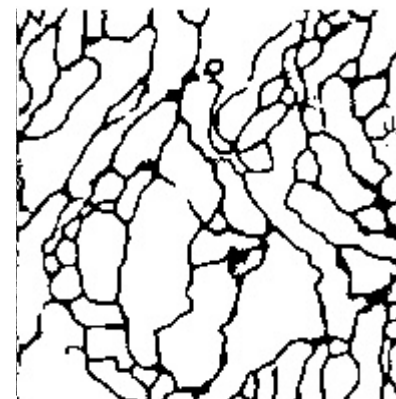
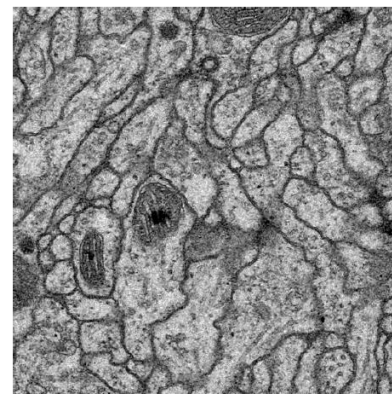
- One way to solve the image segmentation is using the unsupervised algorithms.
 - K means for image segmentation



Note that since this method is unsupervised, it is not aware of the context. K means has put the sky, parts of the ground and car hood in a same group (they have similar colors in the segmented image).

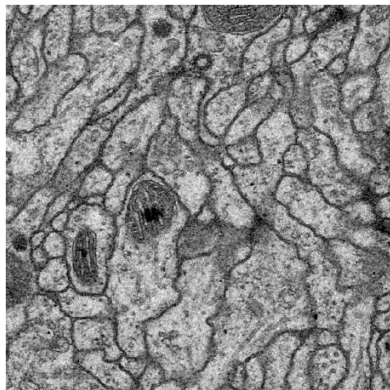
Problem definition

- Let's say given a set of electron microscopy (EM) images of brain, and we want to segment the neural membranes.
- This would help the neurologist study neural structure of the brain.
- Each pixel on the top image is either neural membrane or not (background).
- The bottom image is what we are looking for. The neural membranes are showed in black, the other stuff in white.
- The goal is to develop an algorithm for automated segmentation.

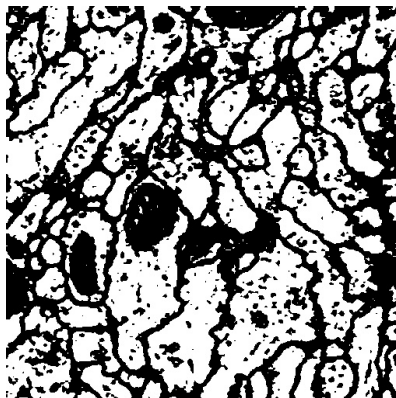


Unsupervised approach: K-means

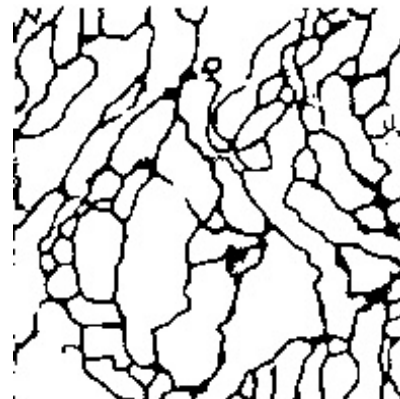
- Since we want membrane and not membrane segments, we might set $k=2$ in K-means.



Input image



Predicted



Ground Truth

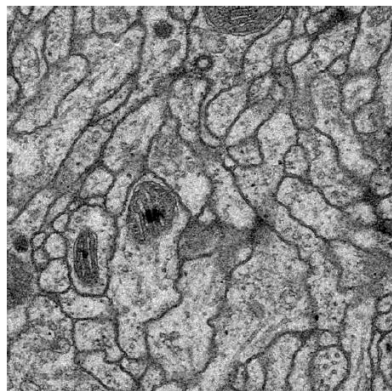
- As you can see, the prediction result is not good enough.

Image segmentation

- To be fair, there are unsupervised segmentation methods that are more accurate than K-means (like normalized cut, watershed, ...) but we will focus on supervised segmentation for couple of reasons:
 - Although unsupervised algorithms for image segmentation are helpful when we do not have labels, studies show that supervised algorithms are more accurate (Hoover et al., 1996, Wanqing et al., 2004)
 - It would be hard (not impossible) to do semantic segmentation unsupervisedly.
 - In unsupervised learning, we have much less control over the problem and how the algorithm solves it.

Supervised image segmentation

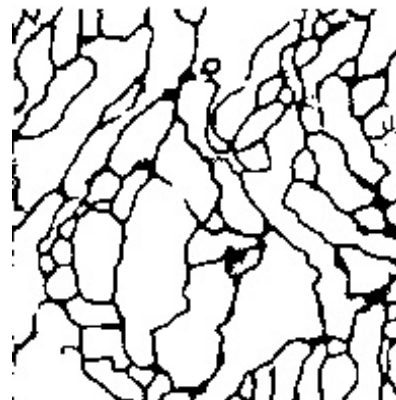
- For every pixel in the image, it either represents a neural membrane or not.
- Semantic segmentation is like image classification, except that in image classification, we assign one label for the whole image, but in segmentation, each pixel has its label.



Input image

```
[[0, 0, 0, ..., 0, 0, 0],  
 [0, 0, 0, ..., 0, 0, 0],  
 [0, 0, 0, ..., 0, 0, 0],  
 ...,  
 [1, 0, 0, ..., 0, 0, 0],  
 [0, 0, 0, ..., 0, 0, 0],  
 [0, 0, 0, ..., 0, 0, 0]]
```

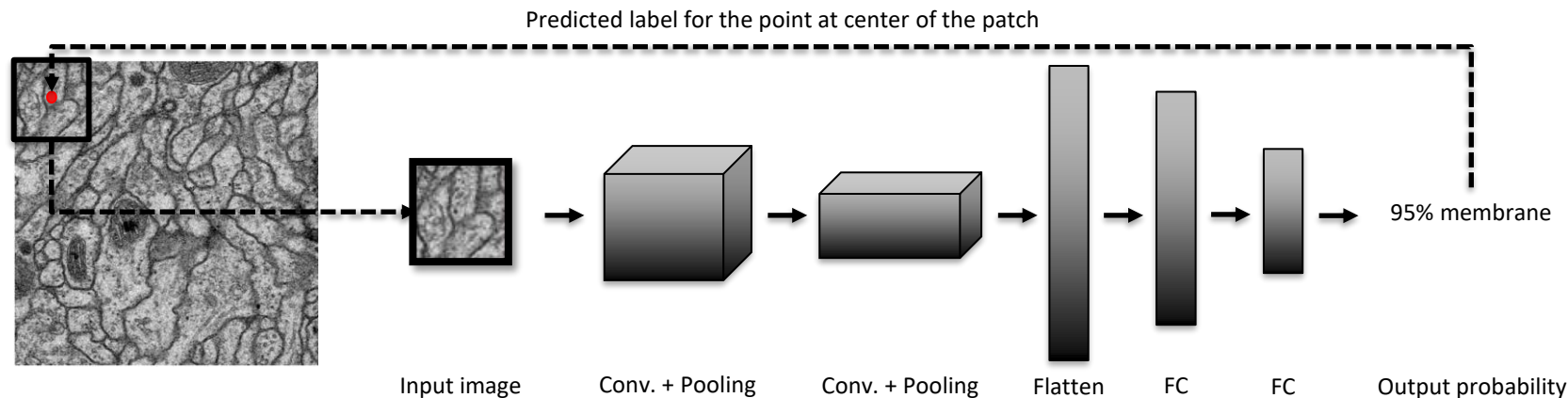
Labels



Visualized labels

Sliding window for segmentation

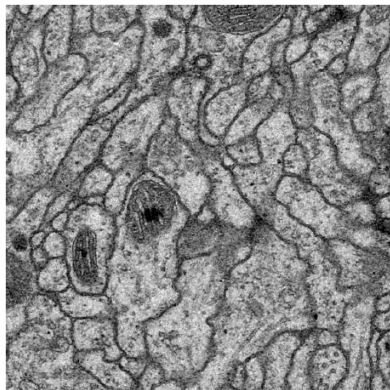
- Pick a fixed size sliding window.
- Extract the patches using that window.
- For all the patches:
 - let a classification network identify the class of the patch.
 - Assign the predicted label to the pixel at the center of the patch.
 - Move to the next patch



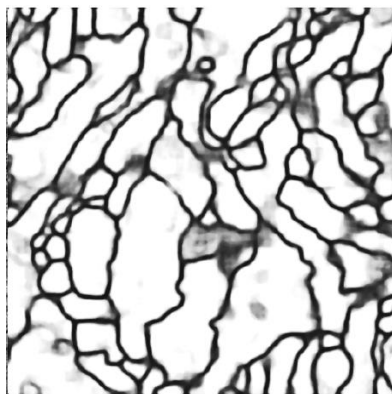
(Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images, Dan C. Ciresan et al., (2012))

Sliding window for segmentation in practice

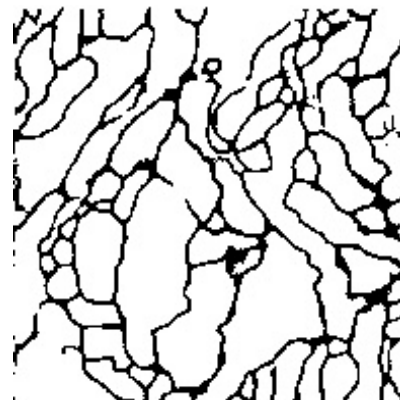
- Much better!



Input image



Predicted

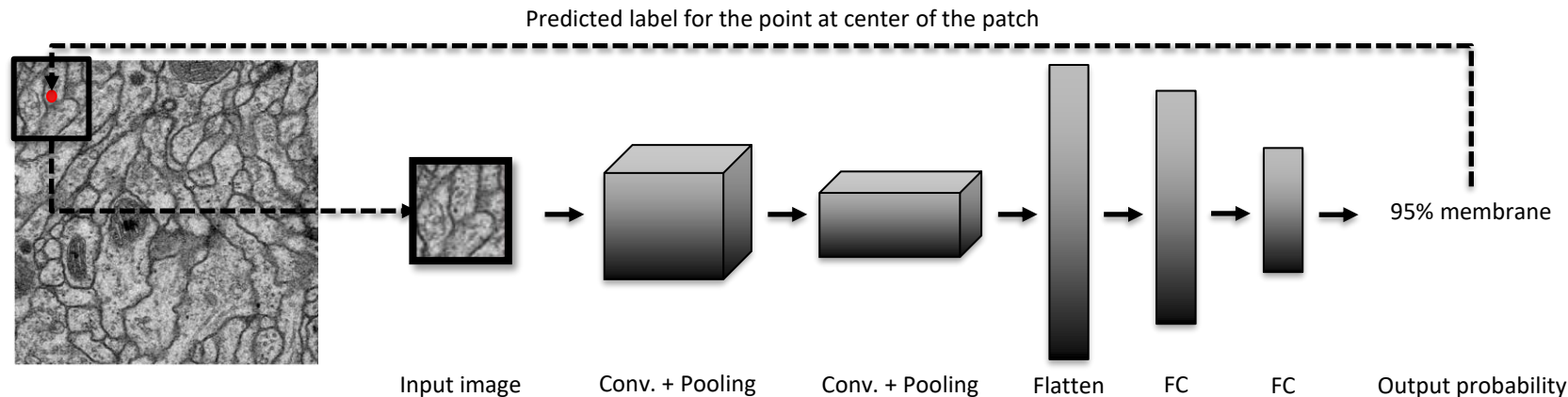


Ground Truth

(Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images, Dan C. Ciresan et al., (2012))

Sliding window for segmentation

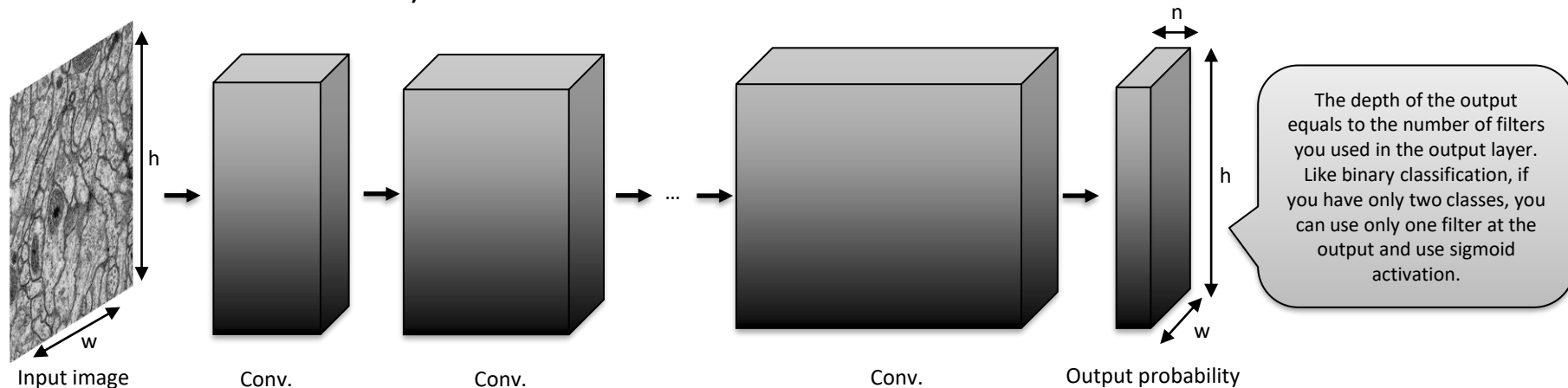
- This idea was proposed by Dan C. Ciresan et al. in 2012
- Although this concept works, it is highly computationally expensive.
 - Imagine that the input image is 512 by 512, so there would be 2^{18} pixels in the picture, and the network must predict 2^{18} patches to segment a single photo!
- As a consequence, this concept can not scale.



(Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images, Dan C. Ciresan et al., (2012))

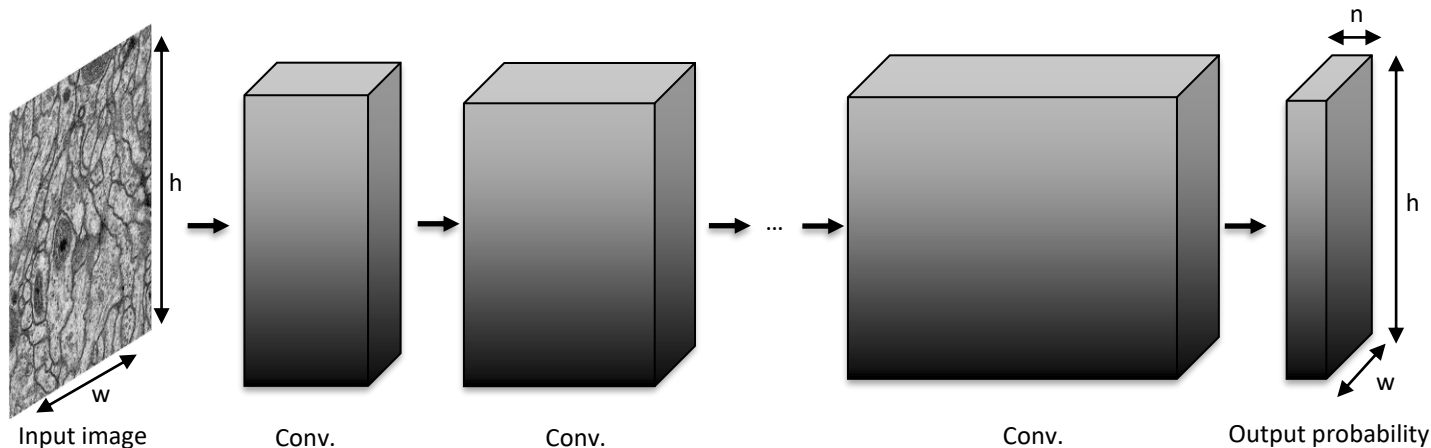
Possible solution: Preserve the resolution

- Use enough “padding” to preserve spatial resolution in feature maps. Do not use pooling layers.
 - Hence, if the input image resolution is $h \times w$, the resolution of the feature maps would be $h \times w$ all over the network.
- Set the number of filters in the output layer the same as the number of classes (n).
 - For each pixel, we would have a $n \times 1$ vector at the final output (The output of the network would be $h \times w \times n$)



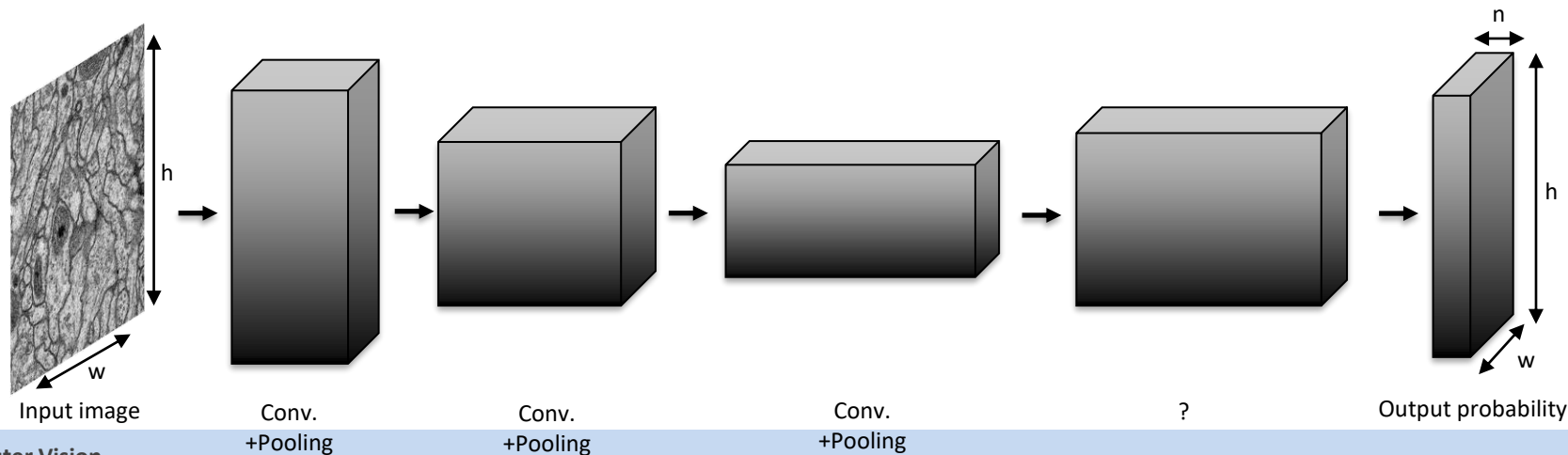
Possible solution: Preserve the resolution

- Preserving spatial resolution is computationally expensive.
 - It requires lots of RAM to train a deep neural network without down-sampling.
- On the other hand, the output of the network should have same spatial resolution as the input image.



Possible solution: Preserve the resolution

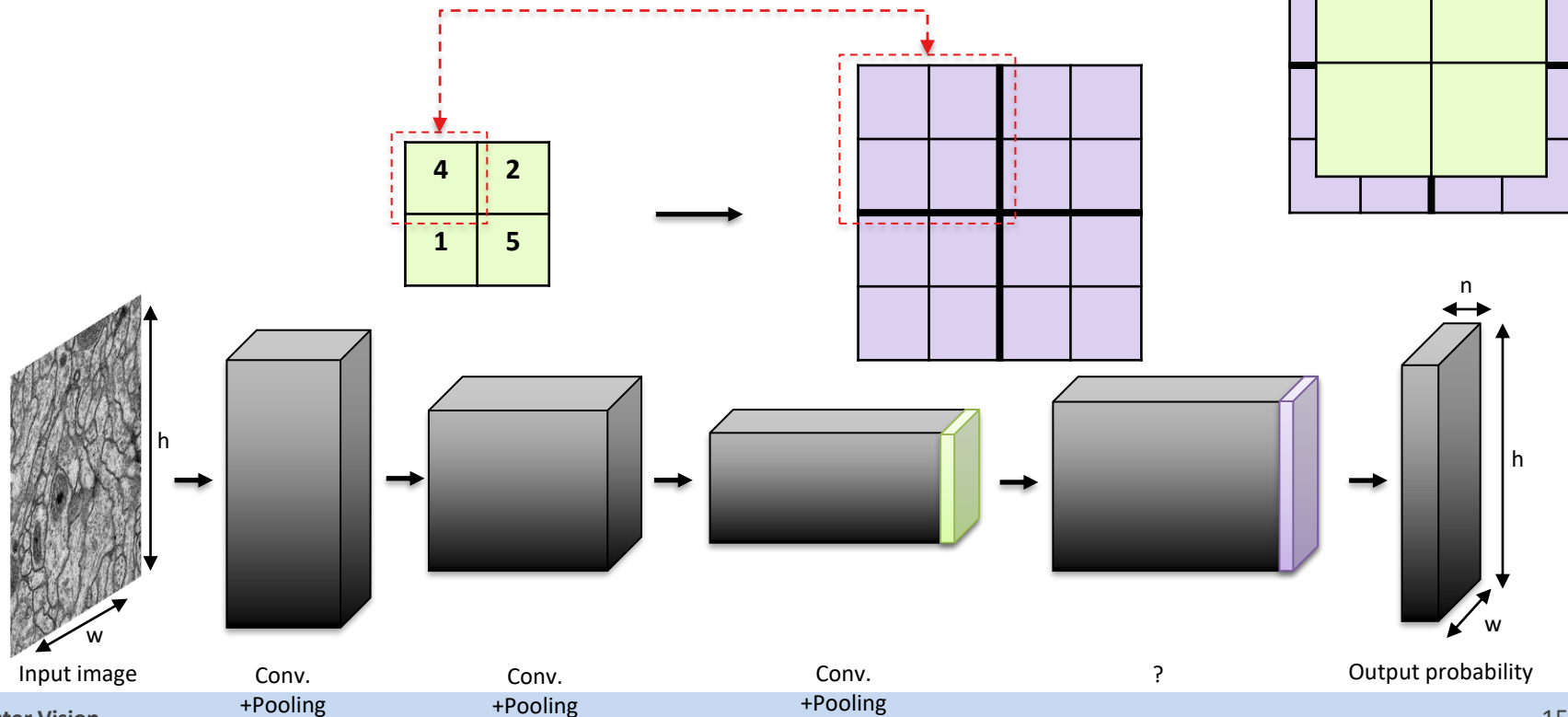
- So, maybe we would better down-sample at first, then up-sample.
- We already know how to down-sample, but how would we up-sample?



Up-sampling: Nearest neighbor

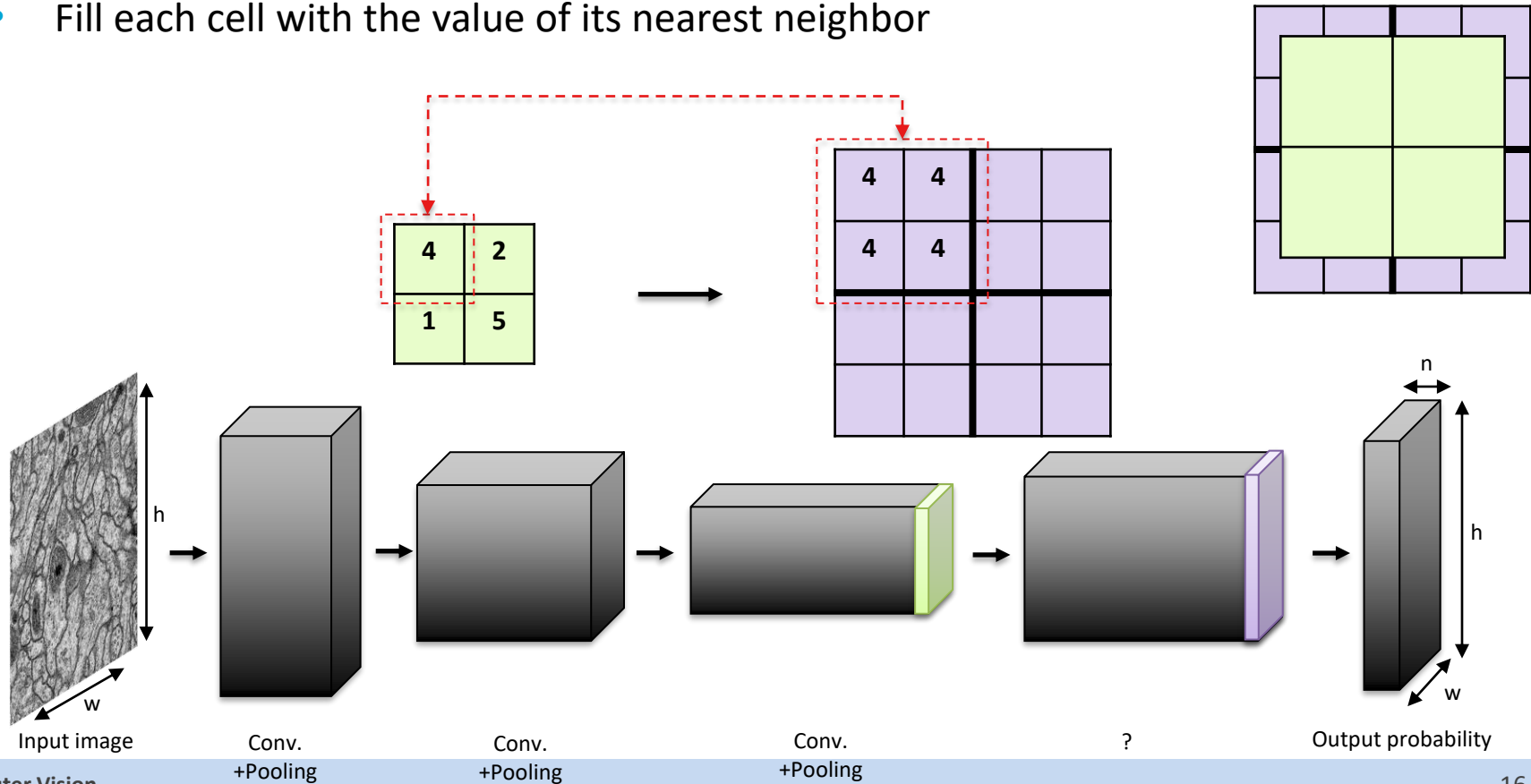
- Find the nearest neighbor of the cells in the up-sampled area.

This picture shows how we would pick up the nearest neighbors



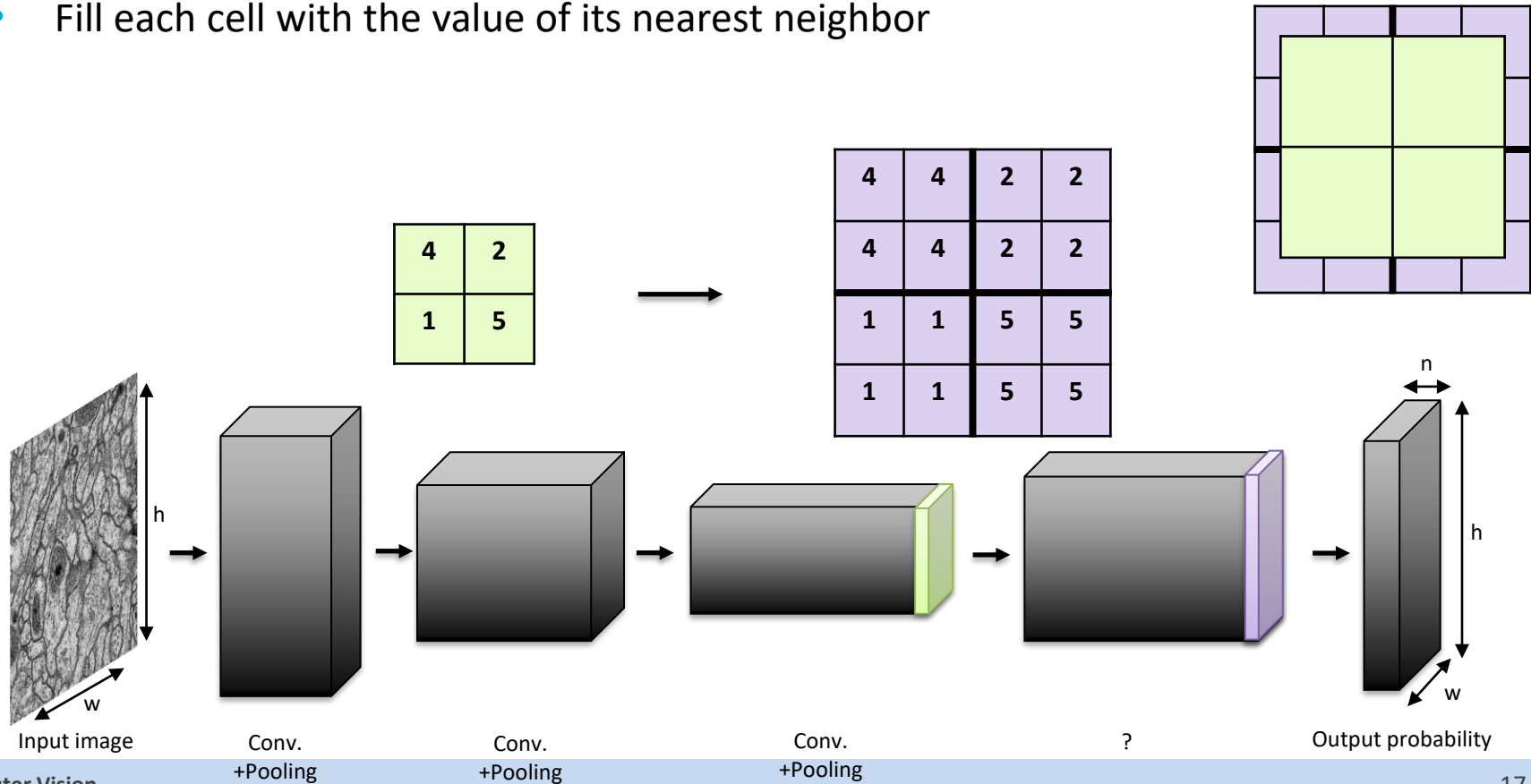
Up-sampling: Nearest neighbor

- Fill each cell with the value of its nearest neighbor



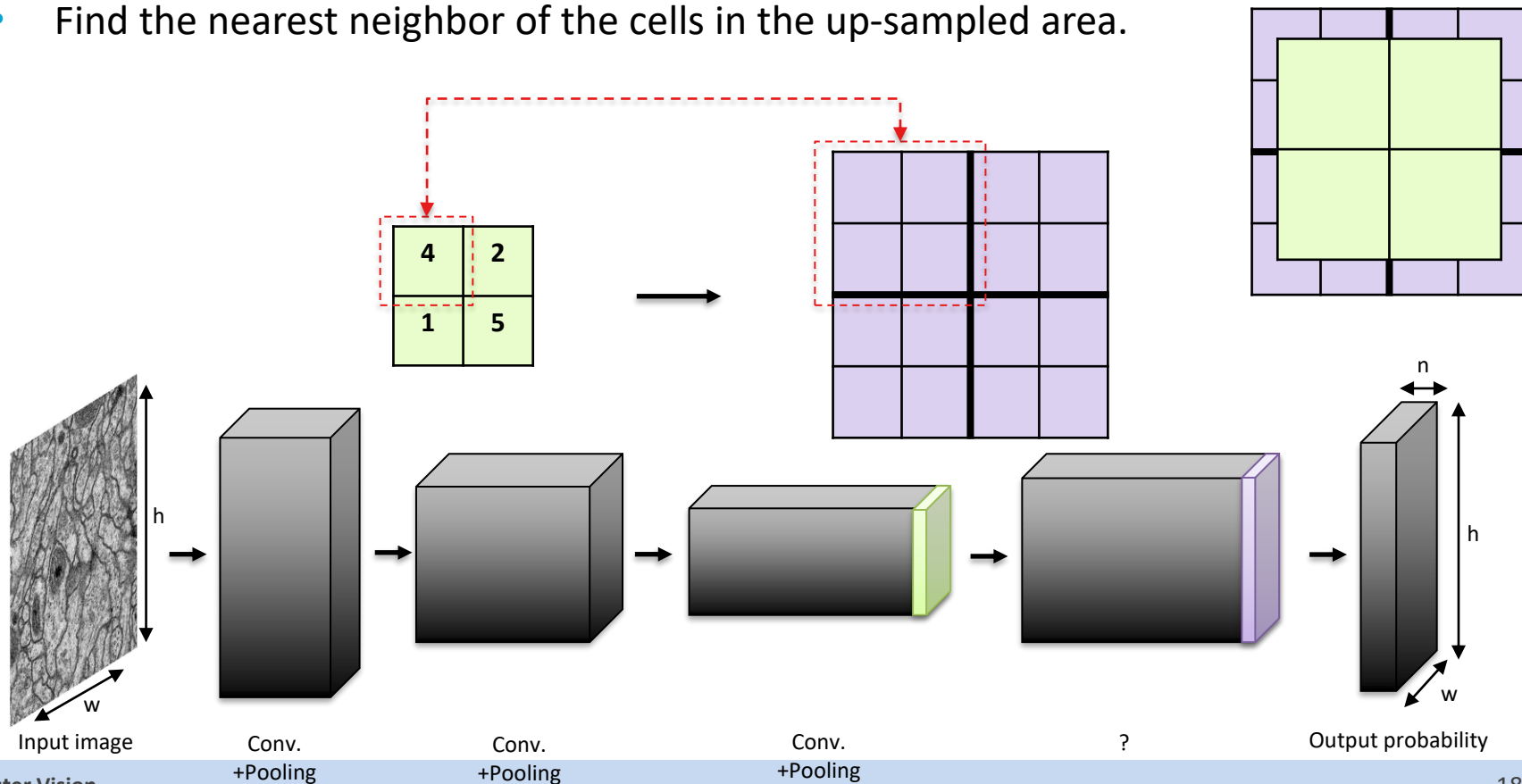
Up-sampling: Nearest neighbor

- Fill each cell with the value of its nearest neighbor



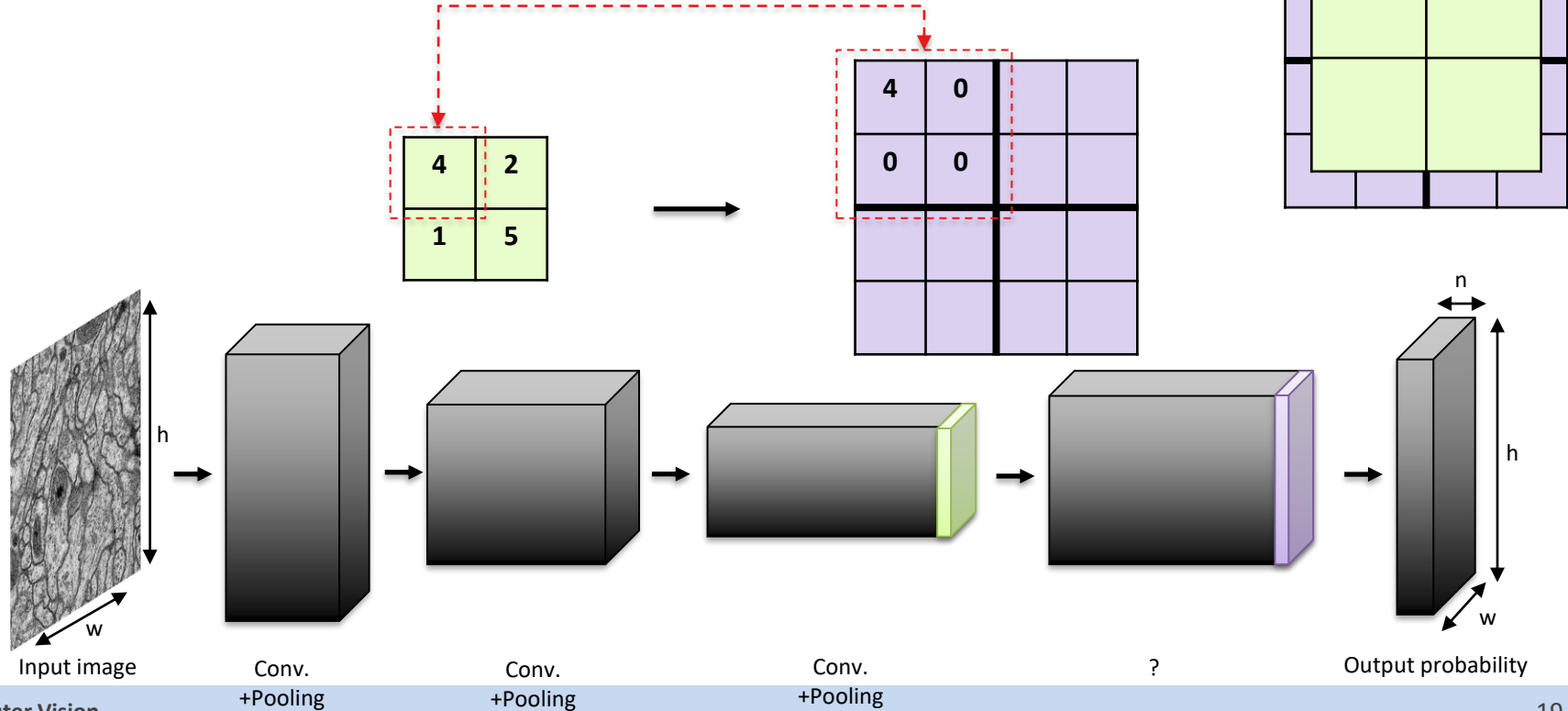
Up-sampling: Bed of nails

- Find the nearest neighbor of the cells in the up-sampled area.



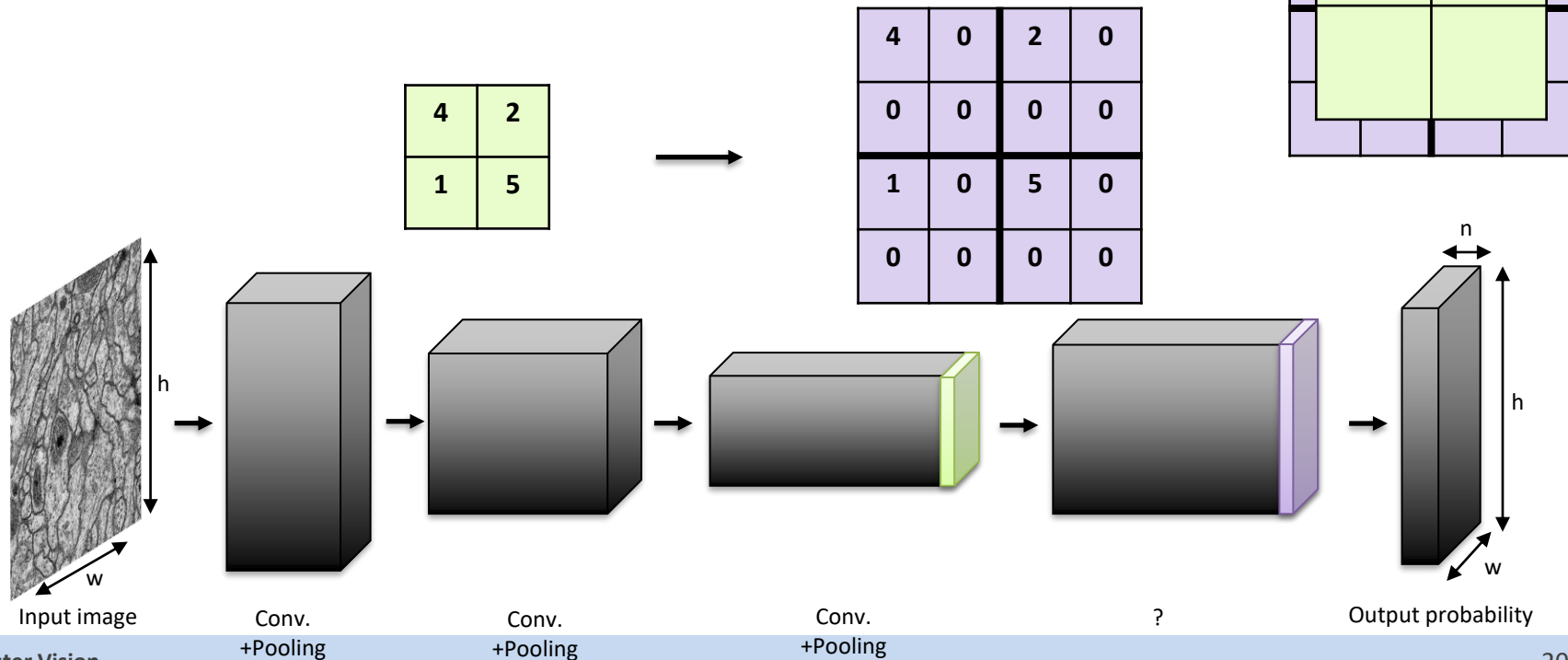
Up-sampling: Bed of nails

- Fill top left cell of each section with the value of its nearest neighbor, set the other cell values to zero.



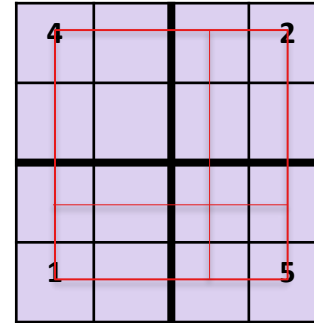
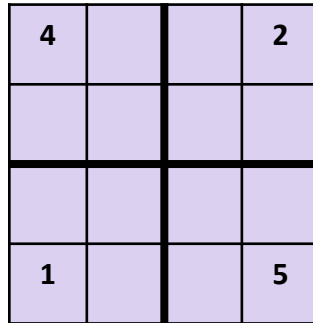
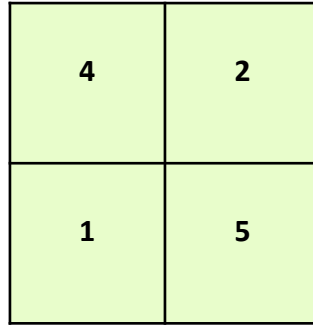
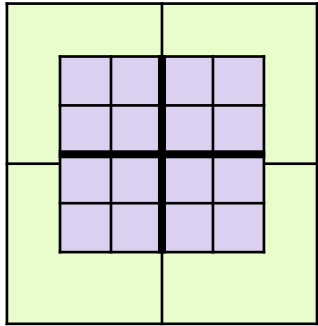
Up-sampling: Bed of nails

- Fill top left cell of each section with the value of its nearest neighbor, set the other cell values to zero.



Up-sampling: Bilinear interpolation

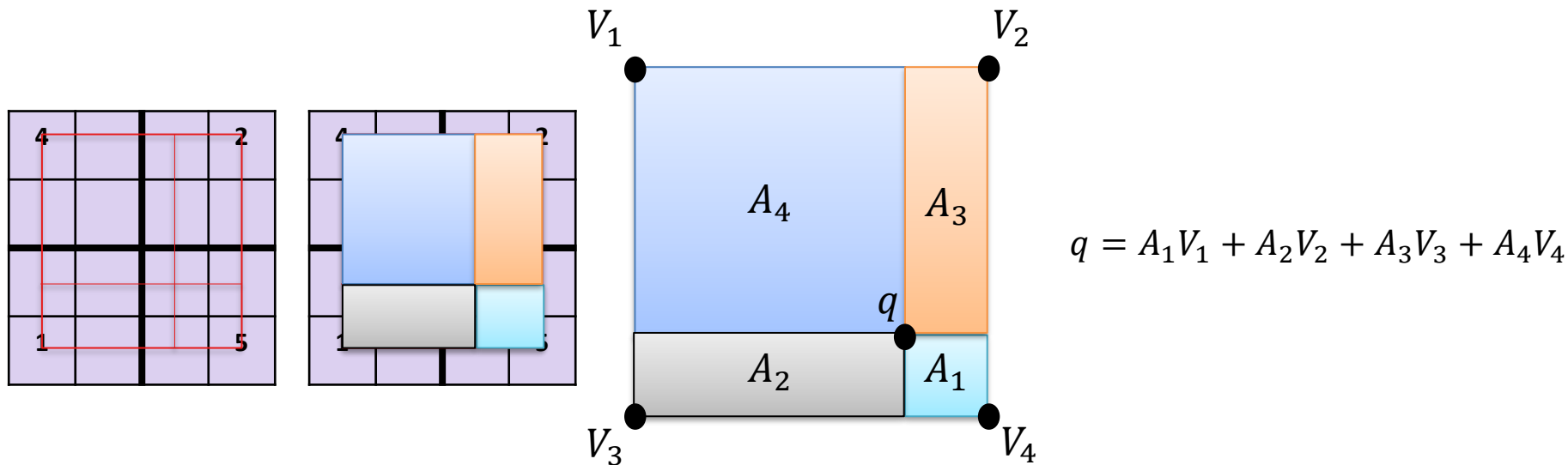
- Up-sampling can be done with bi-linear interpolation



The empty cells need to be interpolated.

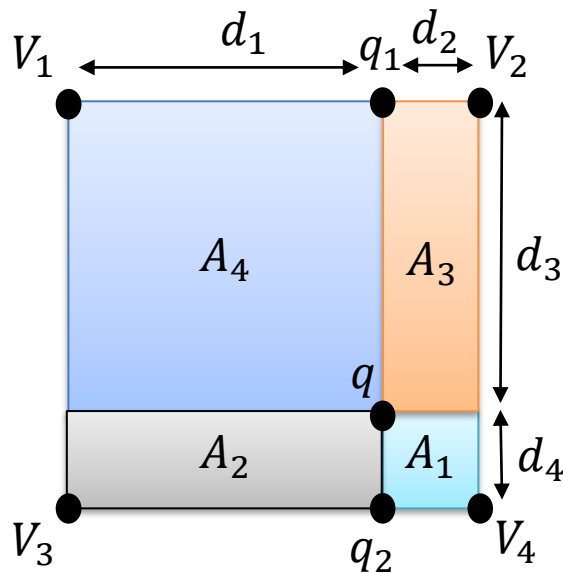
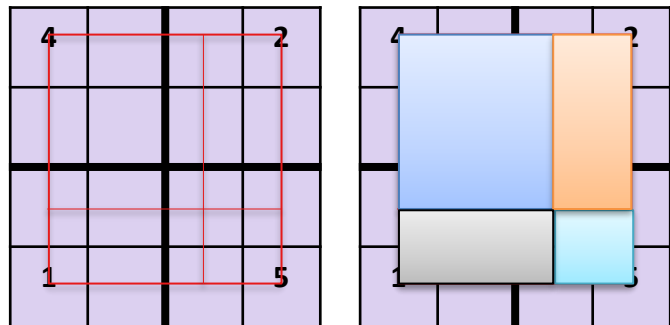
Up-sampling: Bilinear interpolation

- Bi-linear interpolation can be done by using the normalized areas of the corresponding squares.



Up-sampling: Bilinear interpolation

- Or alternatively, we can use the distances for bilinear interpolation



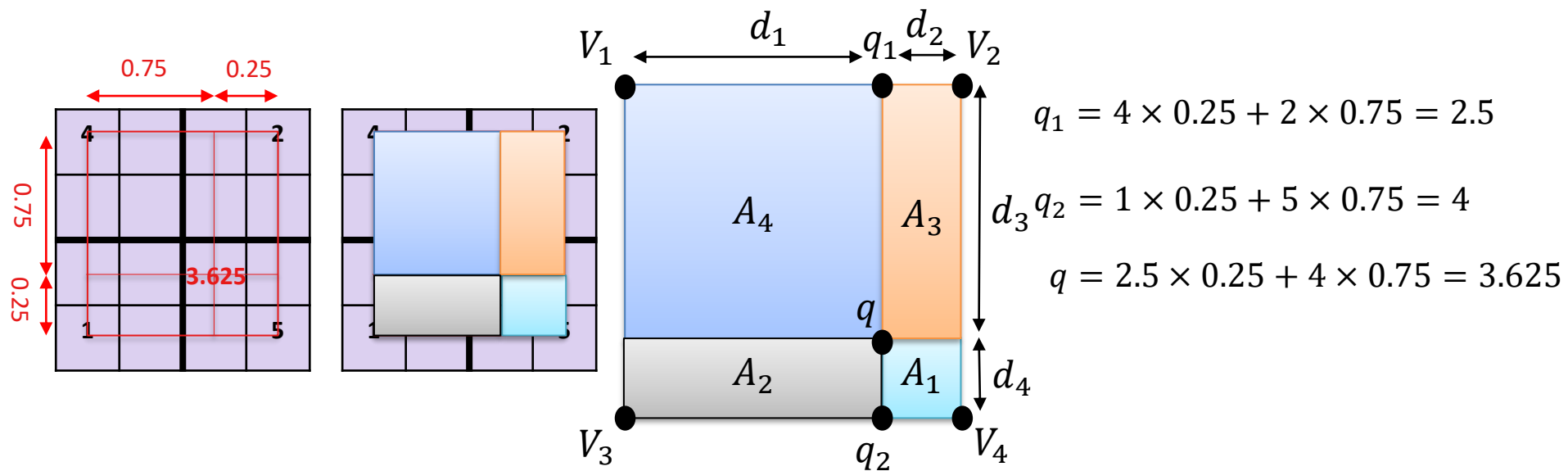
$$q_1 = V_1 d_2 + V_2 d_1$$

$$q_2 = V_3 d_2 + V_4 d_1$$

$$q = q_1 d_4 + q_2 d_3$$

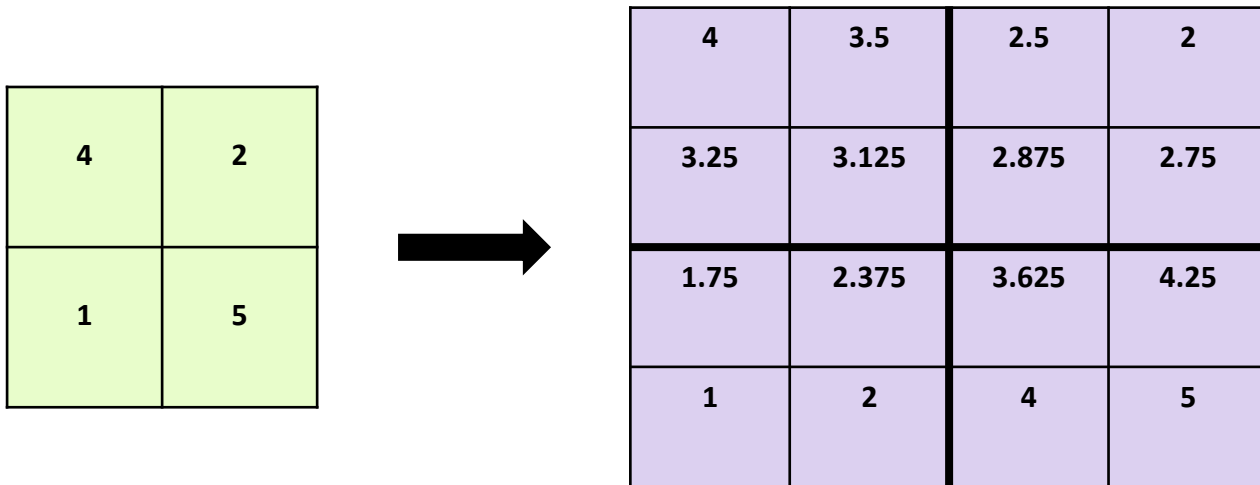
Up-sampling: Bilinear interpolation

- Or alternatively, we can use the distances for bilinear interpolation



Up-sampling: Bilinear interpolation

- Or alternatively, we can use the distances for bilinear interpolation



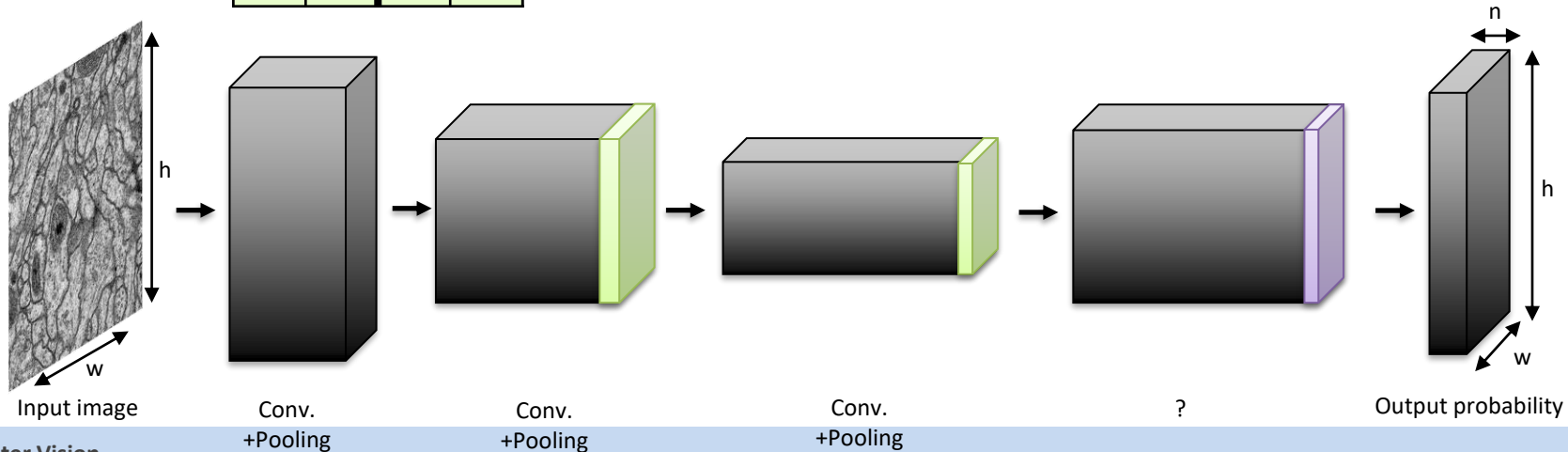
Up-sampling: Max-unpooling

- Down-sampling is usually done by Max-pooling

3	8	1	5
7	5	6	0
7	0	4	2
8	9	6	4

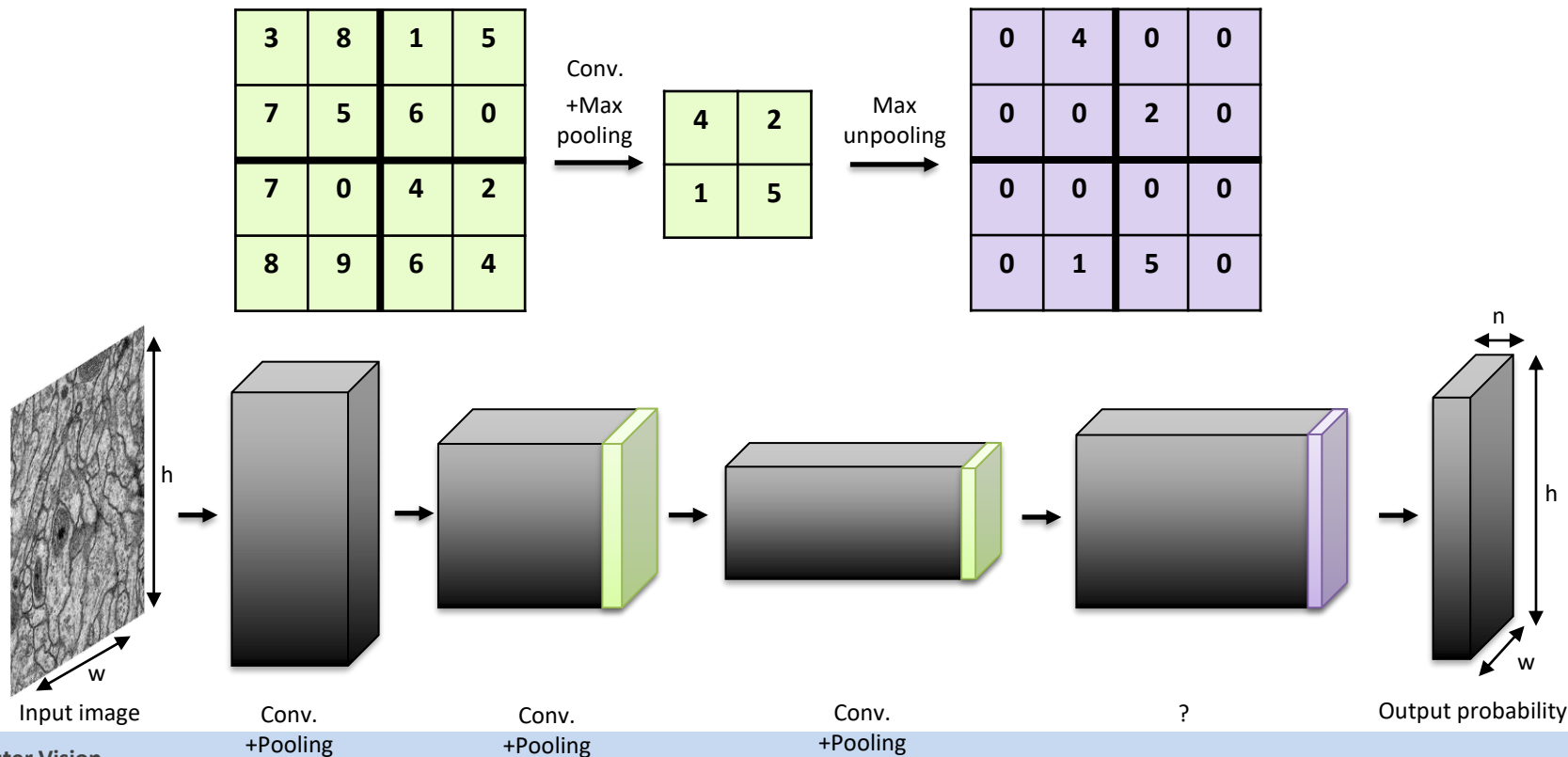
Conv.
+Max
pooling

4	2
1	5



Up-sampling: Max-unpooling

- By remembering the location of maximums, we can do up-sampling.



Up-sampling: Transpose Convolution

- Same as the Unpooling, which undoes the effect of pooling, we can come up with something to undo the convolution.
- This operation is called Transpose convolution.

2	8	1	5
7	2	6	1
1	0	3	2
8	0	6	2

1	0
0	1



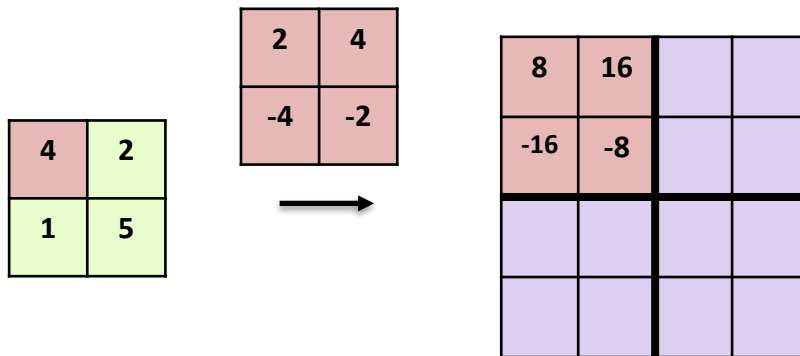
4	2
1	5

+Max
pooling

Conv.

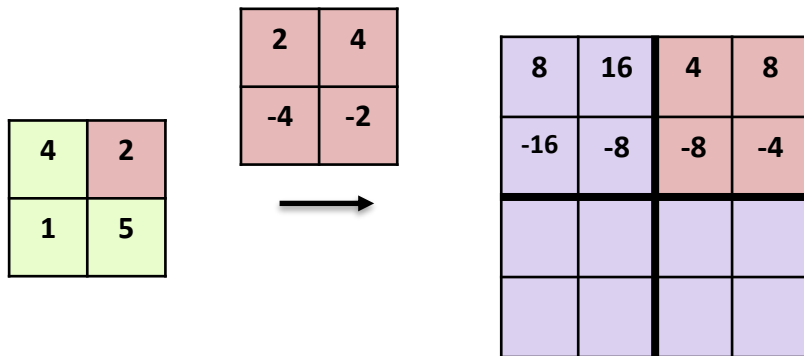
Up-sampling: Transpose Convolution

- Transpose convolution with:
 - 2 by 2 filter
 - Stride 2



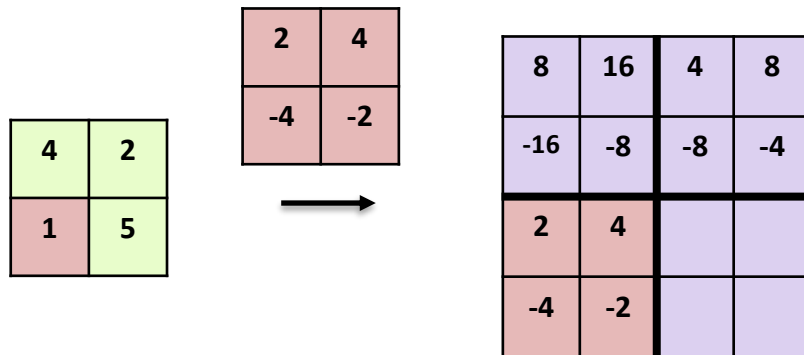
Up-sampling: Transpose Convolution

- Transpose convolution with:
 - 2 by 2 filter
 - Stride 2



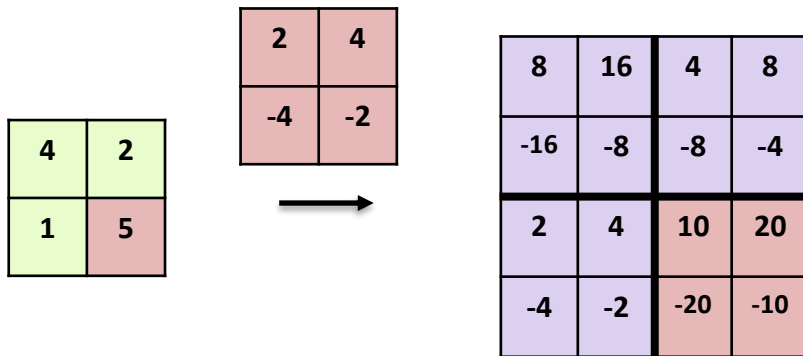
Up-sampling: Transpose Convolution

- Transpose convolution with:
 - 2 by 2 filter
 - Stride 2



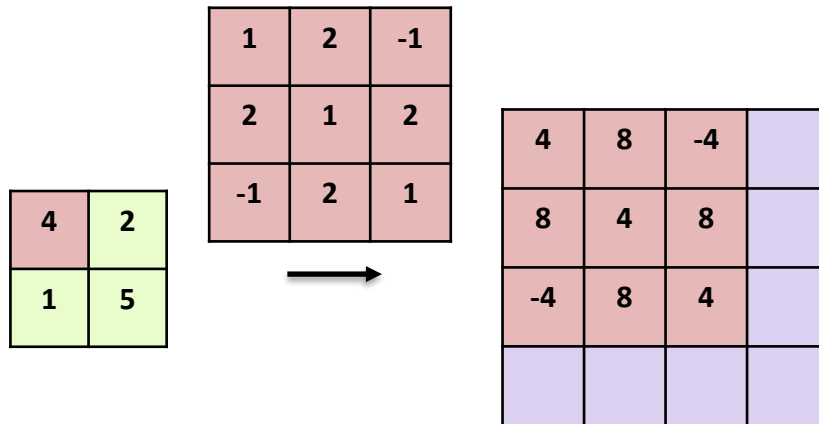
Up-sampling: Transpose Convolution

- Transpose convolution with:
 - 2 by 2 filter
 - Stride 2



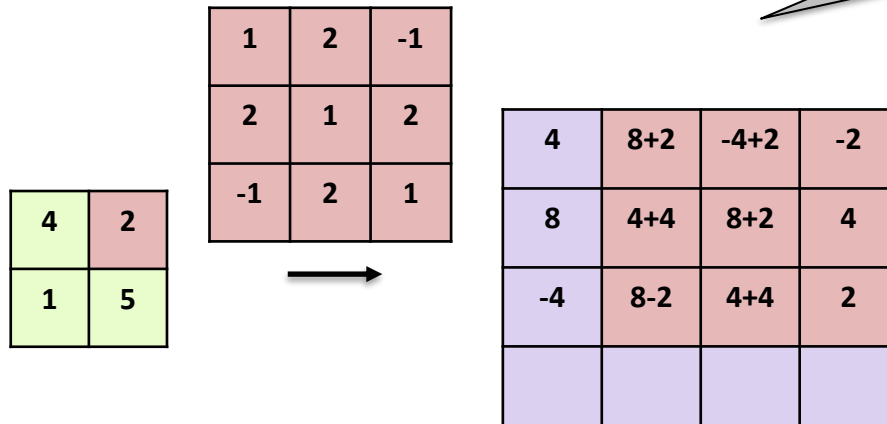
Up-sampling: Transpose Convolution

- Transpose convolution with:
 - 3 by 3 filter
 - Stride 1



Up-sampling: Transpose Convolution

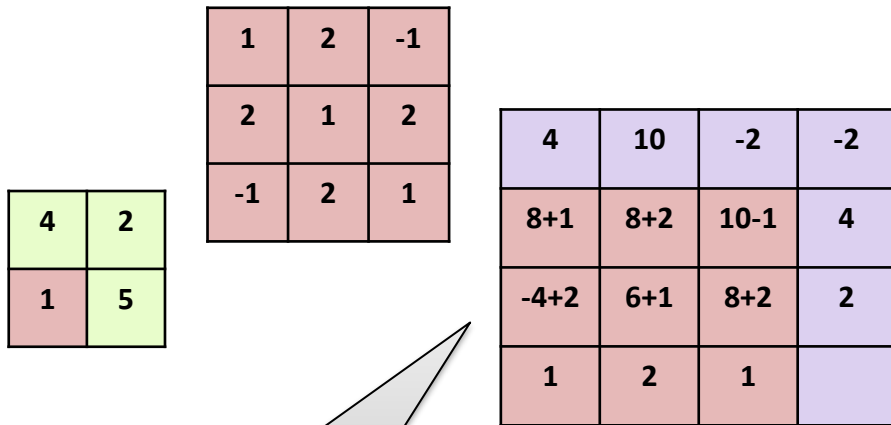
- Transpose convolution with:
 - 3 by 3 filter
 - Stride 1



The outputs on the overlapped area should be summed up.

Up-sampling: Transpose Convolution

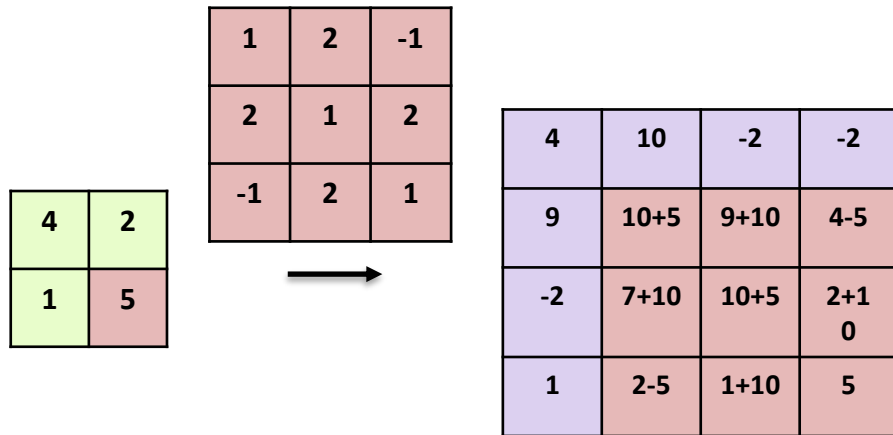
- Transpose convolution with:
 - 3 by 3 filter
 - Stride 1



The outputs on the overlapped area should be summed up.

Up-sampling: Transpose Convolution

- Transpose convolution with:
 - 3 by 3 filter
 - Stride 1



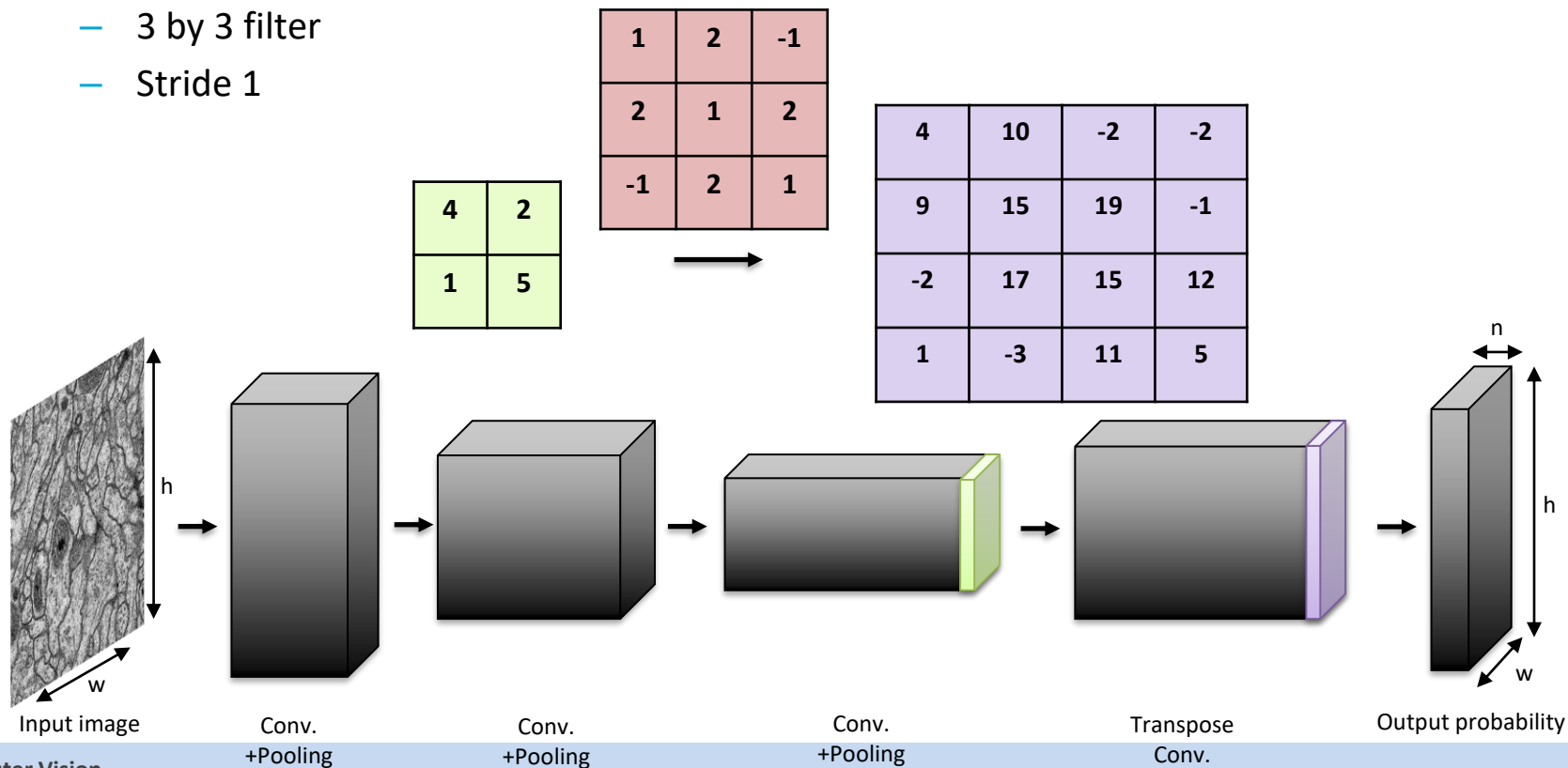
The outputs on the overlapped area should be summed up.

Up-sampling: Transpose Convolution

- Transpose convolution with:

- 3 by 3 filter

- Stride 1



Drawbacks

- Since we want the exact location of each pixel in segmentation, spatial information is crucial in image segmentation.
- Some spatial information is lost when you down-sample the feature maps (whether with strided convolution or pooling).
- Up-sampling can recover some of the lost information, but not all of it.
- Loss of the information leads to inaccurate localization.

U-net

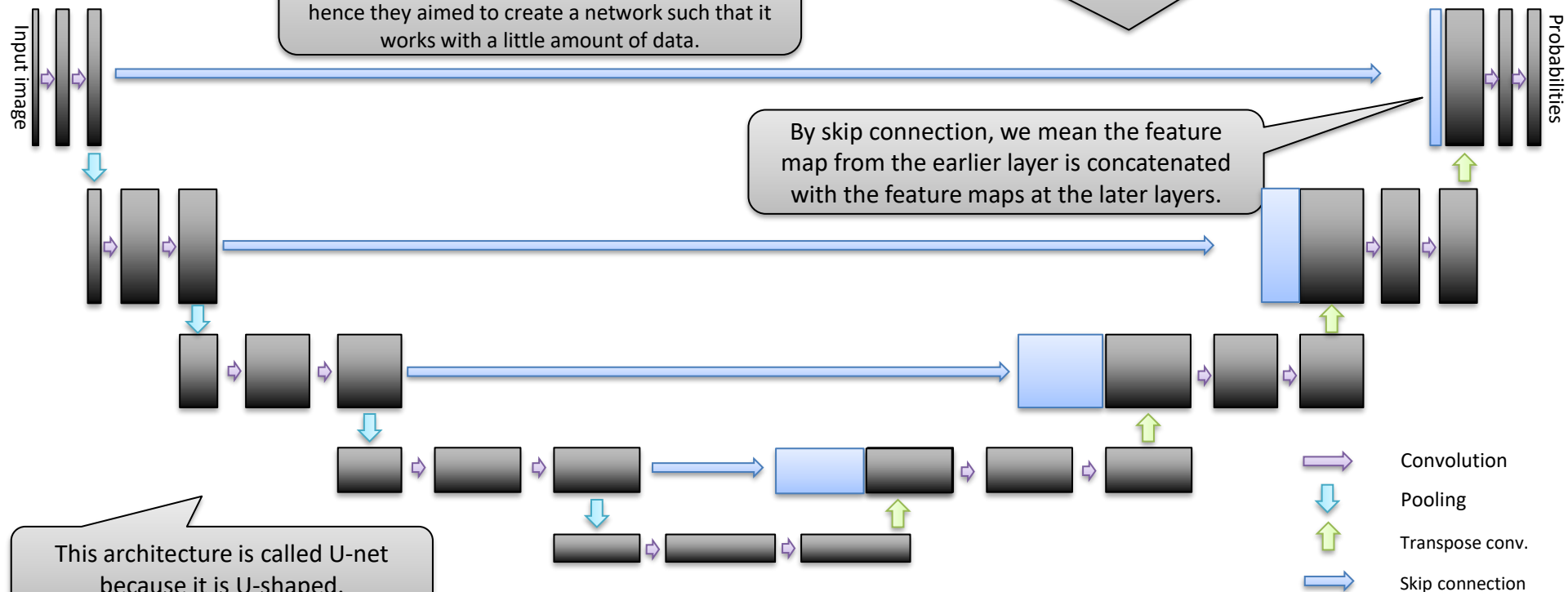
U-net was designed for medical image segmentation. Medical images are hard to acquire hence they aimed to create a network such that it works with a little amount of data.

Feature maps from earlier layers have higher resolutions and rich spatial information. Using a skip connection, later feature maps can use the earlier features to recover some lost data.

By skip connection, we mean the feature map from the earlier layer is concatenated with the feature maps at the later layers.

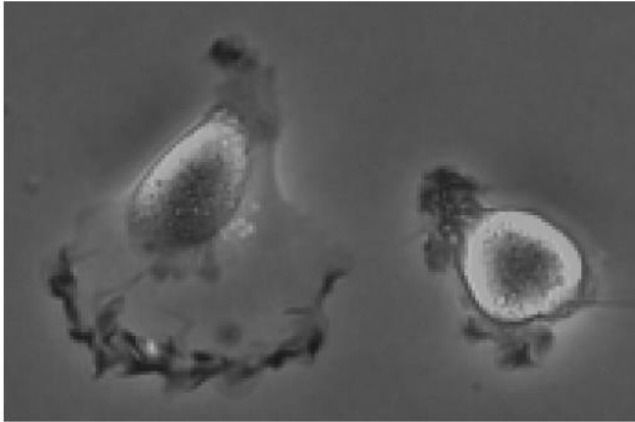
This architecture is called U-net because it is U-shaped.

(U-Net: Convolutional Networks for Biomedical Image Segmentation, Olaf Ronneberger et al., (2015))



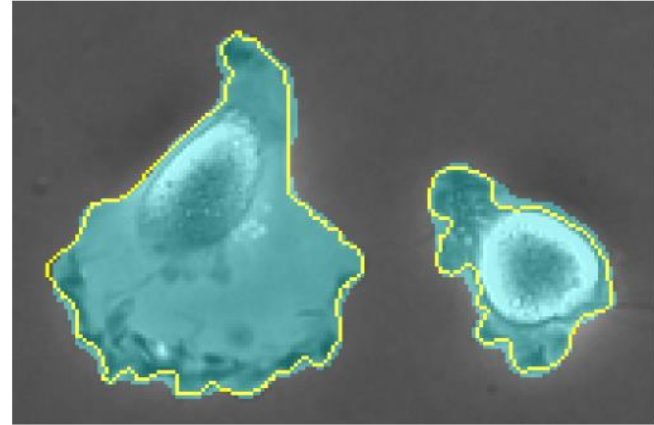
U-net in practice

input image



U-net
→

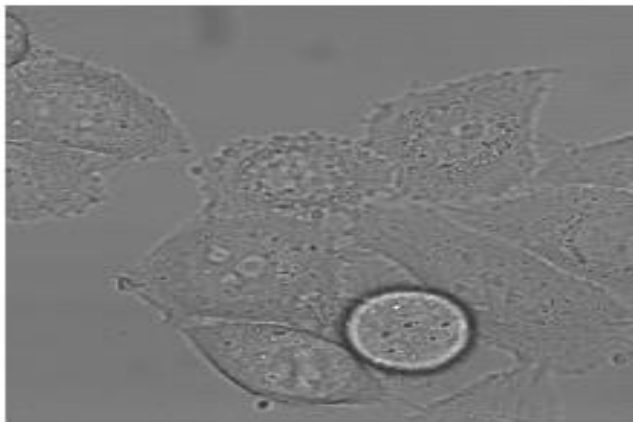
Segmentation result (cyan mask) with
manual ground truth
(yellow border)



(U-Net: Convolutional Networks for Biomedical Image Segmentation,
Olaf Ronneberger et al., (2015))

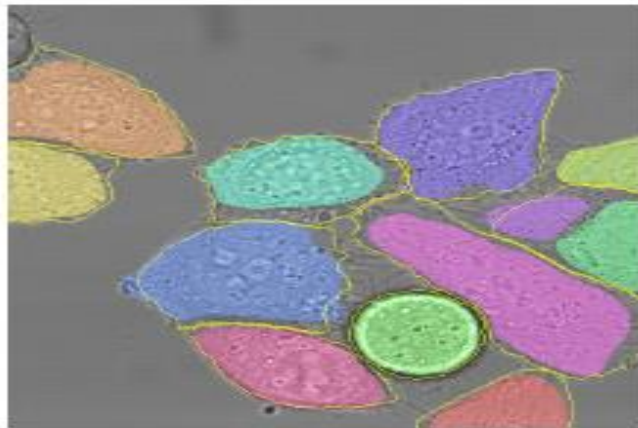
U-net in practice

input image



U-net
→

Segmentation result
(random colored masks) with manual
ground truth (yellow border).



(U-Net: Convolutional Networks for Biomedical Image Segmentation,
Olaf Ronneberger et al., (2015))

Non-medical image segmentation

- For non-medical images, we might be able to gather more pictures but other challenges might appear in non-medical image segmentation:
 - The background has more textures and complexity than medical images.
 - Objects come in different sizes, while things mostly have specific sizes in medical images.
 - The number of classes can be much more than the number of categories in medical image segmentation.
 - We could have lots of images, but it is very costly to provide labels for image segmentation (you need to label every pixels in the image!)

Non-medical image segmentation

- Besides the challenges, there are also opportunities:
 - Image segmentation and image classification are very similar. We have large datasets like ImageNet for image classification.
 - Many studies have been conducted to develop architectures for image classification as a core computer vision task.
- Now, the questions are:
 - can we take advantage of a large-scale dataset such as ImageNet (which is for image classification) to enhance the performances of our model for image segmentation?
 - Can we use the pre-designed architectures developed for image classification to solve image segmentation task?

Segmentation borrows from classification

- VGG16 was introduced for image classification.
 - VGG16 ends up with flatten and fully connected layers.

For segmentation, we do not want to ruin the structure by flattening the features. Locations are essential for segmentation.

Input image 224x224x3	Conv. 3x3 kernel size, 64 kernels	Conv. 3x3 kernel size, 64 kernels	Max pool, 2x2 kernel size, stride 2	Conv. 3x3 kernel size, 128 kernels	Conv. 3x3 kernel size, 128 kernels	Max pool, 2x2 kernel size, stride 2	Conv. 3x3 kernel size, 256 kernels	Conv. 3x3 kernel size, 256 kernels	Max pool, 2x2 kernel size, stride 2	Conv. 3x3 kernel size, 512 kernels	Conv. 3x3 kernel size, 512 kernels	Max pool, 2x2 kernel size, stride 2	Conv. 3x3 kernel size, 512 kernels	Conv. 3x3 kernel size, 512 kernels	Max pool, 2x2 kernel size, stride 2	Flatten	Fully connected, 4096 neurons	Fully connected, 1000 neurons
	224 × 224 × 64	224 × 224 × 64	112 × 112 × 64	112 × 112 × 128	112 × 112 × 128	56 × 56 × 128	56 × 56 × 256	56 × 56 × 256	28 × 28 × 256	28 × 28 × 512	28 × 28 × 512	14 × 14 × 512	14 × 14 × 512	14 × 14 × 512	14 × 14 × 512	25088 × 1	4096 × 1	1000 × 1

Output shapes of each layer

The fully connected layers learned to extract useful features. If we want to preserve the 2D structure, these fully connected layers need to be modified because they do not work with 2D input.

Segmentation borrows from classification

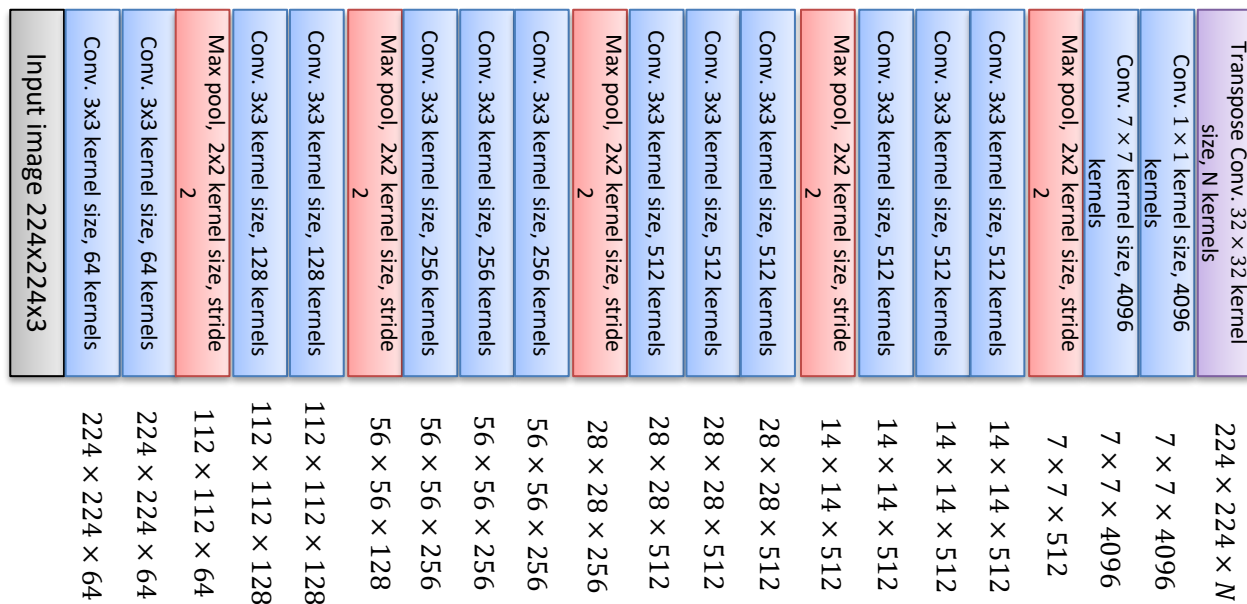
- It can be shown that the fully connected layers can be replaced with convolutional layers. At the same time, the number of parameters remains unchanged, and the output would be exactly the same as the fully connected layer.

Conv. 1 × 1 kernel size, 4096 kernels	7 × 7 × 4096
Conv. 7 × 7 kernel size, 4096 kernels	7 × 7 × 4096
Max pool, 2x2 kernel size, stride 2	7 × 7 × 512
Conv. 3x3 kernel size, 512 kernels	14 × 14 × 512
Conv. 3x3 kernel size, 512 kernels	14 × 14 × 512
Conv. 3x3 kernel size, 512 kernels	14 × 14 × 512
Max pool, 2x2 kernel size, stride 2	14 × 14 × 512
Conv. 3x3 kernel size, 512 kernels	28 × 28 × 512
Conv. 3x3 kernel size, 512 kernels	28 × 28 × 512
Conv. 3x3 kernel size, 512 kernels	28 × 28 × 512
Max pool, 2x2 kernel size, stride 2	28 × 28 × 256
Conv. 3x3 kernel size, 256 kernels	56 × 56 × 256
Conv. 3x3 kernel size, 256 kernels	56 × 56 × 256
Conv. 3x3 kernel size, 256 kernels	56 × 56 × 256
Max pool, 2x2 kernel size, stride 2	56 × 56 × 128
Conv. 3x3 kernel size, 128 kernels	112 × 112 × 128
Conv. 3x3 kernel size, 128 kernels	112 × 112 × 128
Max pool, 2x2 kernel size, stride 2	112 × 112 × 64
Conv. 3x3 kernel size, 64 kernels	224 × 224 × 64
Conv. 3x3 kernel size, 64 kernels	224 × 224 × 64
Input image 224x224x3	224 × 224 × 64

The weights learned from image classification on ImageNet can be used in this modified architecture to do transfer learning.

FCN: Fully convolutional neural network for segmentation

- Up-sample the modified VGG output with transpose convolution to match the input image spatial resolution.
- Since the VGG output needs to be up-sampled 32 times (from 7x7 to 224x224), this architecture is called FCN 32



N is number of classes.

(Fully Convolutional Networks for Semantic Segmentation, Jonathan Long et al., (2015))

FCN: Fully convolutional neural network for segmentation

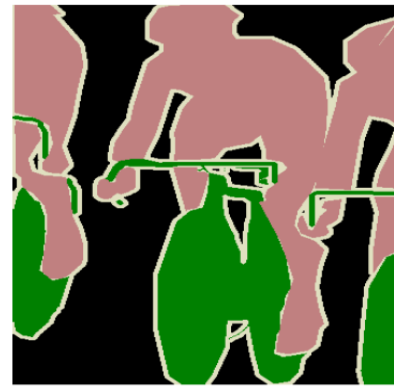
- Lots of details are missed, very poor localization.



Input image



Predicted
(FCN 32)

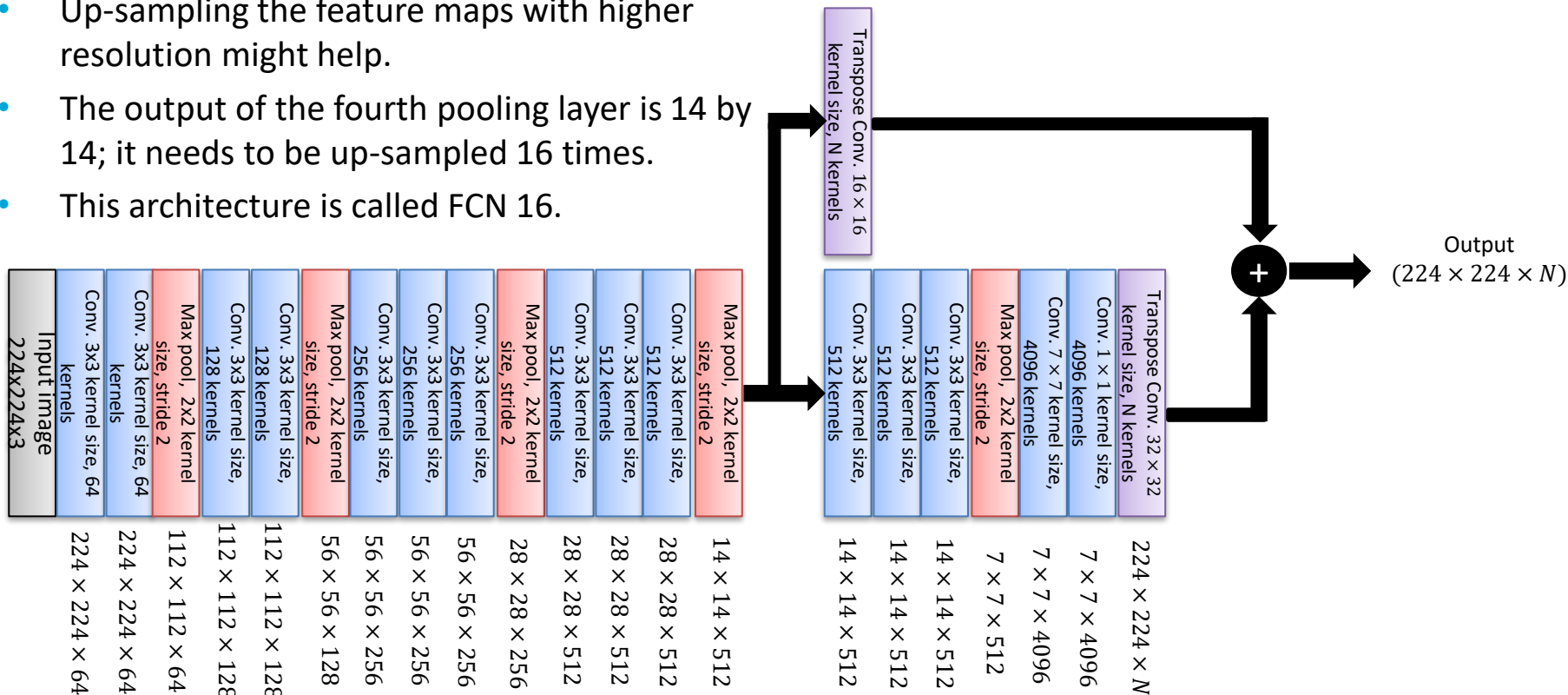


Ground Truth

(Fully Convolutional Networks for Semantic Segmentation, Jonathan Long et al., (2015))

FCN: Fully convolutional neural network for segmentation

- Up-sampling the feature maps with higher resolution might help.
- The output of the fourth pooling layer is 14 by 14; it needs to be up-sampled 16 times.
- This architecture is called FCN 16.



(Fully Convolutional Networks for Semantic Segmentation, Jonathan Long et al., (2015))

FCN: Fully convolutional neural network for segmentation

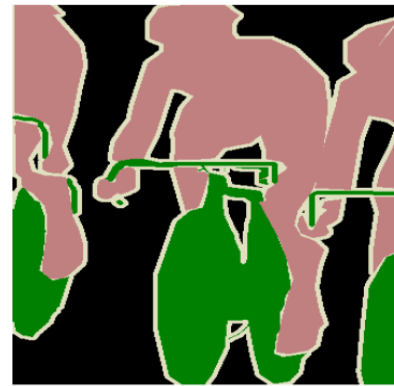
- Better than FCN 32, but good enough.



Input image



Predicted
(FCN 16)

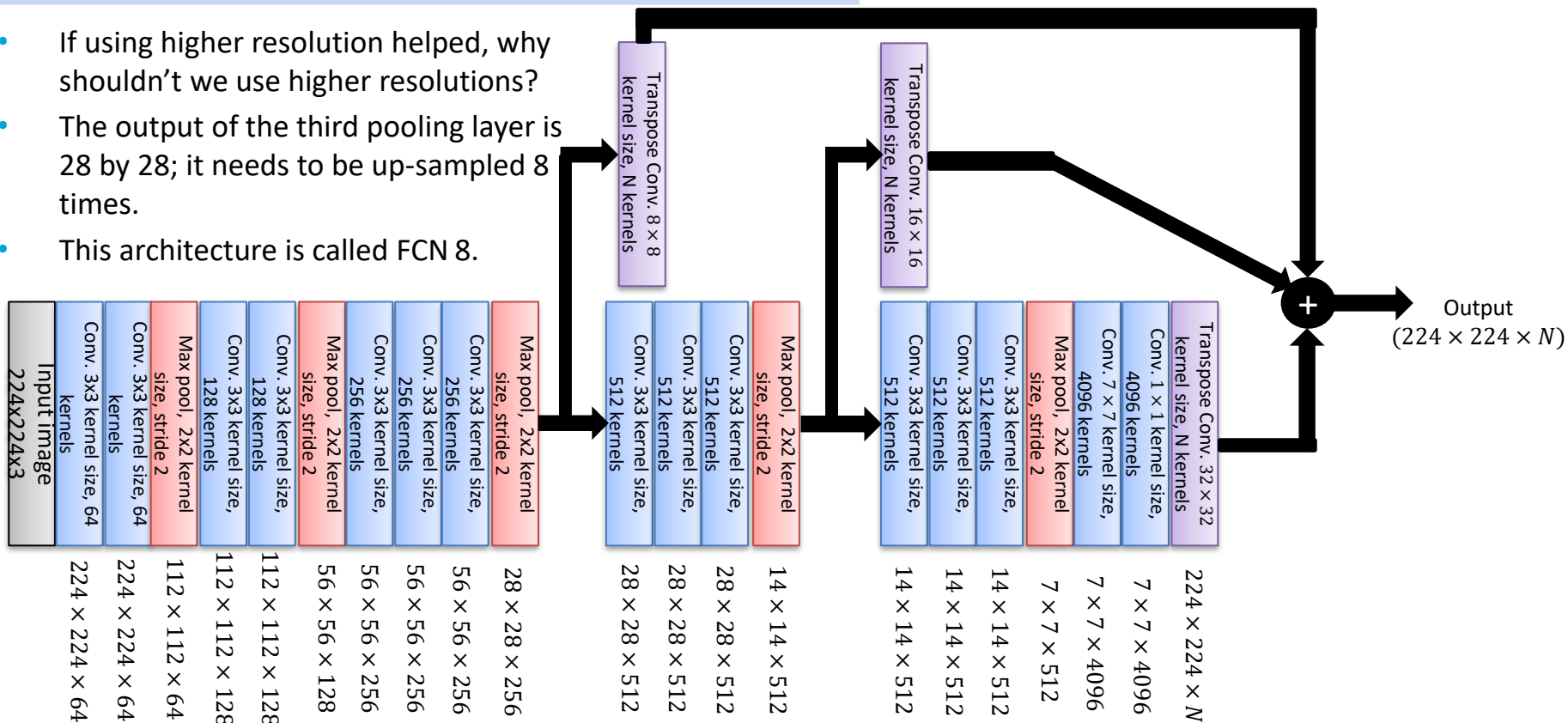


Ground Truth

(Fully Convolutional Networks for Semantic Segmentation, Jonathan Long et al., (2015))

FCN: Fully convolutional neural network for segmentation

- If using higher resolution helped, why shouldn't we use higher resolutions?
- The output of the third pooling layer is 28 by 28; it needs to be up-sampled 8 times.
- This architecture is called FCN 8.



(Fully Convolutional Networks for Semantic Segmentation, Jonathan Long et al., (2015))

FCN: Fully convolutional neural network for segmentation

- Better, but not much.



Input image



Predicted
(FCN 8)



Ground Truth

(Fully Convolutional Networks for Semantic Segmentation, Jonathan Long et al., (2015))

Drawbacks

- There might be two problems with FCN:
 - Up-sampling from 7×7 to 224×224 is very aggressive. Lots of details might be lost.
 - The computational cost in the 14th layer is prohibitive.
 - ✓ Parameters of each kernel in 14th layer = $7 \times 7 \times 512 + 1 = 25089$
 - ✓ Total no. of parameters in the 14th layer = $4096 \times 25089 = 102,764,544$!

Conv. 1×1 kernel size, 4096 kernels	$7 \times 7 \times 4096$
Conv. 7×7 kernel size, 4096 kernels	$7 \times 7 \times 4096$
Max pool, 2×2 kernel size, stride 2	$7 \times 7 \times 512$
Conv. 3×3 kernel size, 512 kernels	$14 \times 14 \times 512$
Conv. 3×3 kernel size, 512 kernels	$14 \times 14 \times 512$
Conv. 3×3 kernel size, 512 kernels	$14 \times 14 \times 512$
Max pool, 2×2 kernel size, stride 2	$14 \times 14 \times 512$
Conv. 3×3 kernel size, 512 kernels	$28 \times 28 \times 512$
Conv. 3×3 kernel size, 512 kernels	$28 \times 28 \times 512$
Conv. 3×3 kernel size, 512 kernels	$28 \times 28 \times 512$
Max pool, 2×2 kernel size, stride 2	$28 \times 28 \times 256$
Conv. 3×3 kernel size, 256 kernels	$56 \times 56 \times 256$
Conv. 3×3 kernel size, 256 kernels	$56 \times 56 \times 256$
Conv. 3×3 kernel size, 256 kernels	$56 \times 56 \times 256$
Max pool, 2×2 kernel size, stride 2	$56 \times 56 \times 128$
Conv. 3×3 kernel size, 128 kernels	$112 \times 112 \times 128$
Conv. 3×3 kernel size, 128 kernels	$112 \times 112 \times 128$
Max pool, 2×2 kernel size, stride 2	$112 \times 112 \times 64$
Conv. 3×3 kernel size, 64 kernels	$224 \times 224 \times 64$
Conv. 3×3 kernel size, 64 kernels	$224 \times 224 \times 64$
Input image $224 \times 224 \times 3$	$224 \times 224 \times 64$

Let's modify the FCN

- We can decrease the stride in the 4th and 5th pooling layers to preserve the 28×28 spatial resolution.
- Smaller filters (3×3) and fewer filters (1024) can be used in the 14th and 15th Conv. layers to lower the computational cost.
 - Parameters of each kernel in 14th layer = $3 \times 3 \times 512 + 1 = 4609$
 - Total no. of parameters in the 14th layer = $4096 \times 25089 = 4,719,616$

The number of filters has reduced from 4096 to 1024 in the two last layers.

Conv. 1×1 kernel size, 1024 kernels	$28 \times 28 \times 1024$
Conv. 3×3 kernel size, 1024 kernels	$28 \times 28 \times 1024$
Max pool, 2×2 kernel size, stride 1	$28 \times 28 \times 512$
Conv. 3×3 kernel size, 512 kernels	$28 \times 28 \times 512$
Conv. 3×3 kernel size, 512 kernels	$28 \times 28 \times 512$
Conv. 3×3 kernel size, 512 kernels	$28 \times 28 \times 512$
Max pool, 2×2 kernel size, stride 1	$28 \times 28 \times 512$
Conv. 3×3 kernel size, 512 kernels	$28 \times 28 \times 512$
Conv. 3×3 kernel size, 512 kernels	$28 \times 28 \times 512$
Conv. 3×3 kernel size, 512 kernels	$28 \times 28 \times 512$
Max pool, 2×2 kernel size, stride 2	$28 \times 28 \times 256$
Conv. 3×3 kernel size, 256 kernels	$56 \times 56 \times 256$
Conv. 3×3 kernel size, 256 kernels	$56 \times 56 \times 256$
Conv. 3×3 kernel size, 256 kernels	$56 \times 56 \times 256$
Max pool, 2×2 kernel size, stride 2	$56 \times 56 \times 128$
Conv. 3×3 kernel size, 128 kernels	$112 \times 112 \times 128$
Conv. 3×3 kernel size, 128 kernels	$112 \times 112 \times 128$
Max pool, 2×2 kernel size, stride 2	$112 \times 112 \times 64$
Conv. 3×3 kernel size, 64 kernels	$224 \times 224 \times 64$
Conv. 3×3 kernel size, 64 kernels	$224 \times 224 \times 64$
Input image $224 \times 224 \times 3$	$224 \times 224 \times 64$

3 by 3 filters are used rather than 7 by 7.

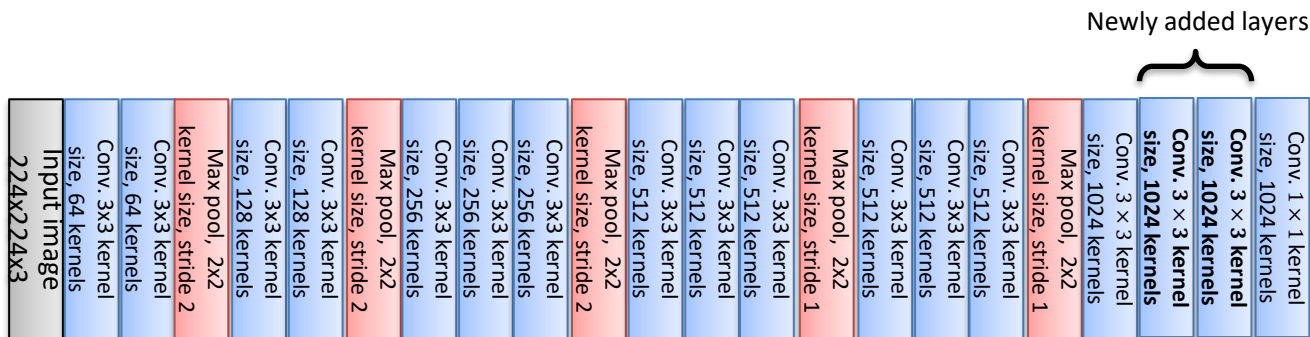
One more thing to note

- Detecting large objects in the image requires large receptive fields.
- Large receptive fields are acquired either with large kernels or increased network depth with smaller filters.
 - Using large filters was what we tried to avoid during the modification process. We got rid of 7×7 kernels and replaced them with 3×3 filters to reduce the computational cost.
 - As we have learned before, a receptive field of a 7×7 filter equals to receptive field of 3 consequent 3×3 filters. It means if we want to use smaller filters, we need to add 3 layers to the network with 3×3 kernels.

Conv. 1×1 kernel size, 1024 kernels
Conv. 3×3 kernel size, 1024 kernels
Max pool, 2x2 kernel size, stride 1
Conv. 3x3 kernel size, 512 kernels
Conv. 3x3 kernel size, 512 kernels
Conv. 3x3 kernel size, 512 kernels
Max pool, 2x2 kernel size, stride 1
Conv. 3x3 kernel size, 512 kernels
Conv. 3x3 kernel size, 512 kernels
Conv. 3x3 kernel size, 512 kernels
Max pool, 2x2 kernel size, stride 2
Conv. 3x3 kernel size, 256 kernels
Conv. 3x3 kernel size, 256 kernels
Conv. 3x3 kernel size, 256 kernels
Max pool, 2x2 kernel size, stride 2
Conv. 3x3 kernel size, 128 kernels
Conv. 3x3 kernel size, 128 kernels
Max pool, 2x2 kernel size, stride 2
Conv. 3x3 kernel size, 64 kernels
Input image 224x224x3

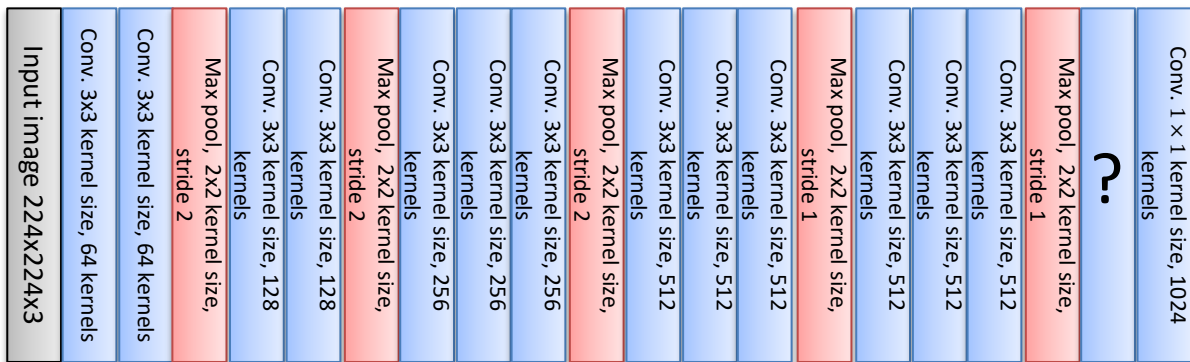
One more thing to note

- As we have learned before, a receptive field of a 7×7 filter equals to receptive field of 3 consequent 3×3 filters. It means if we want to use smaller filters, we need to add 2 more layers to the network with 3×3 kernels.
 - Parameters of each kernel in each of the newly added layers = $3 \times 3 \times 1024 + 1 = 9217$
 - Total no. of parameters in each of the newly added layers = $1024 \times 9217 = 9,438,208$
 - Total no. of parameters of three new layers = $9,438,208 \times 2 = 18,876,416$
 - This is much less than using a 7×7 layer with 1024 filters. (Such a layer would have 51 millions of parameters)



One more thing to note

- Now, the question is: can we further decrease the number of parameters while maintaining the same receptive field as a 7 by 7 filter?



One more thing to note

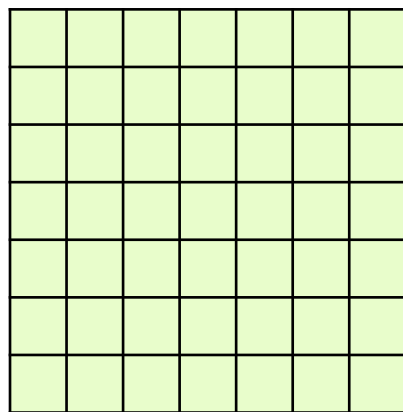
- Now, the question is: can we further decrease the number of parameters while maintaining the same receptive field as a 7 by 7 filter?
 - Yes! Atrous convolution!

Conv. 1×1 kernel size, 1024 kernels
Atrous convolution
Max pool, 2x2 kernel size, stride 1
Conv. 3x3 kernel size, 512 kernels
Conv. 3x3 kernel size, 512 kernels
Conv. 3x3 kernel size, 512 kernels
Max pool, 2x2 kernel size, stride 1
Conv. 3x3 kernel size, 512 kernels
Conv. 3x3 kernel size, 512 kernels
Conv. 3x3 kernel size, 512 kernels
Max pool, 2x2 kernel size, stride 2
Conv. 3x3 kernel size, 256 kernels
Conv. 3x3 kernel size, 256 kernels
Conv. 3x3 kernel size, 256 kernels
Max pool, 2x2 kernel size, stride 2
Conv. 3x3 kernel size, 128 kernels
Conv. 3x3 kernel size, 128 kernels
Max pool, 2x2 kernel size, stride 2
Conv. 3x3 kernel size, 64 kernels
Conv. 3x3 kernel size, 64 kernels
Input image 224x224x3

What is Atrous convolution?

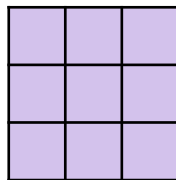
The purple area shows where the kernel is applied in each stage.

- Atrous convolution has a non-trainable called dilatation rate.
- This rate defines a spacing between the values in a kernel.
- Let's convolve the input with a 3 by 3 kernel, once with regular convolution (what we had so far) and another time with Atrous convolution to see the difference. (stride is 1 for both of them)



Input feature map

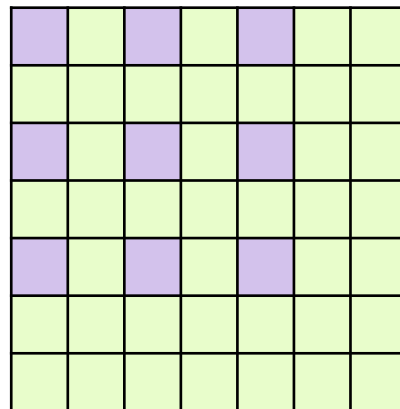
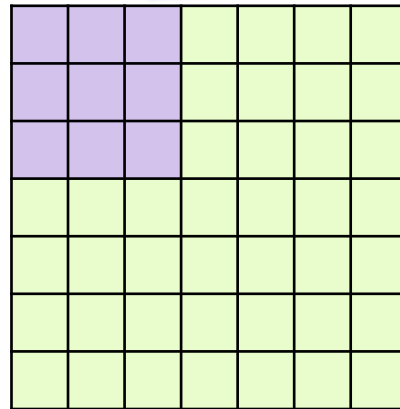
*



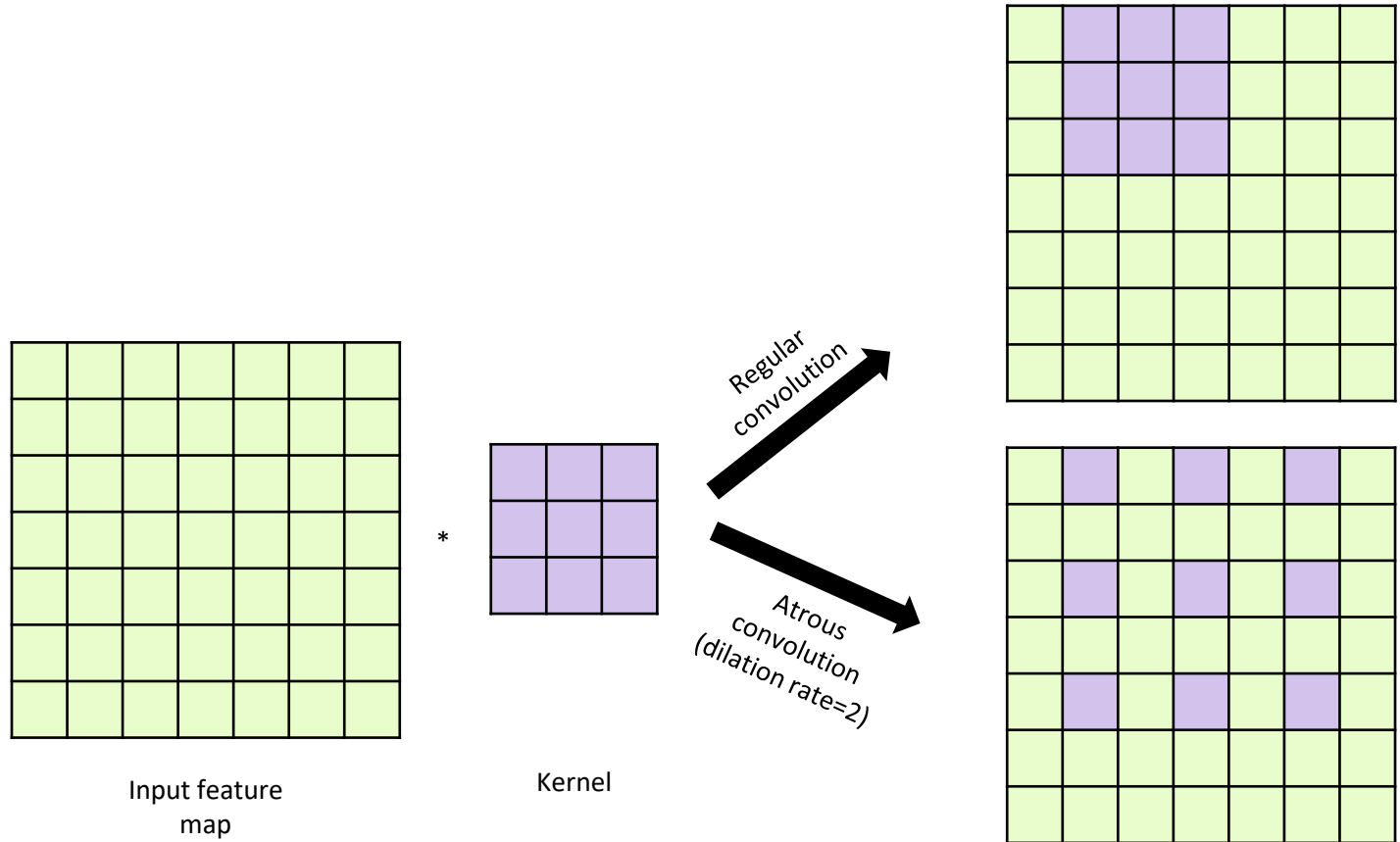
Kernel

Regular convolution

Atrous convolution (dilation rate=2)



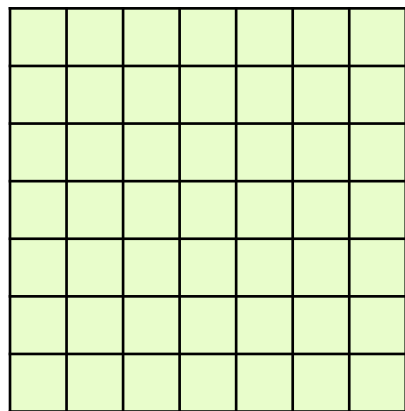
What is Atrous convolution?



What is Atrous convolution?

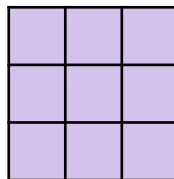
Atrous convolution is also known as dilated convolution or hole algorithm.

As you see, regular convolution is a particular case of Atrous convolution where the dilatation rate is equal to 1.



Input feature map

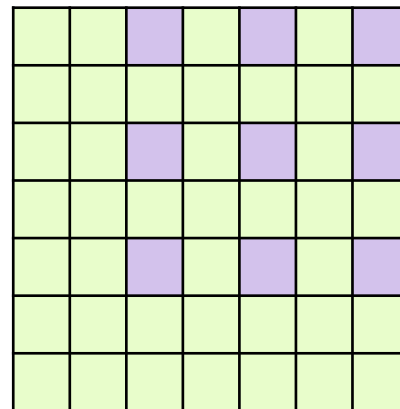
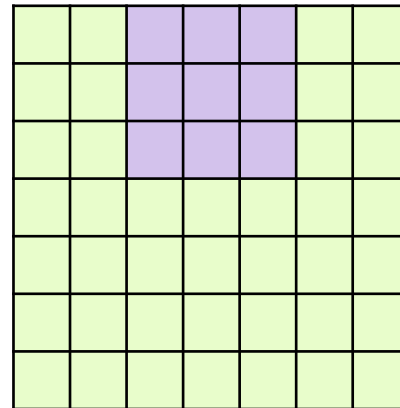
*



Kernel

Regular
convolution

Atrous
convolution
(dilation rate=2)



Although the kernel is 3 by 3, with dilation rate of 2, the kernel has a 5 by 5 receptive field.

Modification has been almost completed!

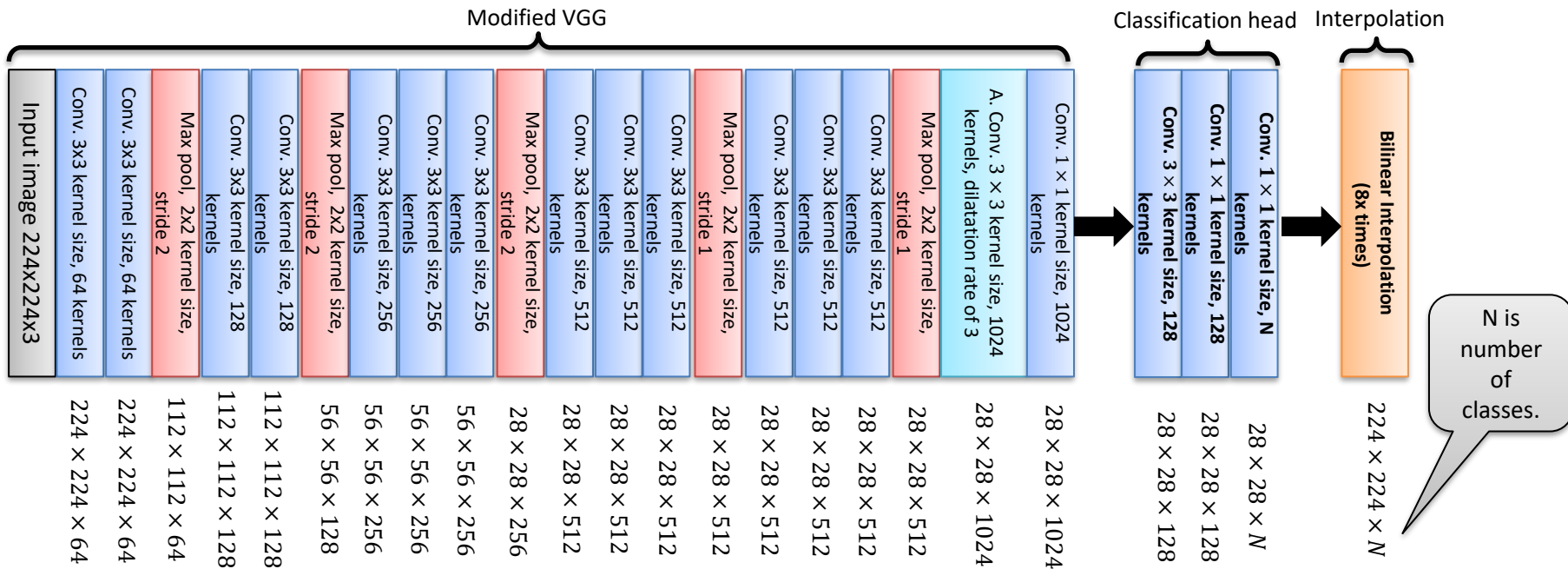
- As you guess, a 3 by 3 filter with a dilation rate of 3 would have the same receptive field as a 7 by 7 filter, while it has much fewer no. of parameters.
- This idea was used in DeepLab-v1 which was introduced in 2016
- To do the segmentation, the output feature maps need to be up-sampled to match the input image resolution.

Conv. 1×1 kernel size, 1024 kernels
A. Conv. 3×3 kernel size, 1024 kernels, dilation rate of 3
Max pool, 2×2 kernel size, stride 1
Conv. 3×3 kernel size, 512 kernels
Conv. 3×3 kernel size, 512 kernels
Conv. 3×3 kernel size, 512 kernels
Max pool, 2×2 kernel size, stride 1
Conv. 3×3 kernel size, 512 kernels
Conv. 3×3 kernel size, 512 kernels
Conv. 3×3 kernel size, 512 kernels
Max pool, 2×2 kernel size, stride 2
Conv. 3×3 kernel size, 256 kernels
Conv. 3×3 kernel size, 256 kernels
Conv. 3×3 kernel size, 256 kernels
Max pool, 2×2 kernel size, stride 2
Conv. 3×3 kernel size, 128 kernels
Conv. 3×3 kernel size, 128 kernels
Max pool, 2×2 kernel size, stride 2
Conv. 3×3 kernel size, 64 kernels
Conv. 3×3 kernel size, 64 kernels
Input image $224 \times 224 \times 3$

(SEMANTIC IMAGE SEGMENTATION WITH DEEP CONVOLUTIONAL NETS AND FULLY CONNECTED CRFS, Liang-Chieh Chen et al., 2016)

DeepLab-v1

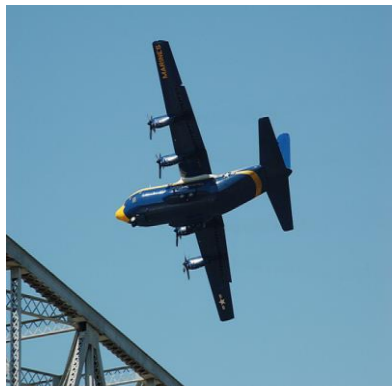
- DeepLab-v1 was founded based on the VGG architecture.



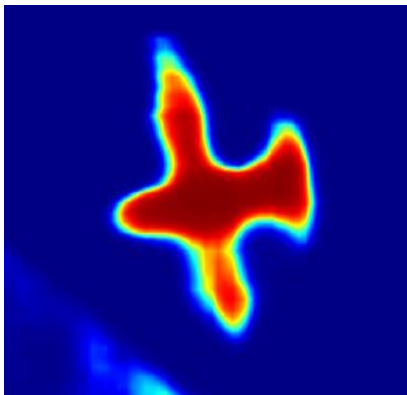
(SEMANTIC IMAGE SEGMENTATION WITH DEEP CONVOLUTIONAL NETS AND FULLY CONNECTED CRFS, Liang-Chieh Chen et al., 2016)

DeepLab-v1

- The result is not bad, but lots of details are missed, especially in the boundaries of the plane.



Input image



Predicted
(DeepLab-v1 (almost!))

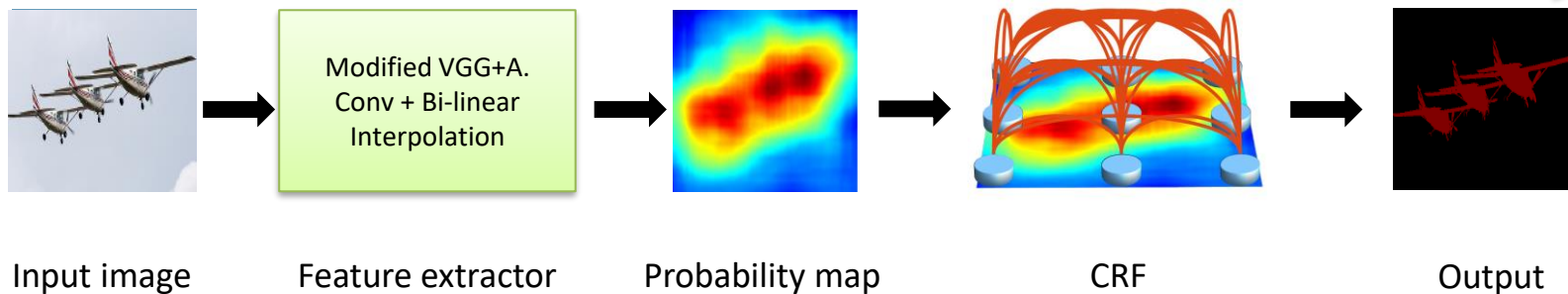


Ground Truth

(SEMANTIC IMAGE SEGMENTATION WITH DEEP CONVOLUTIONAL NETS AND FULLY CONNECTED CRFS, Liang-Chieh Chen et al., 2016)

DeepLab-v1 + CRF

- One way to enhance the results, is using a fully connected conditional random field (CRF).
- The mathematics behind this CRF model is beyond the scope of this course.
- You can think of it as a module which investigates the relationship between different areas in the probability map.
- Based on the obtain information, it tries to recover lost details.



(SEMANTIC IMAGE SEGMENTATION WITH DEEP CONVOLUTIONAL NETS AND FULLY CONNECTED CRFS, Liang-Chieh Chen et al., 2016)

DeepLab-v1 + CRF vs. FCN8

DeepLab-v1 produces sharper boundaries with more details

Input image



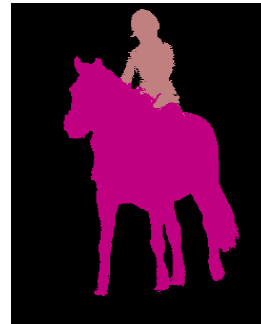
Ground Truth



FCN8



DeepLab-v1 + CRF



(SEMANTIC IMAGE SEGMENTATION WITH DEEP CONVOLUTIONAL NETS AND FULLY CONNECTED CRFS, Liang-Chieh Chen et al., 2016)

We need a quantitative metric for localization

- We spot the quality of localization, visually but it makes more sense to define a metric for that.
- In object detection, we saw that quality of localization could be measured through IoU (Intersection over Union)
- The higher the IoU, the better localization.
 - The IoU for the yellow box is higher than the IoU for the blue box.
 - Hence, the model which predicted the yellow box is better at localization than the other model which predicted the blue box.

Prediction 1




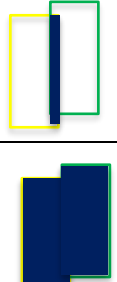
Prediction 2



Ground Truth



(Rich feature hierarchies for accurate object detection and semantic segmentation
, Ross Girshick et. al, 2014)

$$\text{IoU for box 1} = \frac{\text{Intersection}}{\text{Union}}$$

$$\text{IoU for box 2} = \frac{\text{Intersection}}{\text{Union}}$$


We need a quantitative metric for localization

- In segmentation, rather than bounding boxes, for each instance we have a mask.
- The IoU for each class can be calculated as:
 - $\text{IoU} = \frac{\text{Target mask} \cap \text{Prediction Mask}}{\text{Target mask} \cup \text{Prediction Mask}}$
 - The intersection ($\text{Target mask} \cap \text{Prediction Mask}$) is comprised of the pixels found in both the prediction mask and the ground truth mask
 - The union ($\text{Target mask} \cup \text{Prediction Mask}$) is simply comprised of all pixels found in either the prediction or target mask.

Ground Truth



Prediction



No. of yellow pixels()

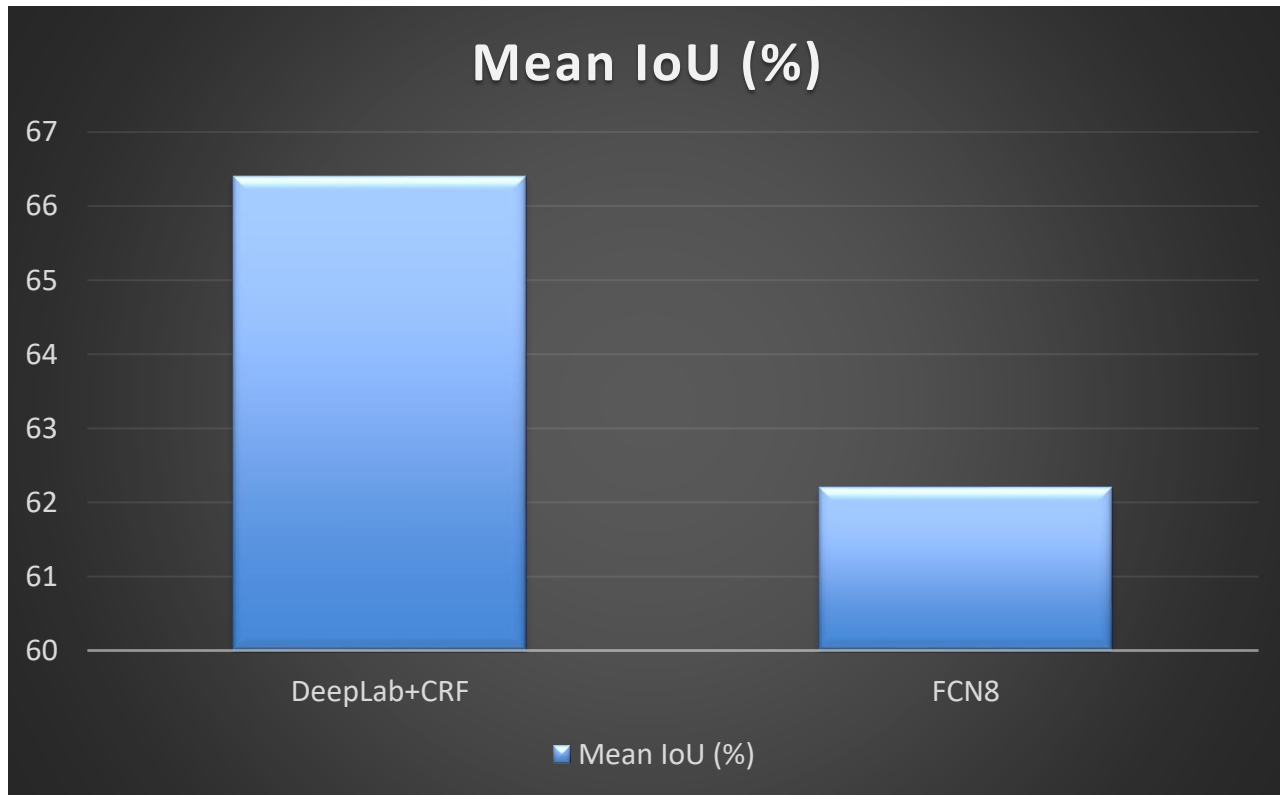


$$\text{IoU}_{\text{person}} = \frac{\text{No. of yellow pixels()}}{\text{No. of yellow pixels()}} = 0.68$$

No. of yellow pixels()



DeepLab-v1 + CRF vs. FCN8 on Pascal VOC dataset

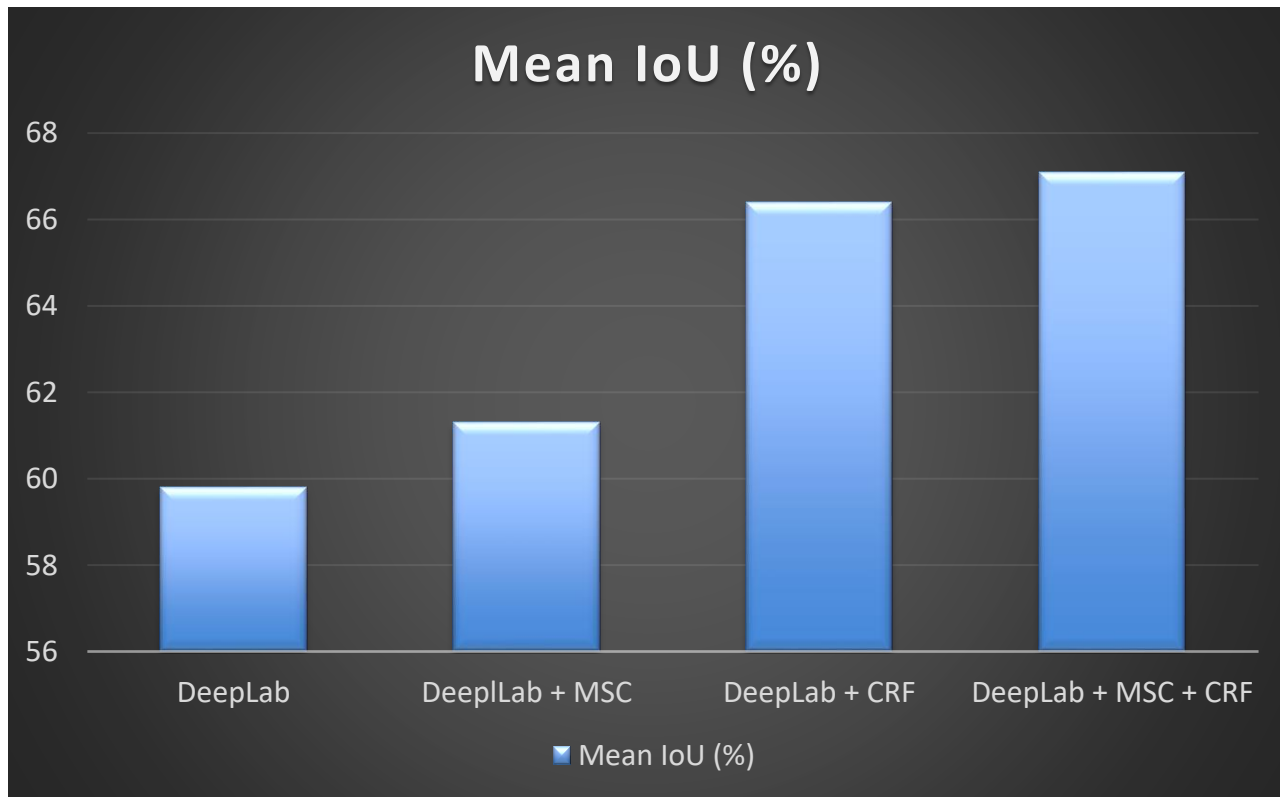


(SEMANTIC IMAGE SEGMENTATION WITH DEEP CONVOLUTIONAL NETS AND FULLY CONNECTED CRFS, Liang-Chieh Chen et al., 2016)

We want more

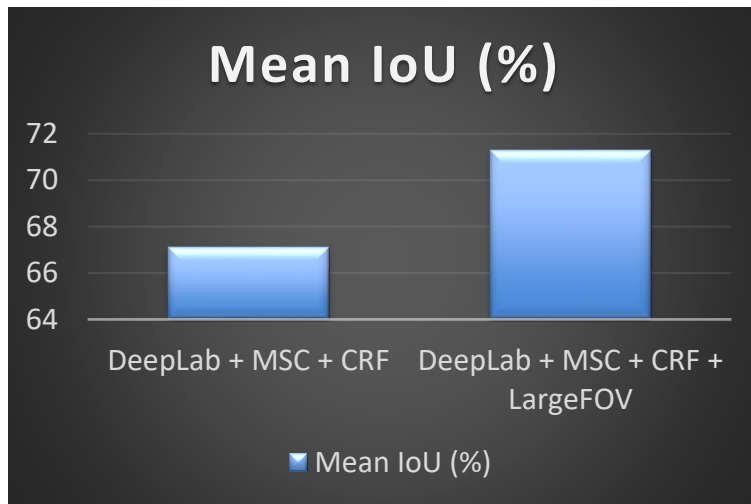
- Since using features from earlier layers helped U-Net and FCN to improve the results.
- The DeepLab-v1 also has a multi-scale version to take advantage of the early features.
- This version is called MSC DeepLab-v1

Putting it all together: DeepLab-v1 + CRF + MSC on Pascal VOC dataset



Effect of dilation rate

- We know that increasing the dilation rate helps the network to see larger areas in the image.
- So, what happens if we increase the dilation rate from $r = 3$ to $r = 12$?
- With $r = 12$, the network sees larger areas in the image. Because of that, we can say this network has a large field of view (FOV) or large receptive field.

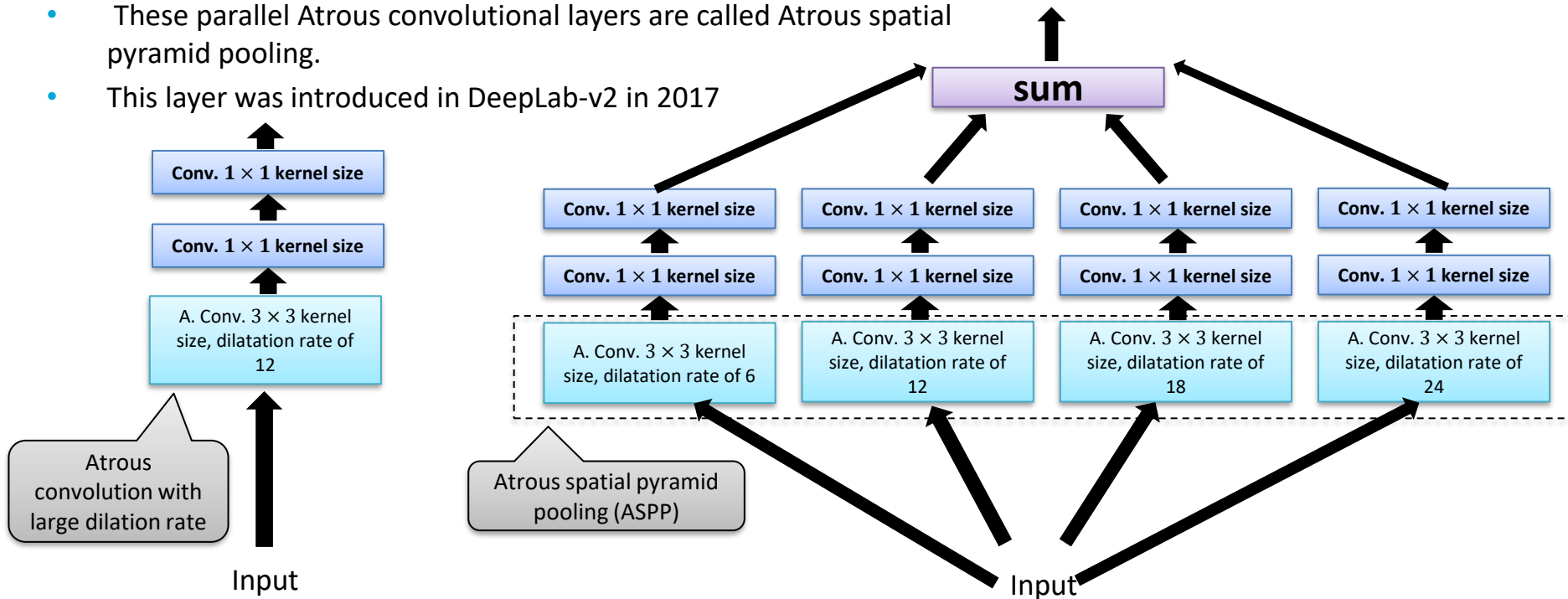


Objects come in different sizes

- Although increasing the dilation rate helps the network larger areas in the image, objects are not large all the time.
- Increasing the dilation rate in the Atrous convolutional layer increases the performance of the model on large objects.
- But, the network could be inaccurate when facing small or medium-sized objects.
- So, what should we do?

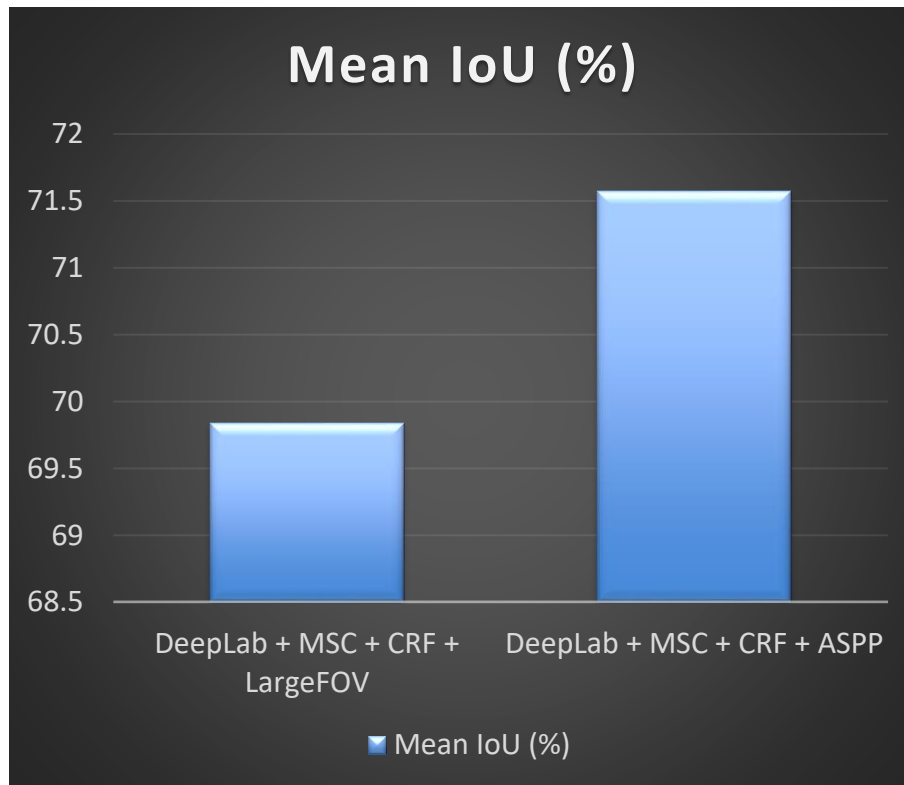
Atrous Spatial Pyramid Pooling

- Using Atrous convolution layers in parallel with different dilation rates help the network to see objects of different sizes.
- These parallel Atrous convolutional layers are called Atrous spatial pyramid pooling.
- This layer was introduced in DeepLab-v2 in 2017



(DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, Liang-Chieh Chen et al., 2017)

Effect of ASPP layer



(DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, Liang-Chieh Chen et al., 2017)

VGG might not be the best

- In the previous lectures, we saw that VGG is not the most accurate image classifier.
- So, what happens if we substitute VGG with another network like ResNet in DeepLab-v2?



Input image



VGG
+ASPP



VGG
+ASPP
+CRF



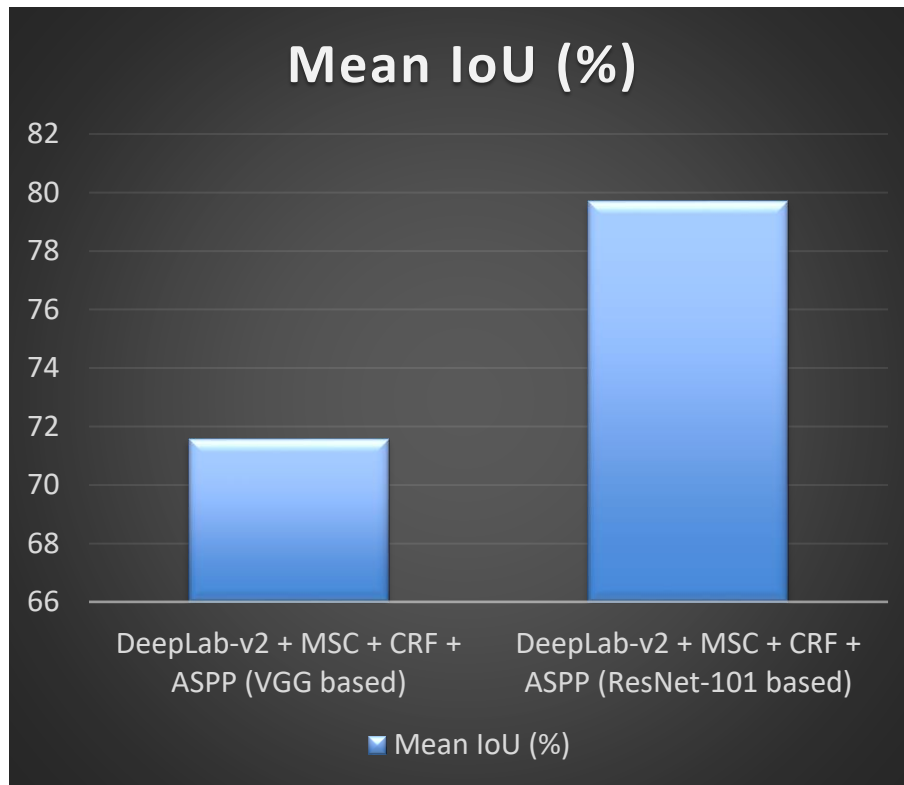
ResNet-101
+ASPP



ResNet-101
+ASPP
+CRF

- As you can see, the CRF is critical for accurate prediction along object boundaries with VGG, whereas ResNet-101 has acceptable performance even before CRF.

VGG might not be the best



(DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, Liang-Chieh Chen et al., 2017)

From DeepLab-v2 to DeepLab-v3

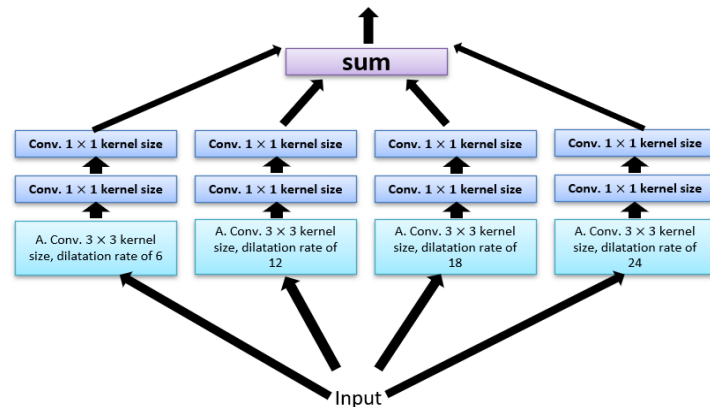
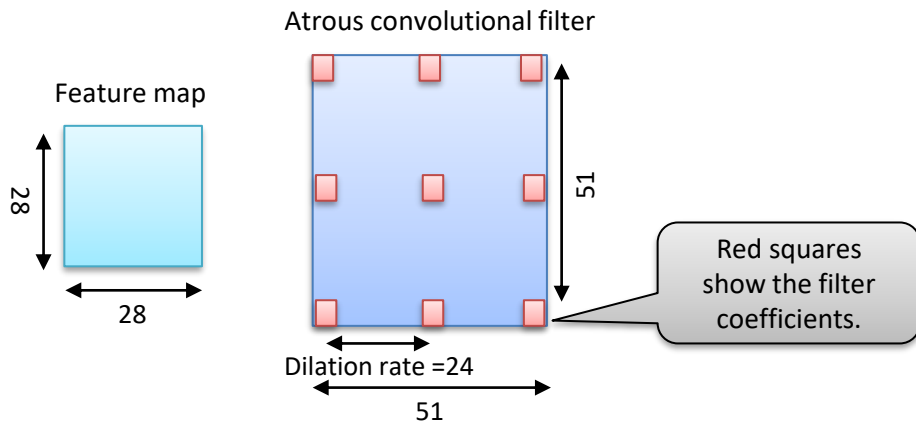
- The authors of DeepLab-v2 saw that while ResNet was used as the feature extractor, the performance was acceptable, even without CRF.
- So, they got rid of this module!
- DeepLab-v3 does not have CRF.



(Rethinking Atrous Convolution for Semantic Image Segmentation, Liang-Chieh Chen et al., 2017)

From DeepLab-v2 to DeepLab-v3

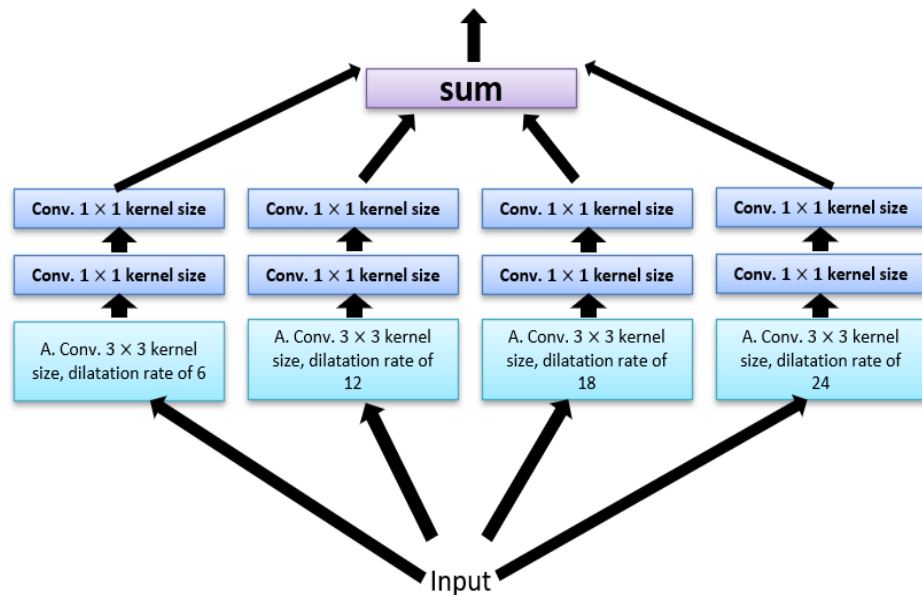
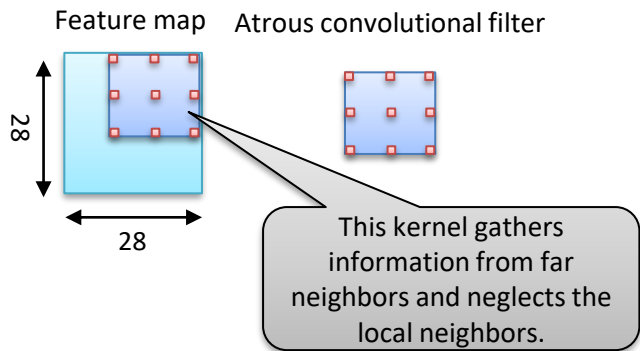
- There are problems with ASPP module in DeepLab-v2:
 - Dilation rate of 24 is too much.
 - ✓ Remember VGG as the backbone of DeepLab-v1. The spatial resolution of the final feature map was 28×28 . When the dilation rate is 24, the kernel size is 51×51 , even greater than the feature map. Consequently, most of the time, the filter weights are multiplied by zero-padding around the feature map, not the actual features.
 - ✓ This problem might not be as severe in ResNet as it is in VGG, but it exists. (The output feature map in ResNet would be 65×65)



(Rethinking Atrous Convolution for Semantic Image Segmentation, Liang-Chieh Chen et al., 2017)

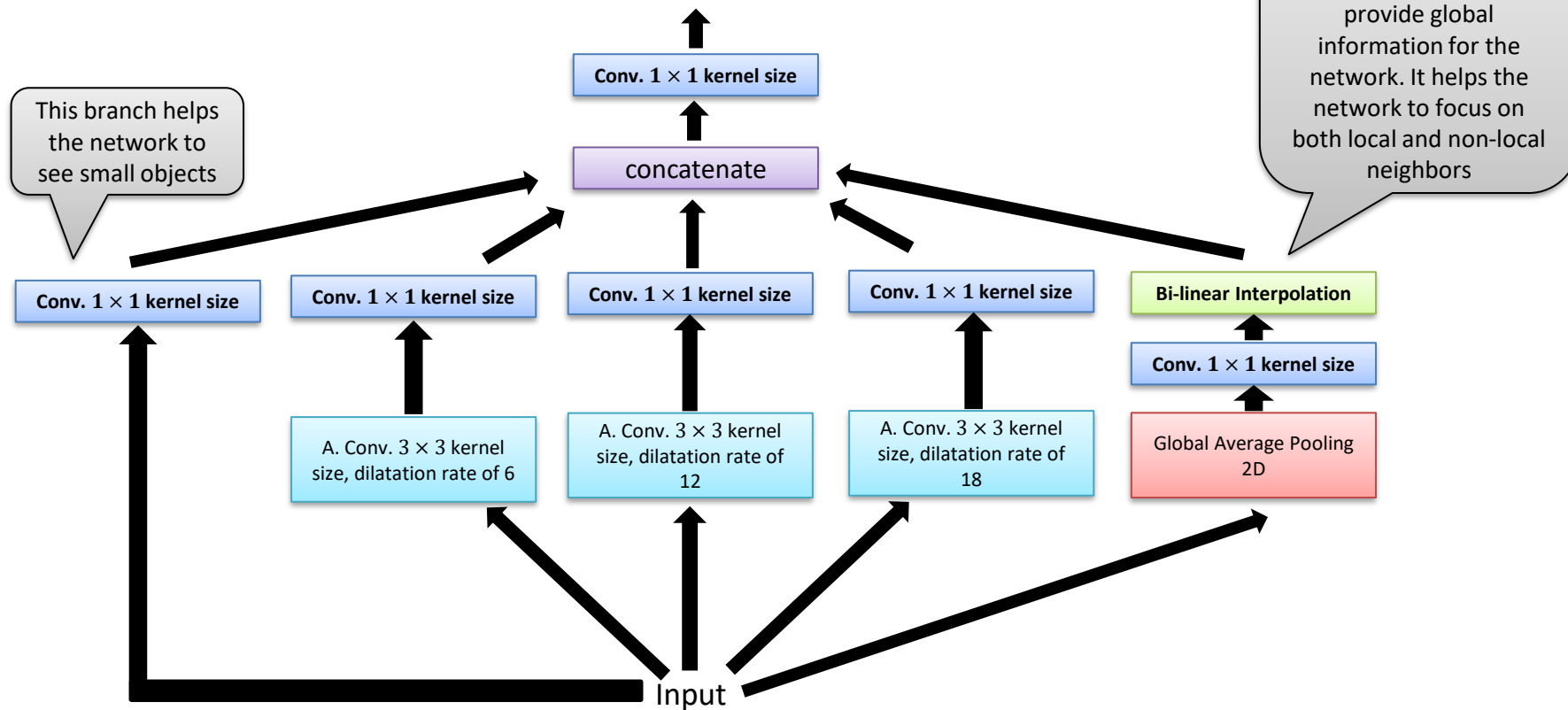
From DeepLab-v2 to DeepLab-v3

- There are problems with ASPP module in DeepLab-v2:
 - Even if the dilation rate is low enough, there are many empty spaces between the filter coefficients, and the filter is unaware of the local neighborhood.



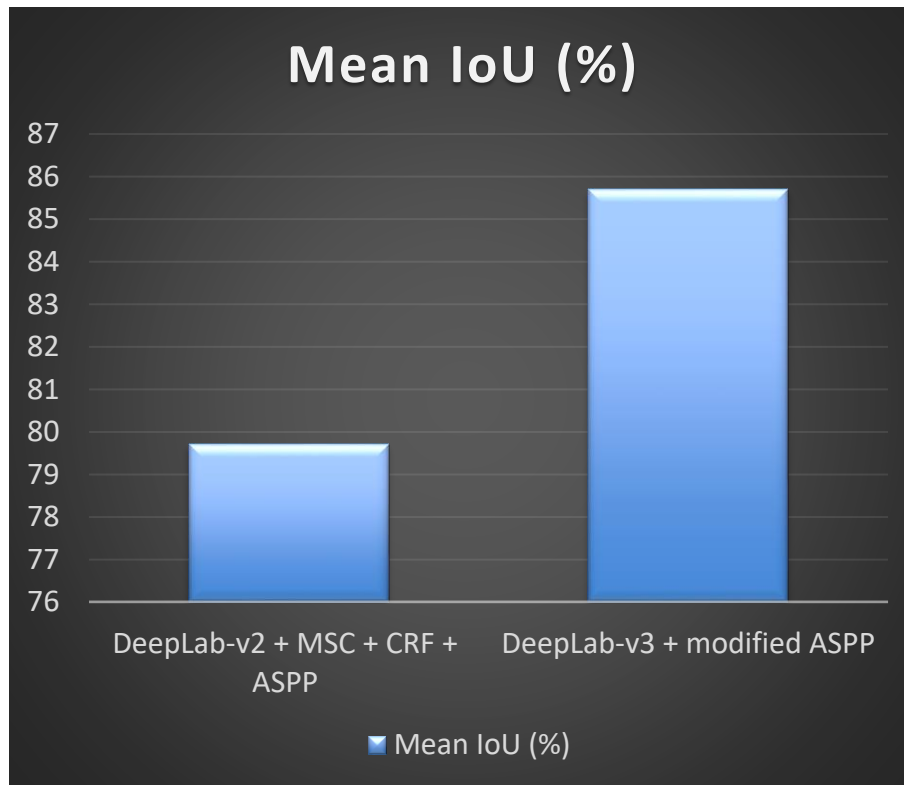
(Rethinking Atrous Convolution for Semantic Image Segmentation, Liang-Chieh Chen et al., 2017)

Modified Atrous Spatial Pyramid Pooling



(DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, Liang-Chieh Chen et al., 2017)

DeepLab-v3



(DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, Liang-Chieh Chen et al., 2017)