Welcome to Computer Vision

Computer Vision

Dr. Muhammad Tahir

DEPARTMENT OF COMPUTER SCIENCE,

FAST-NUCES, Peshawar

Course Details

LECTURES: Monday
& Wednesday

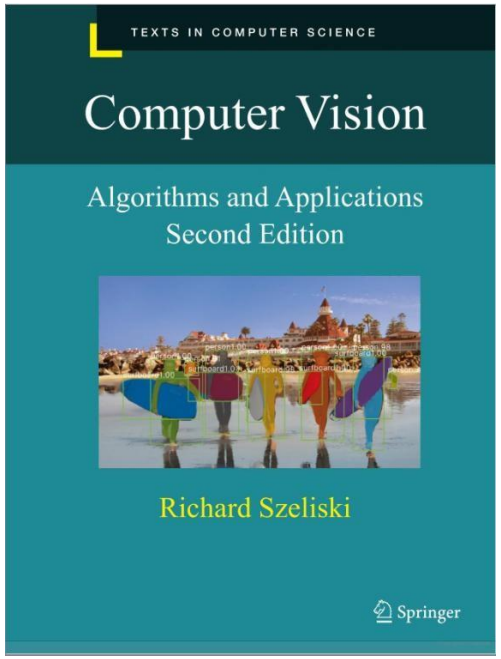
TIMINGS:

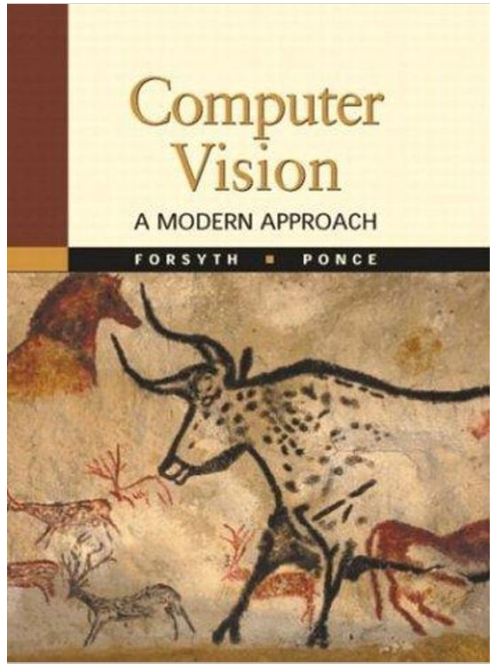
9:30 am – 11:00 am

MY OFFICE:

OFFICE HOURS:

EMAIL: m.tahir@nu.edu.pk





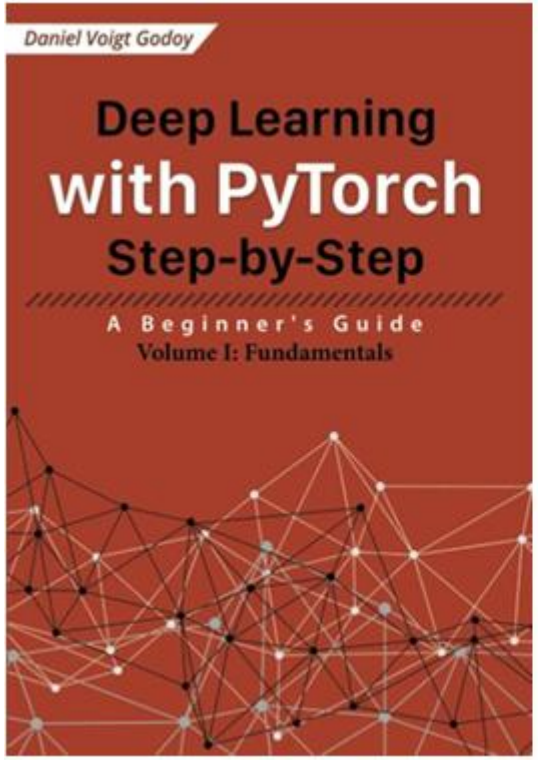
References

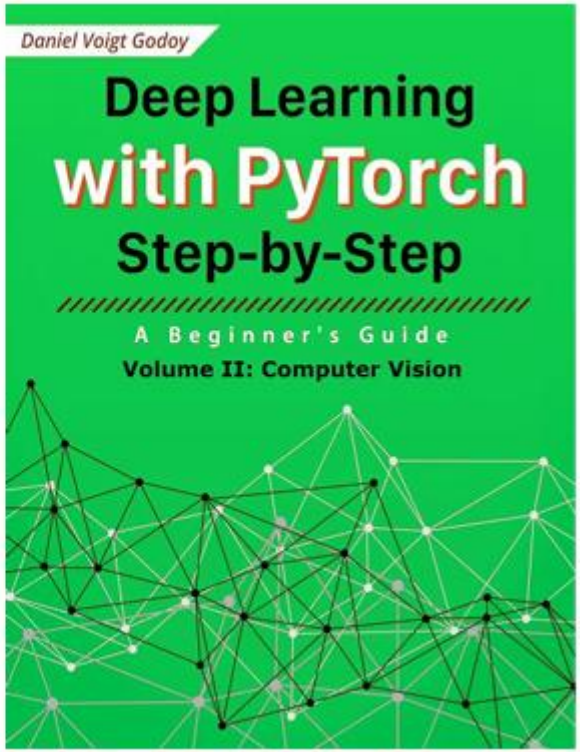
The material in these slides are based on:

1. Rick Szeliski's book: Computer Vision: Algorithms and Applications
2. Forsythe and Ponce: Computer Vision: A Modern Approach

Recommended Books

Deep Learning with PyTorch Step-by-Step by Daniel Voigt Godoy





Course Learning Outcomes

No	CLO	Domain	Taxonomy Level	PLO
1	Understand the view geometry concepts, multi-scale representation, edge detection and detection of other primitives, stereo, motion and object recognition.	Cognitive		
2	Assess which methods to use for solving a given problem, and analyse the accuracy of the methods Skills	Cognitive		
3	Apply appropriate image processing methods for image filtering, image restoration, image reconstruction, segmentation, classification and representation	Cognitive		

Outline

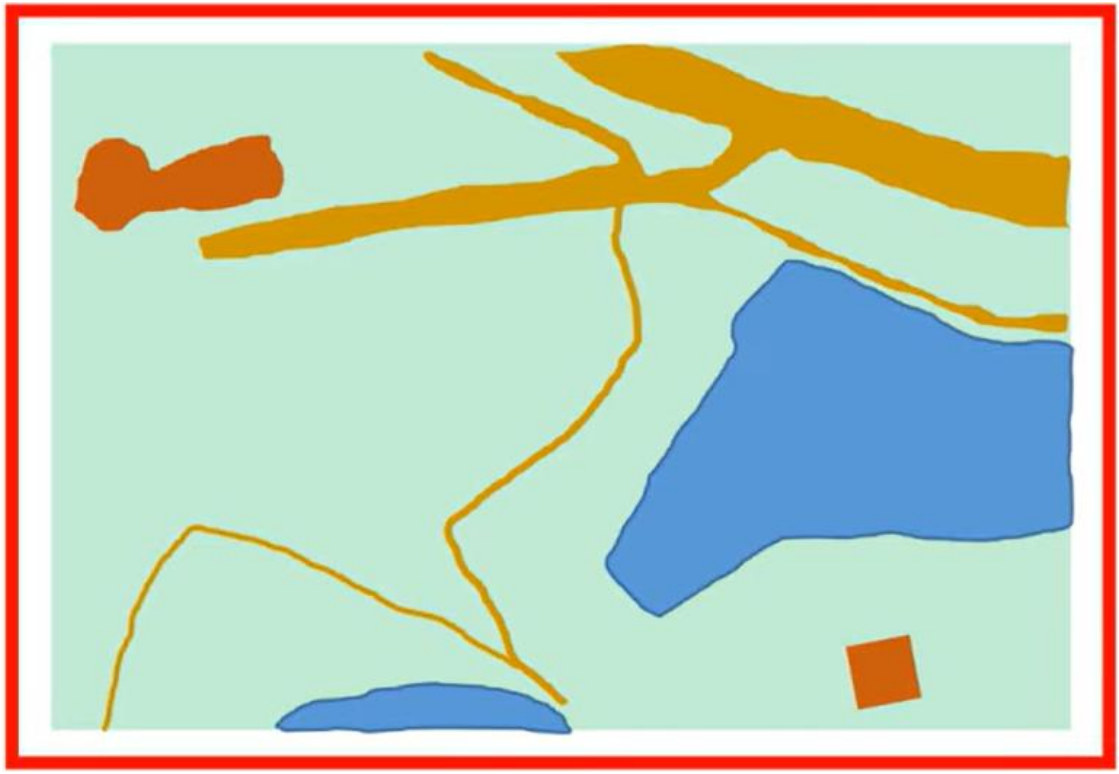
Image Segmentation – Unet Part I

Segmentation

Satellite Image

Segmentation

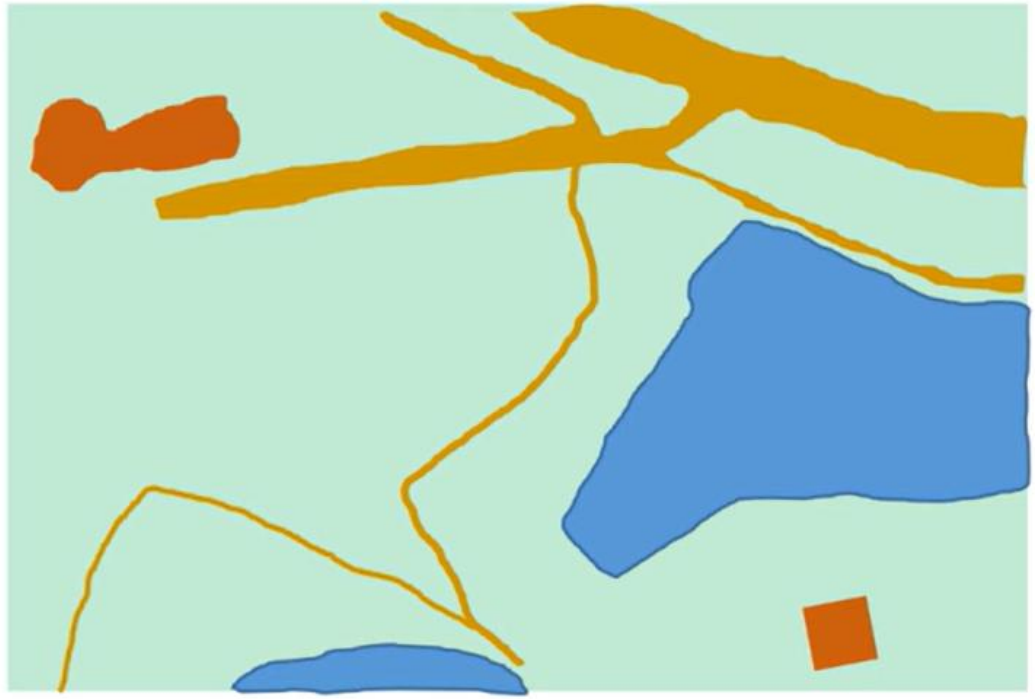
Satellite Image



A Segmented Image (mask)

Segmentation

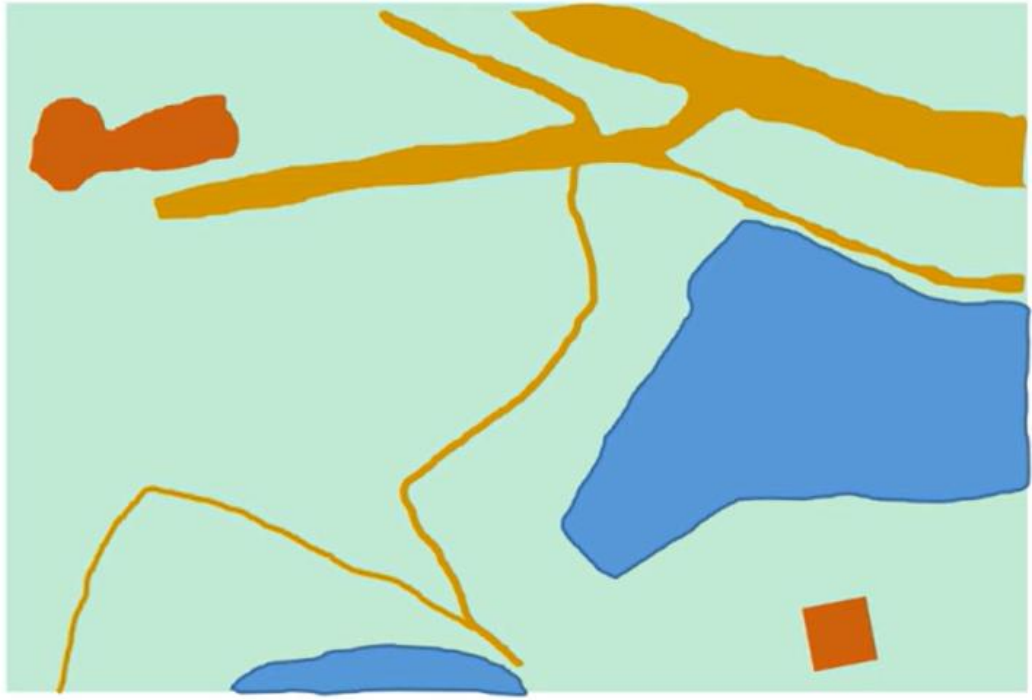
Satellite Image



A Segmented Image (mask)

Segmentation

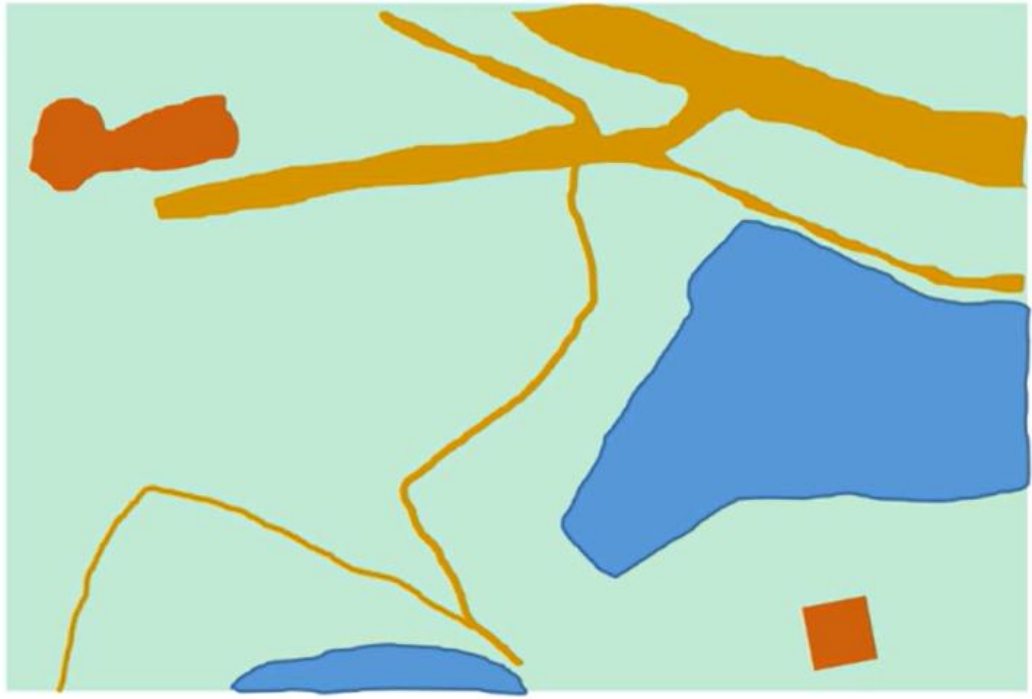
Satellite Image



A Segmented Image (mask)

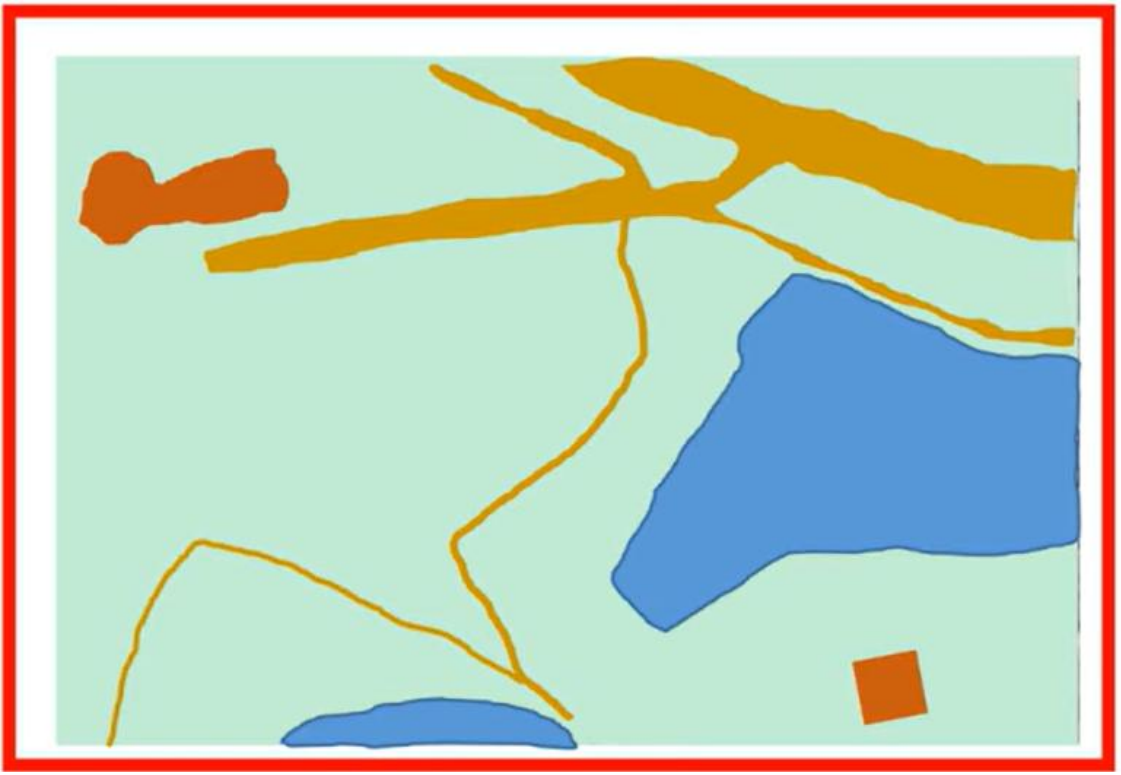
Segmentation

Satellite Image

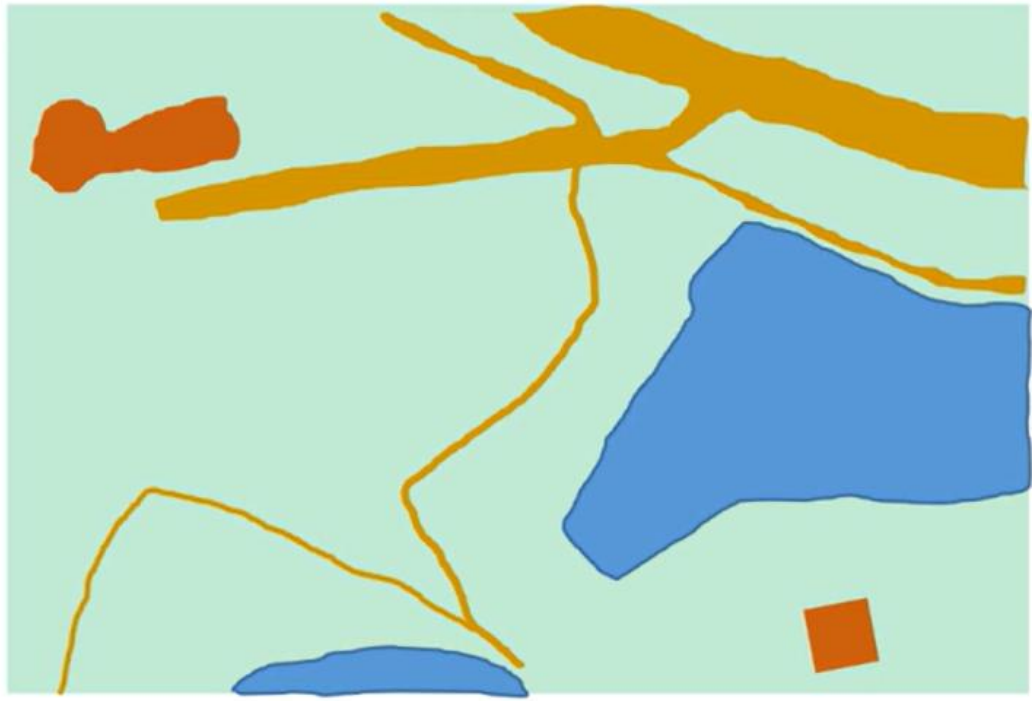


A Segmented Image (mask)

Segmentation



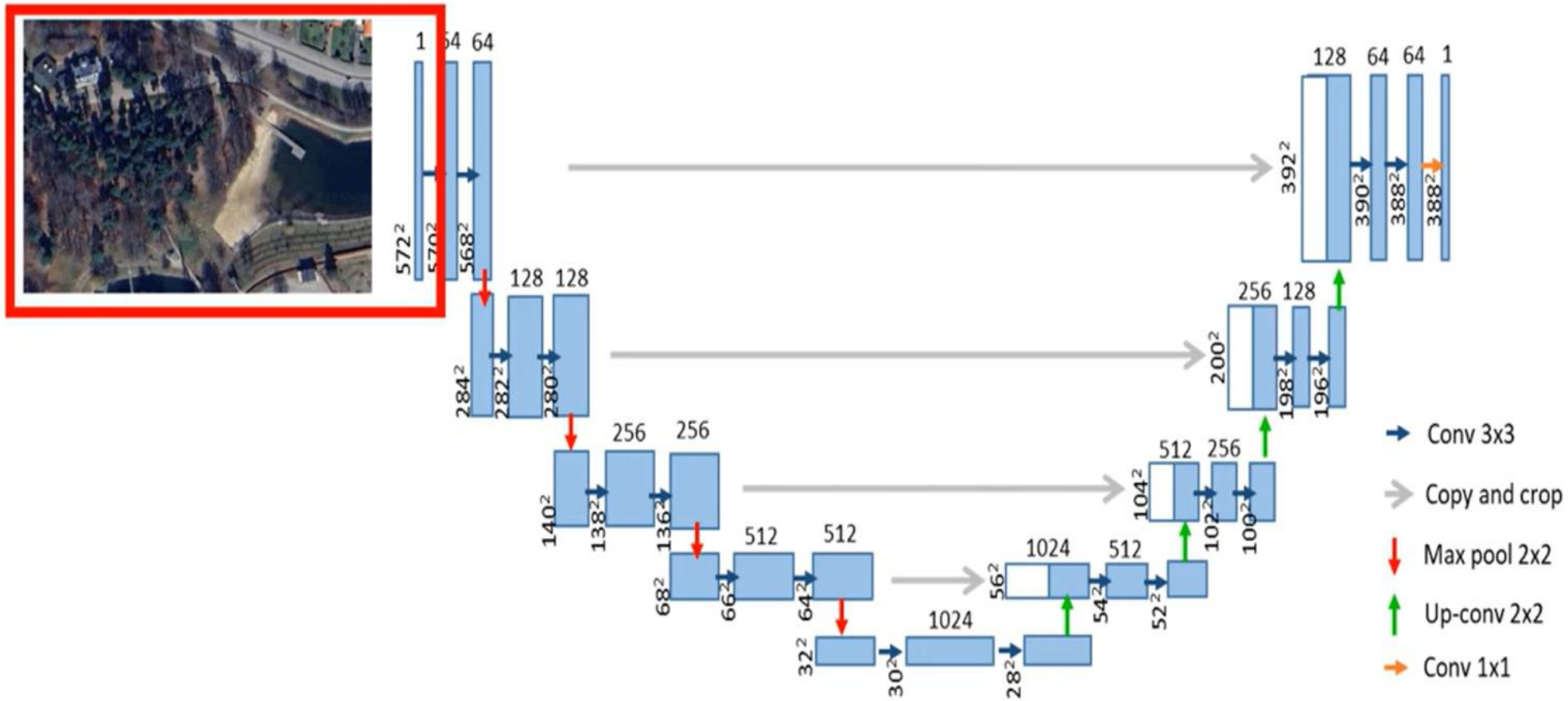
Satellite Image



A Segmented Image (mask)

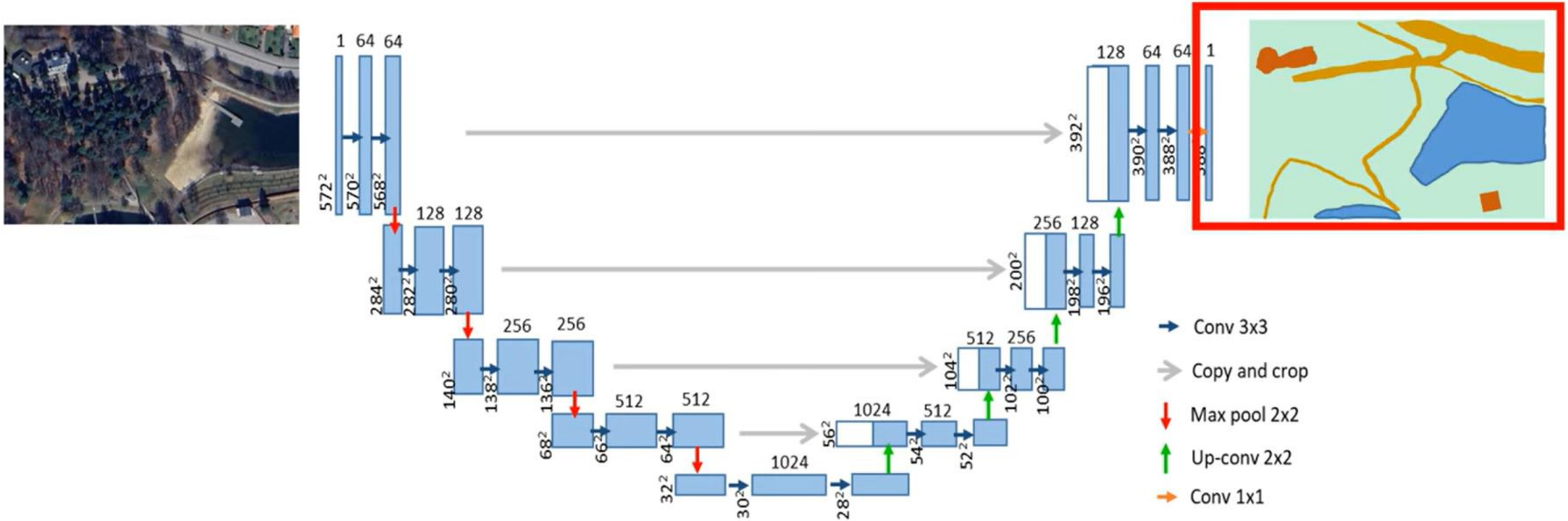
Segmentation

A simpler way would be to input the image into a trained U-Net



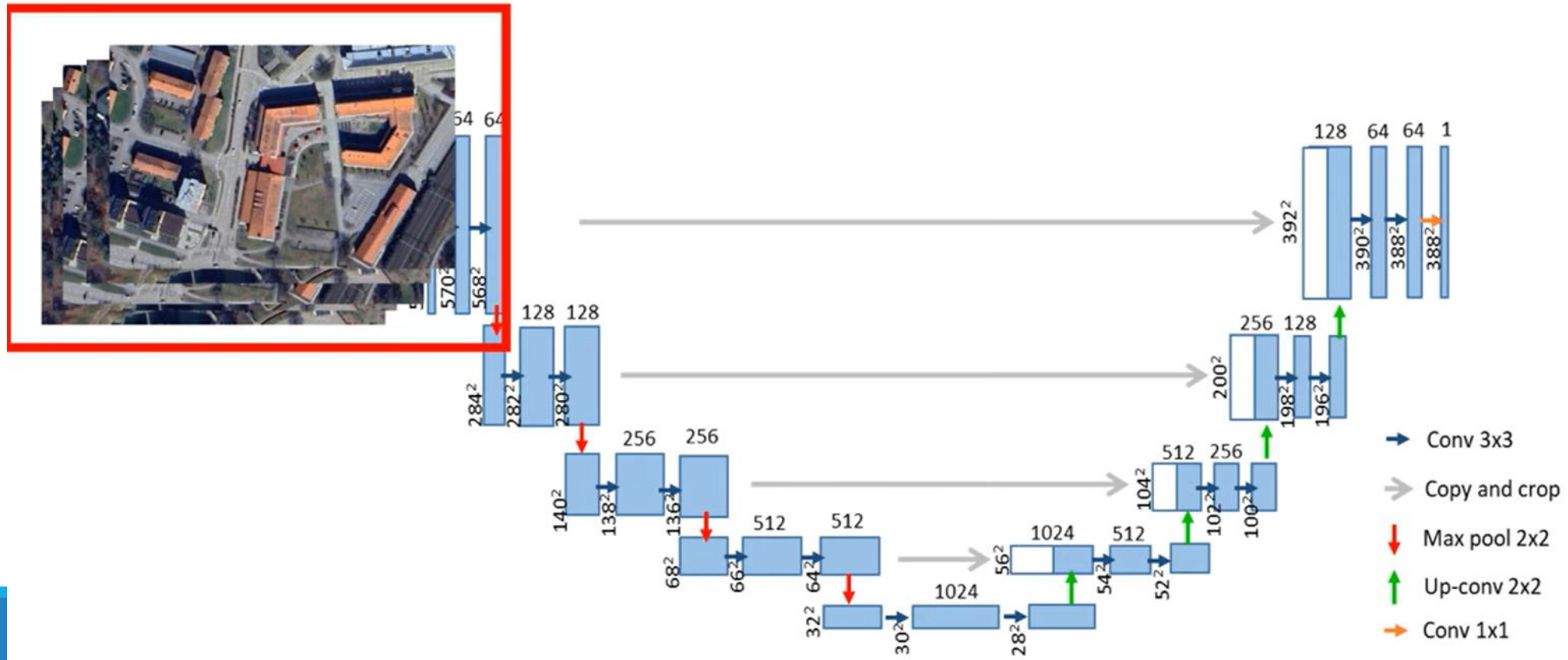
Segmentation

And let the U-Net create a segmented image automatically.



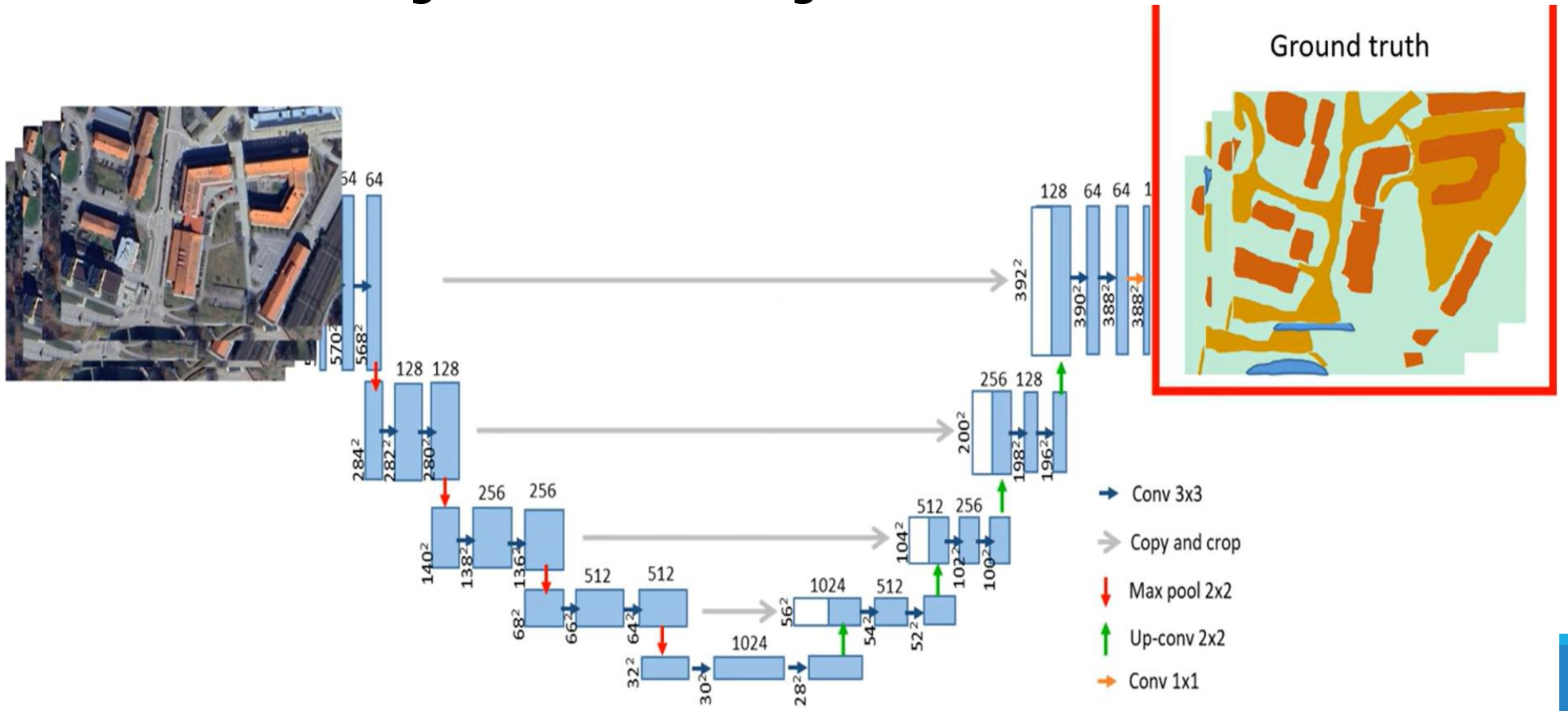
Segmentation

To create a U-Net that can perform segmentations like this, we first need to train the network on thousands of satellite images



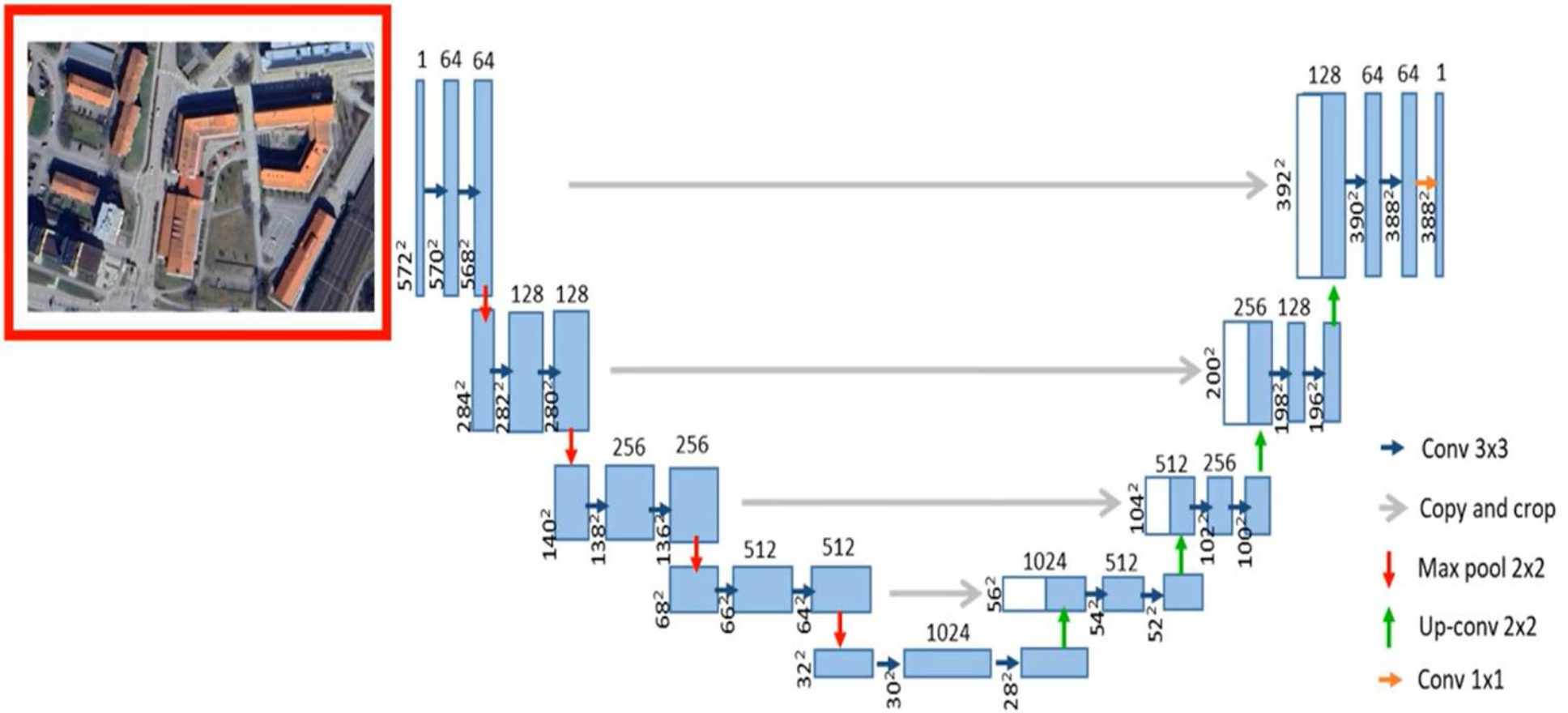
Segmentation

So that it learns to convert the satellite image into segmented images based on hand-made ground truth images



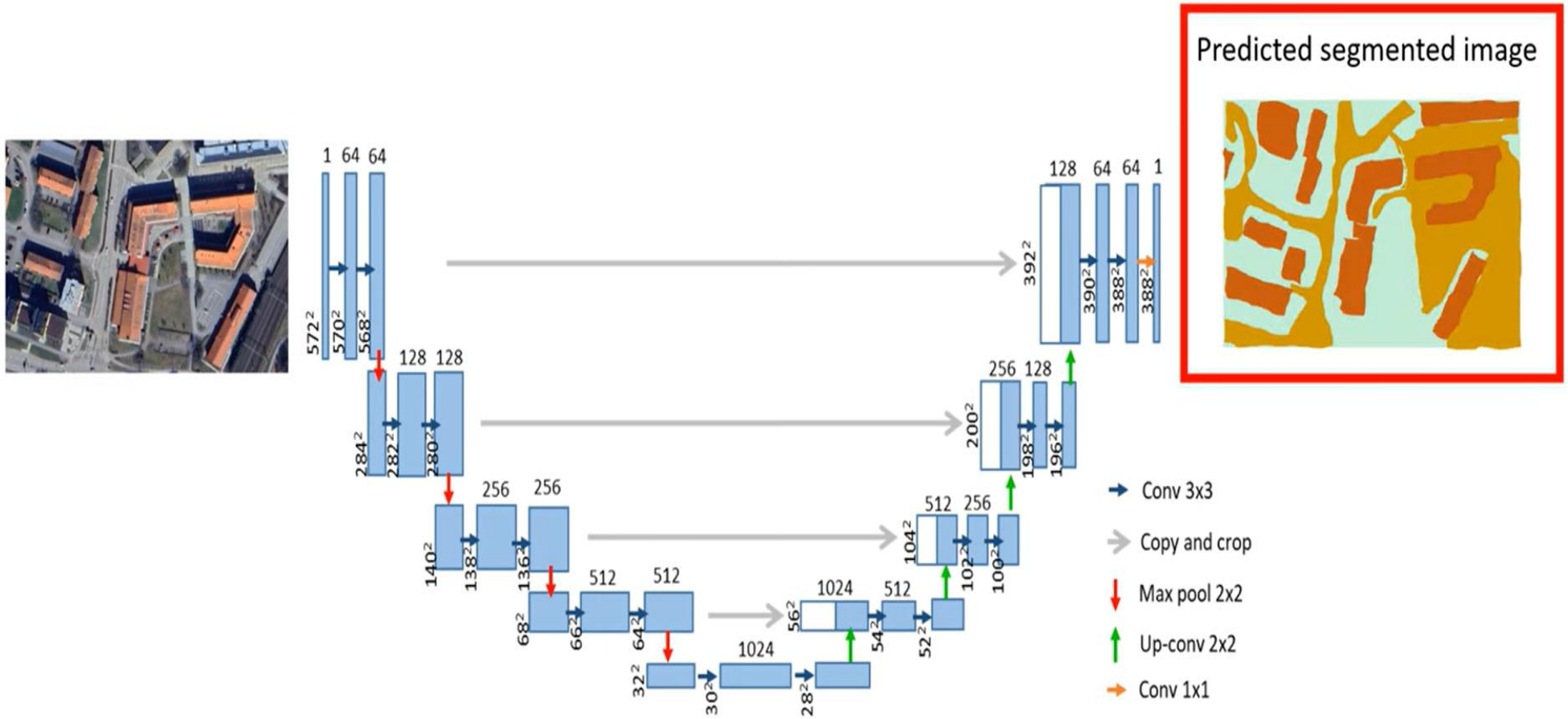
Segmentation

Once the network has been trained, we can input a new satellite image



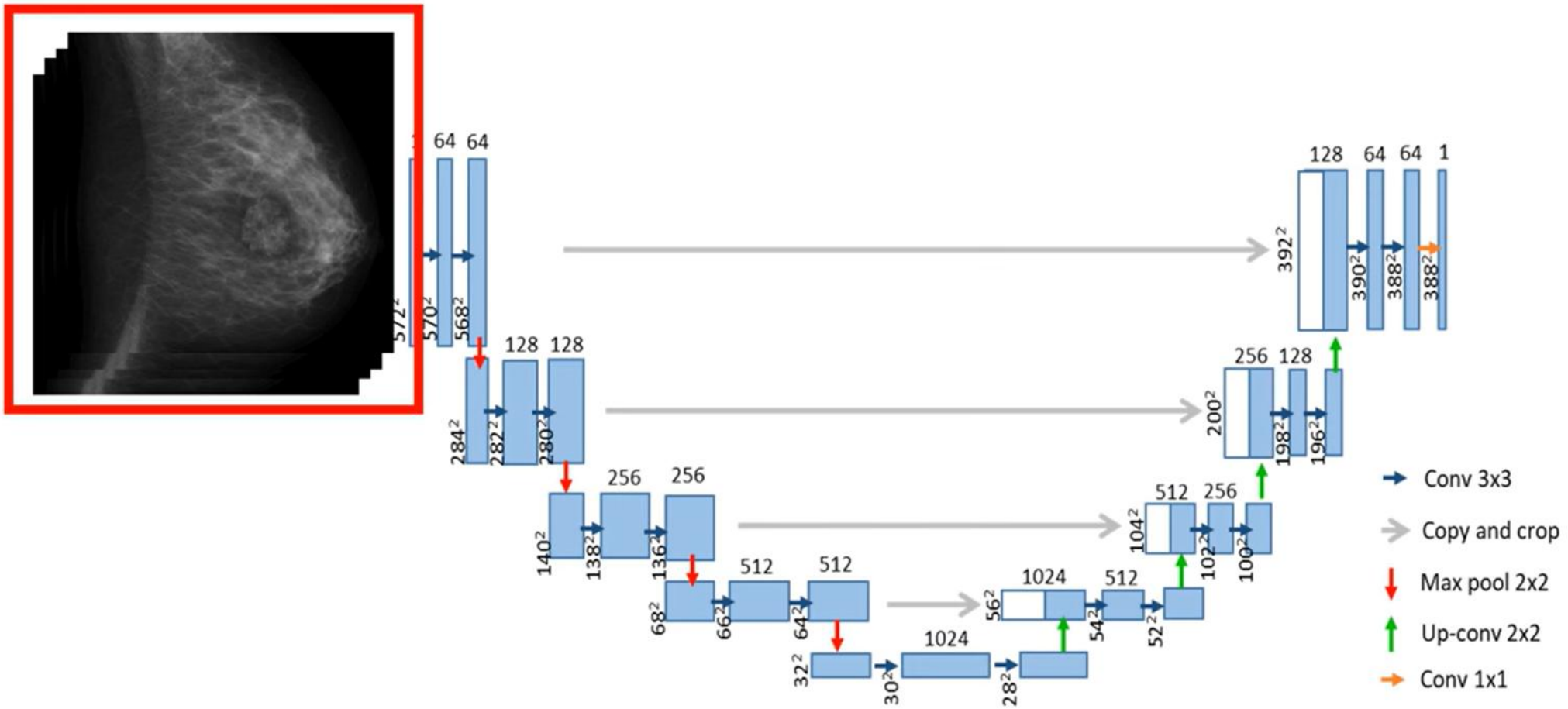
Segmentation

And let the network predict the corresponding mask



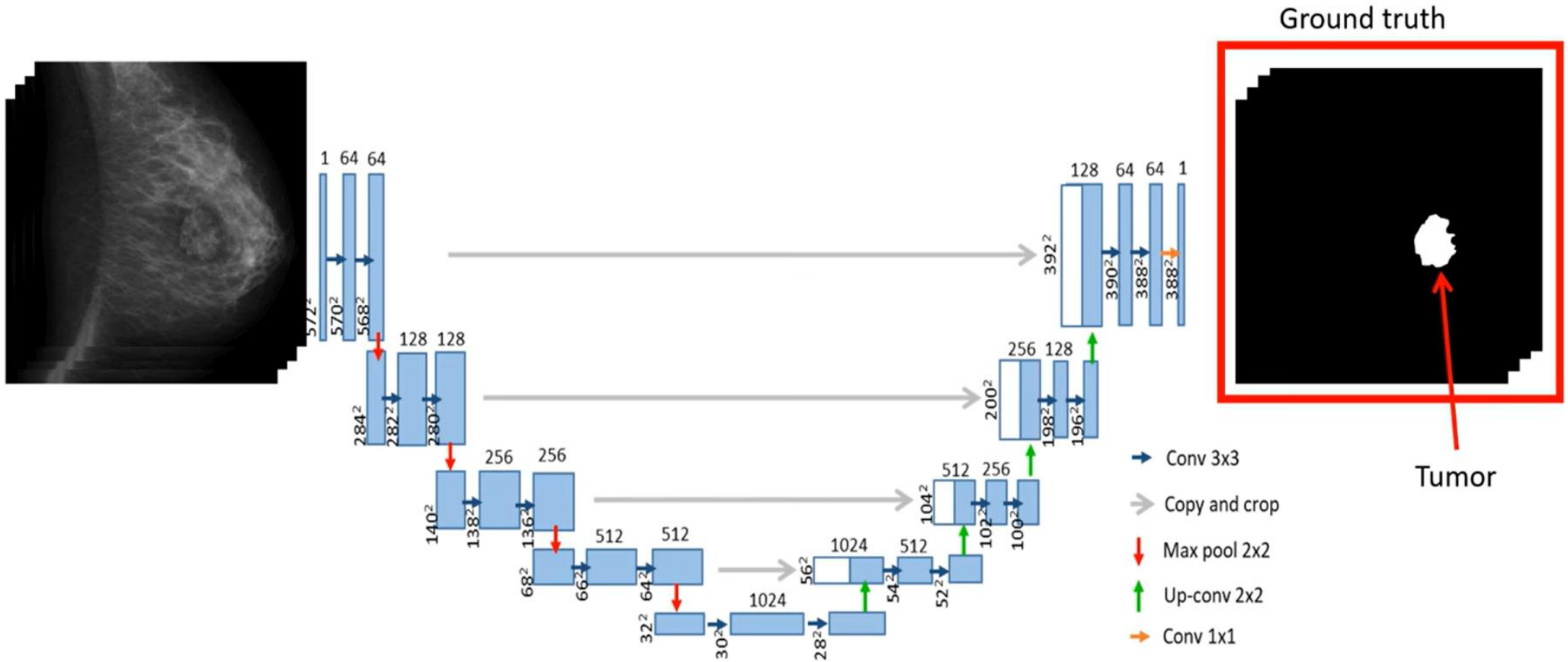
Segmentation

Similarly, we can train the network on X-ray images



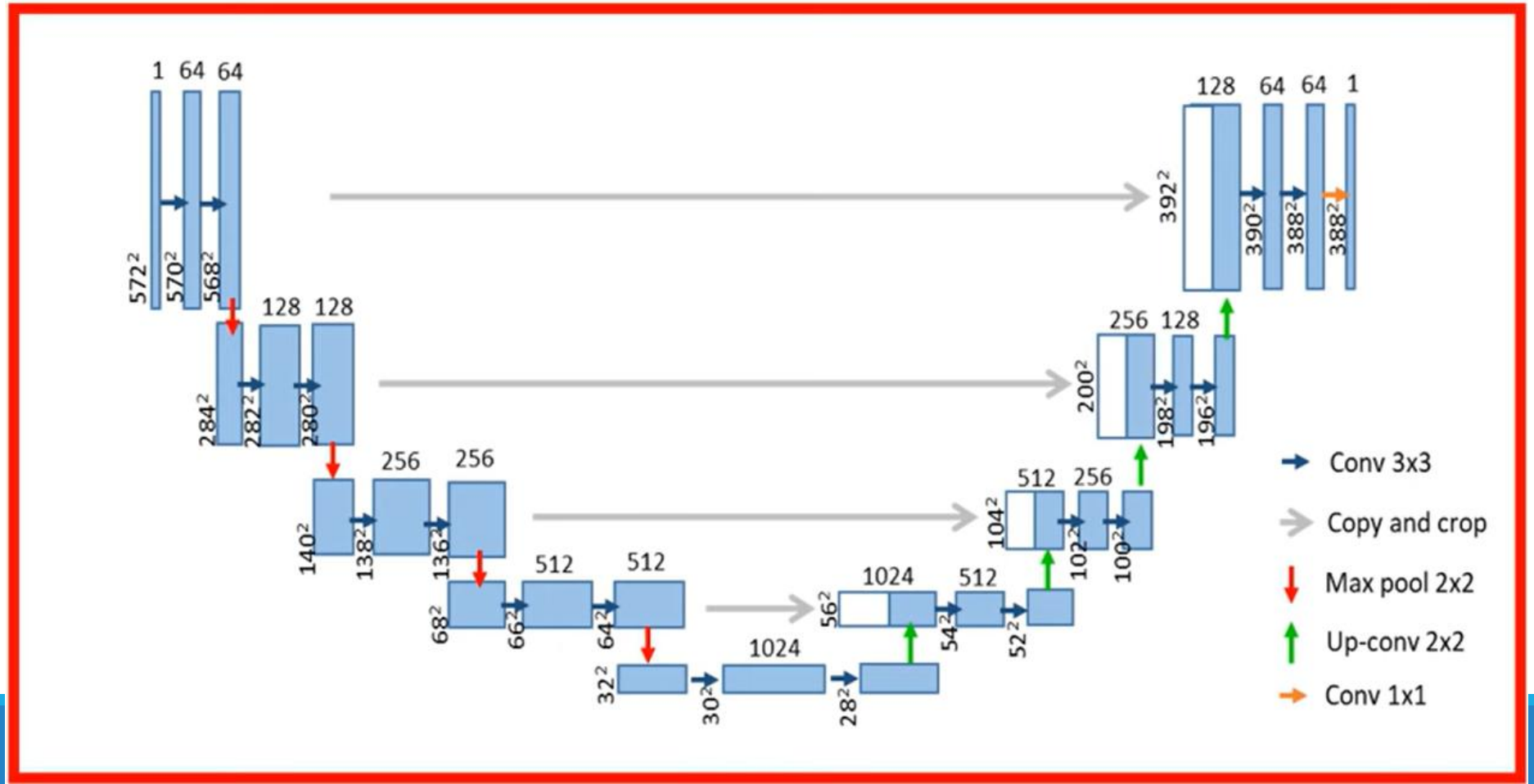
Segmentation

Which have been segmented by a specialized medical doctor who knows how to identify tumors in such images



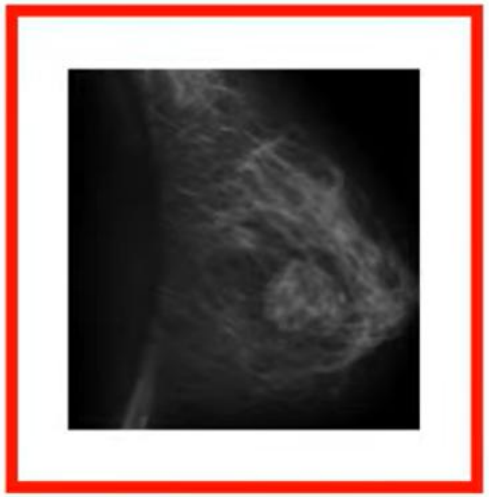
Segmentation

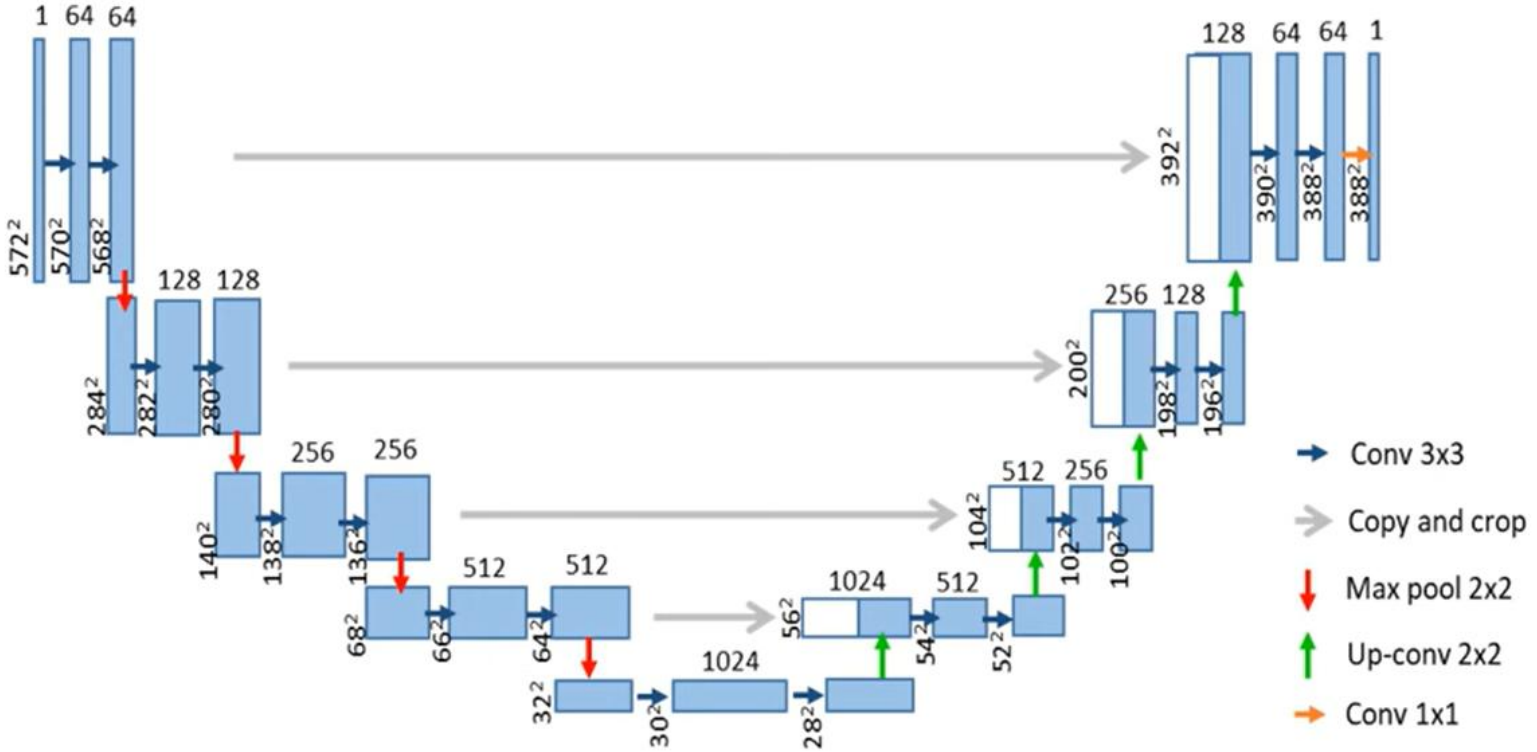
Once we have trained such a network



Segmentation

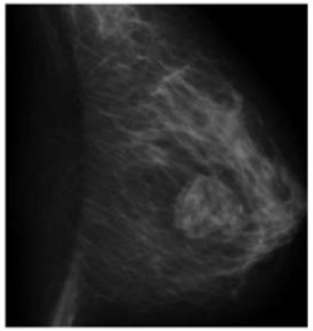
We can input a new X-ray image into the network

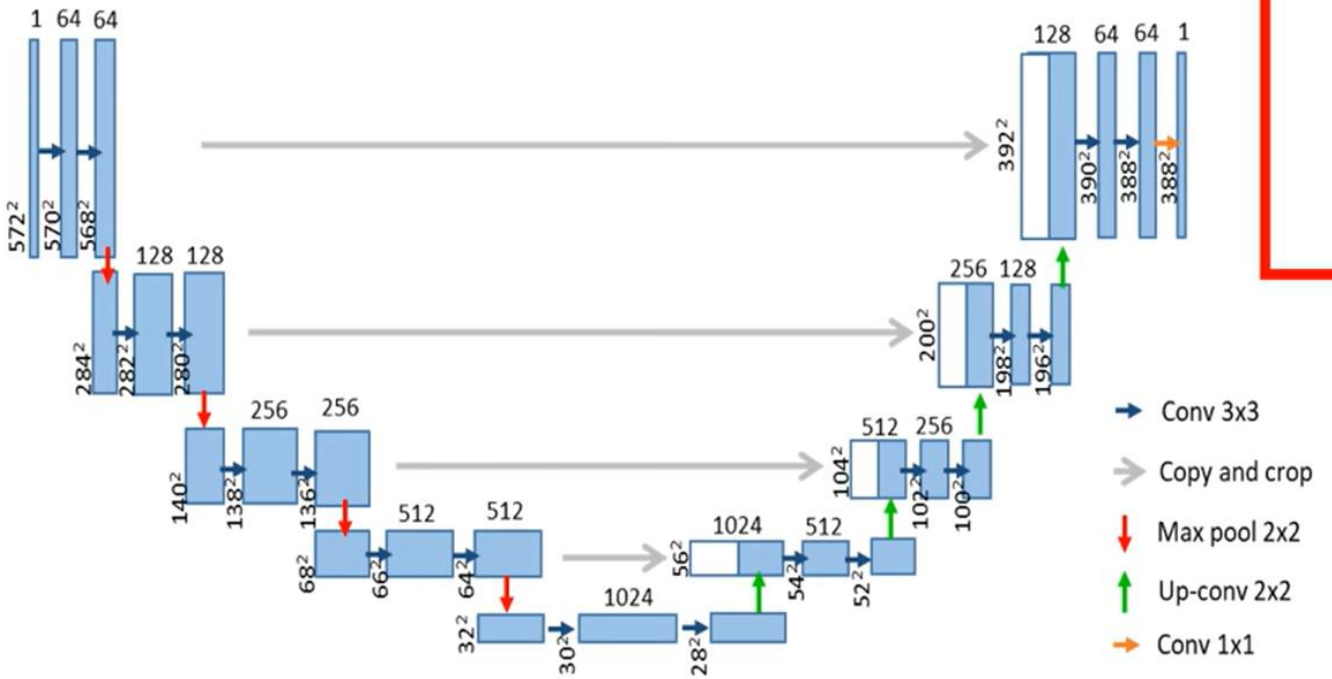


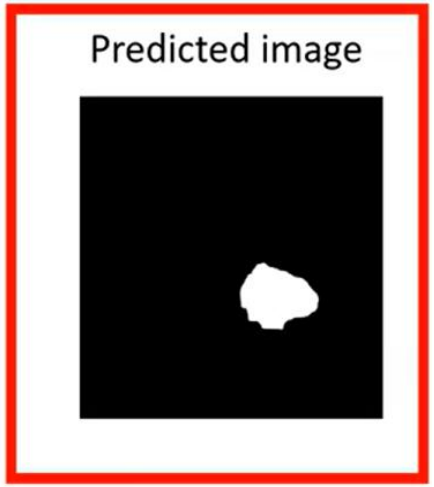


Segmentation

And let the network identify possible tumors and also predict their location and shape





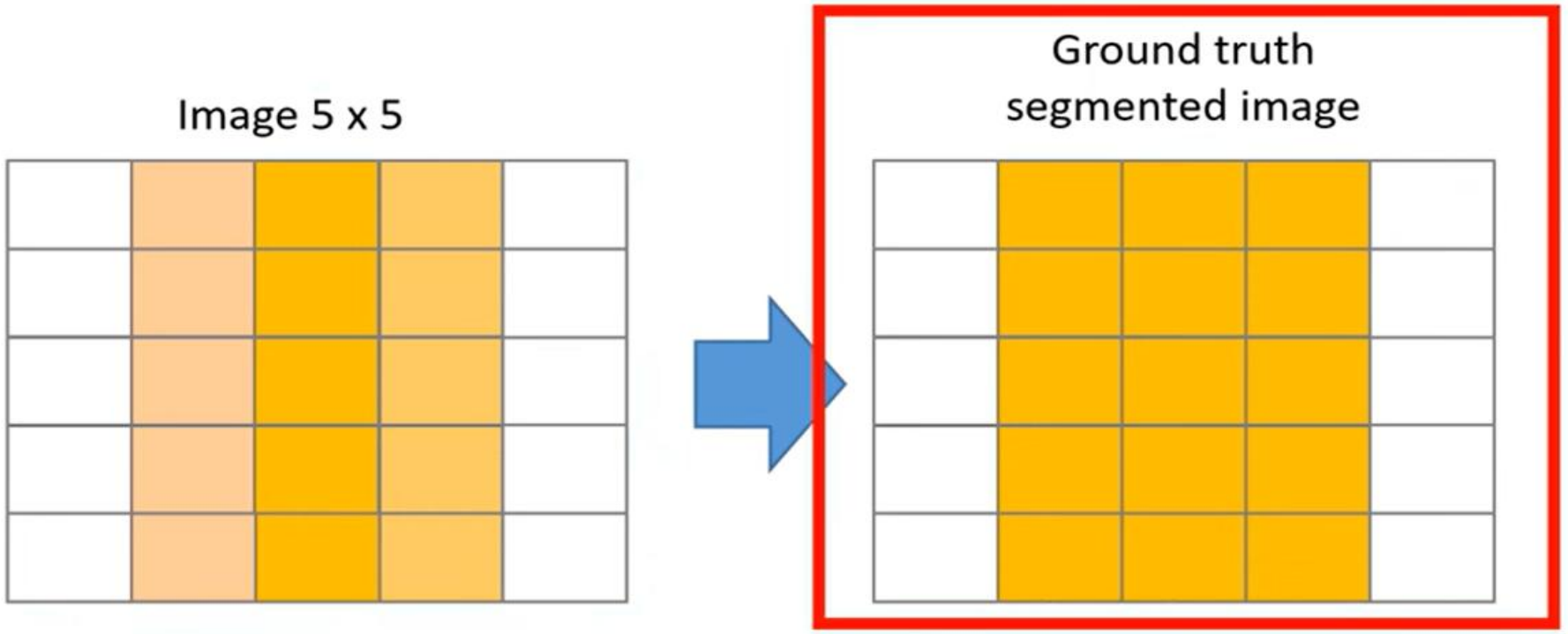


How a U-Net Works?

To explain how a U-Net works, Let's consider a simple image with 25 pixels

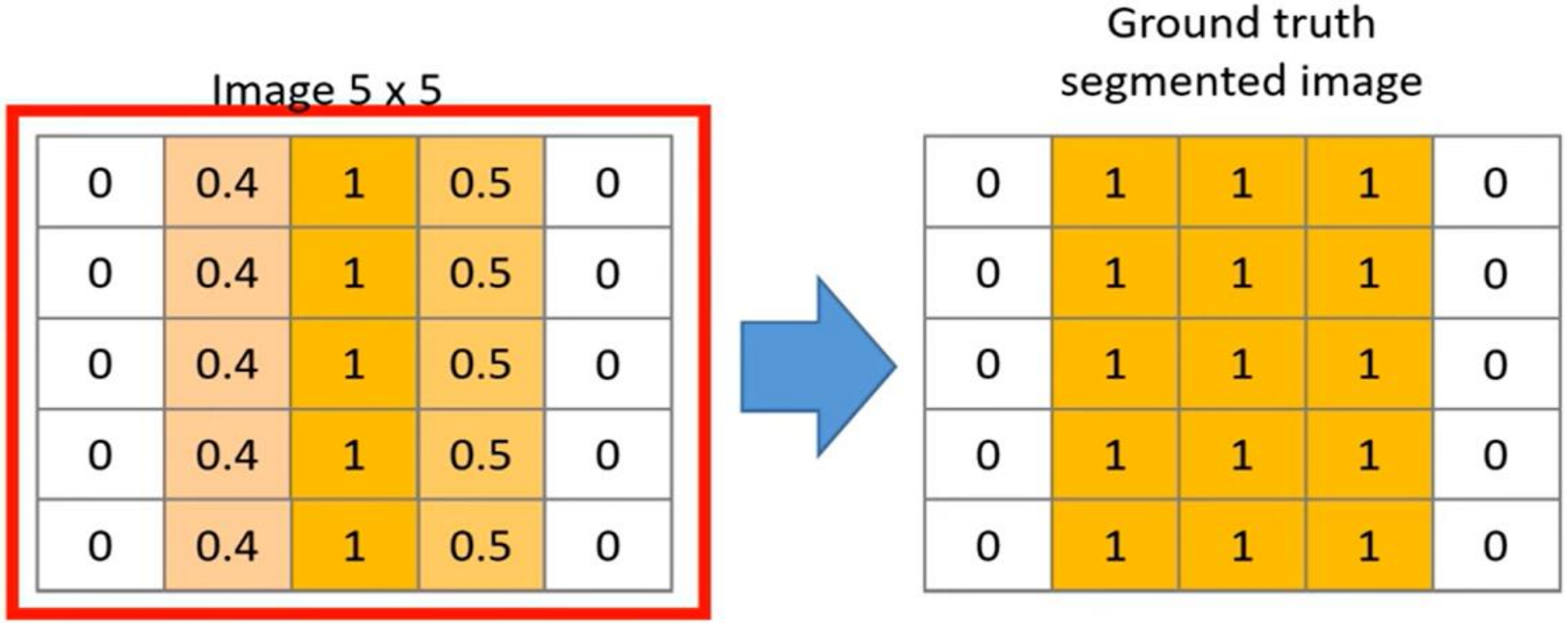
How a U-Net Works?

And train a network to recreate this ground truth mask



How a U-Net Works?

- Note that an image is just a matrix with numbers.
- The numbers have been normalized here.



How a U-Net Works?

Where **ones** represent the **brightest** color

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Ground truth segmented image

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

How a U-Net Works?

And **zero** represents the white **background**

Image 5 x 5

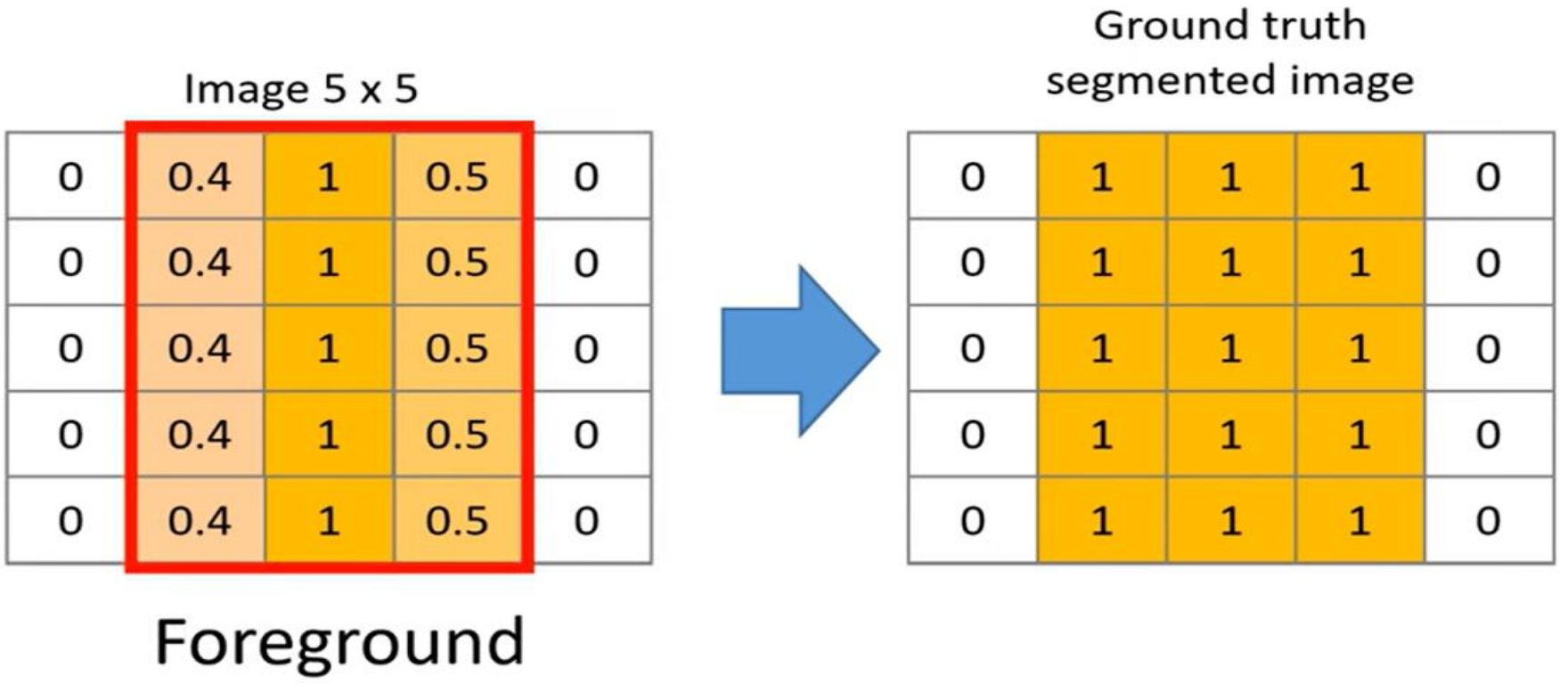
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Ground truth segmented image

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

How a U-Net Works?

Since these pixels do not belong to the background, they are here defined as belonging to the **foreground**.

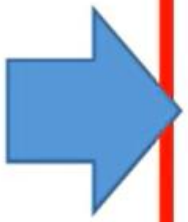


How a U-Net Works?

The **ground truth image** only includes **ones** and **zeros**

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0



Ground truth segmented image

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

How a U-Net Works?

Where the **ones** represent the foreground, such as a tumor

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Ground truth segmented image

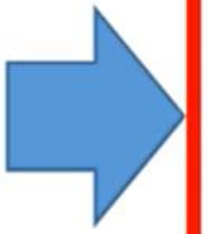
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

How a U-Net Works?

Whereas the **zeros** represent the **background**

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0



Ground truth segmented image

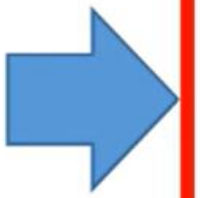
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

How a U-Net Works?

If we want to classify several different types of objects as foreground, we assign each object type a unique integer in the ground truth mask, for example, 1, 2, or 3

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

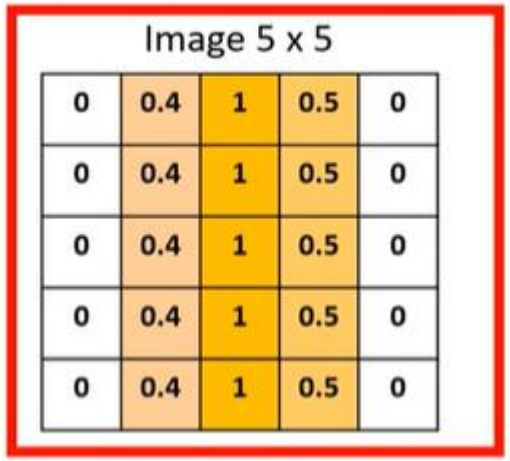


Ground truth segmented image

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

How a U-Net Works?

We will now use this image to illustrate the basic steps in how a U-Net computes a segmented image



How a U-Net Works?

- We will first use convolution to create a feature map, which extracts local features in the image.
- This convolution procedure is the same as used in CNNs

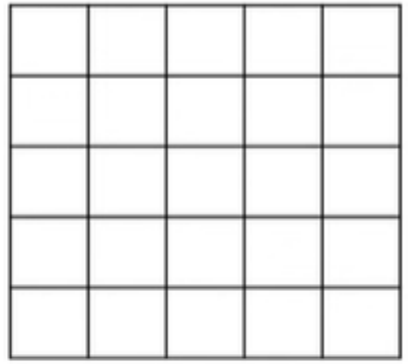
How a U-Net Works?

- The following filter will be used in this example.
- This filter has four weights that will be optimized during the training process.
- For this simple example, the weights were optimized to the following values.

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map



Filter = Kernel

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

The filter is then placed on top of the image where we perform a dot product.

Image 5 x 5

1.45	0.31		0.5	0
0.54	-0.04		0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Which means that we multiply the values in the overlapping elements as given:

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map

$$0 \cdot 1.45 + 0.4 \cdot 0.31 + 0 \cdot 0.54 + 0.4 \cdot (-0.04) - 0.11 = -0.002$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map

$$0 \cdot 1.45 + 0.4 \cdot 0.31 + 0 \cdot 0.54 + 0.4 \cdot (-0.04) - 0.11 = -0.002$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map

$$0 \cdot 1.45 + 0.4 \cdot 0.31 + 0 \cdot 0.54 + 0.4 \cdot (-0.04) - 0.11 = -0.002$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map

$$0 \cdot 1.45 + 0.4 \cdot 0.31 + 0 \cdot 0.54 + 0.4 \cdot (-0.04) - 0.11 = -0.002$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

The bias is also optimized during the training process

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map

Bias

$$0 \cdot 1.45 + 0.4 \cdot 0.31 + 0 \cdot 0.54 + 0.4 \cdot (-0.04) - 0.11 = -0.002$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

The terms are added, and the sum is input into the feature map

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map

0				

$$0 \cdot 1.45 + 0.4 \cdot 0.31 + 0 \cdot 0.54 + 0.4 \cdot (-0.04) - 0.11 = -0.002$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Then we slide the filter one step to the right

Image 5 x 5

0	0.4	1	.5	0
0	0.4	1	.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

1.45 0.31

0.54 -0.04

Feature map

0				

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

The stride = 1

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map

0				

Stride = 1

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Then we do the same calculations,

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map

0				

$$0.4 \cdot 1.45 + 1 \cdot 0.31 + 0.4 \cdot 0.54 + 1 \cdot (-0.04) - 0.11 = 0.96$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

and the final value is input into the feature map

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

1.45 0.31

0.54 -0.04

Feature map

0	0.96			

$$0.4 \cdot 1.45 + 1 \cdot 0.31 + 0.4 \cdot 0.54 + 1 \cdot (-0.04) - 0.11 = 0.96$$

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

1.45 0.31

0.54 -0.04

Feature map

0	0.96	2.0		

$$1 \cdot 1.45 + 0.5 \cdot 0.31 + 1 \cdot 0.54 + 0.5 \cdot (-0.04) - 0.11 = 2.015$$

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

1.45	0.31
0.54	-0.04

Filter

Feature map

0	0.96	2.0	0.89	

$$0.5 \cdot 1.45 + 0 \cdot 0.31 + 0.5 \cdot 0.54 + 0 \cdot (-0.04) - 0.11 = 0.885$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

In this case, we like the feature map to be of the same size

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

1.45 0.31

0.54 -0.04

Feature map

0	0.96	2.0	0.89	

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

As our image

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

1.45 0.31

0.54 -0.04

Feature map

0	0.96	2.0	0.89	

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Therefore, we will use padding in this case

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

1.45 0.31

0.54 -0.04

Padding

Feature map

0	0.96	2.0	0.89	

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

This will allow the filter to slide for one additional step

Image 5 x 5

0	0.4	1	0.5	0 1.45	0 0.31
0	0.4	1	0.5	0 0.54	0 -0.04
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

And do the calculations

Image 5 x 5

0	0.4	1	0.5	0 (1.45)	0 (0.31)
0	0.4	1	0.5	0 (0.54)	0 (-0.04)
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11

$$0 \cdot 1.45 + 0 \cdot 0.31 + 0 \cdot 0.54 + 0 \cdot (-0.04) - 0.11 = -0.11$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Then we continue to slide the filter until we have filled the whole feature map

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0				

$$0 \cdot 1.45 + 0.4 \cdot 0.31 + 0 \cdot 0.54 + 0.4 \cdot (-0.04) - 0.11 = -0.002$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

1.45 0.31

0.54 -0.04

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96			

$$0.4 \cdot 1.45 + 1 \cdot 0.31 + 0.4 \cdot 0.54 + 1 \cdot (-0.04) - 0.11 = 0.96$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

1.45 0.31

0.54 -0.04

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0		

$$1 \cdot 1.45 + 0.5 \cdot 0.31 + 1 \cdot 0.54 + 0.5 \cdot (-0.04) - 0.11 = 2.015$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	

$$0.5 \cdot 1.45 + 0 \cdot 0.31 + 0.5 \cdot 0.54 + 0 \cdot (-0.04) - 0.11 = 0.885$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Image 5 x 5

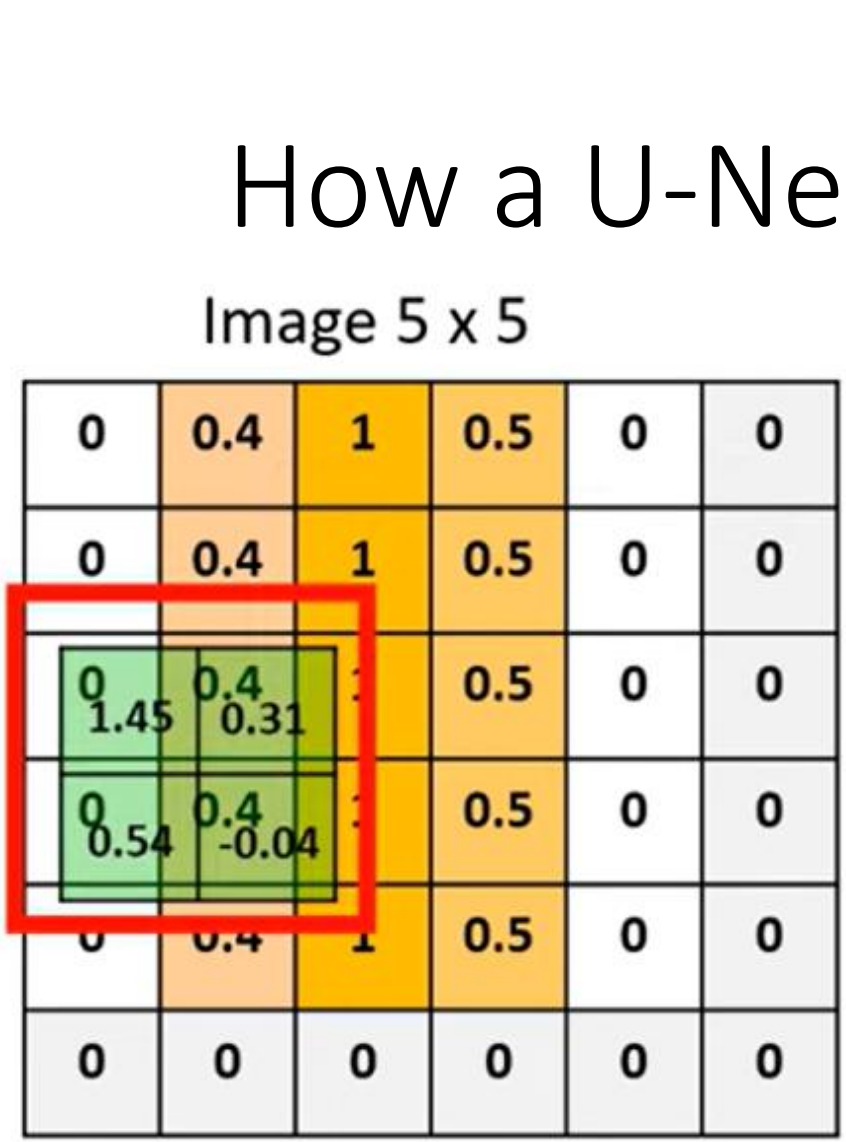
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

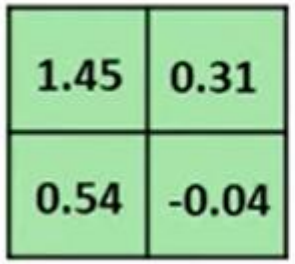
Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11

$$0 \cdot 1.45 + 0 \cdot 0.31 + 0 \cdot 0.54 + 0 \cdot (-0.04) - 0.11 = -0.11$$

How a U-Net Works?





Filter

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0				

$$0 \cdot 1.45 + 0.4 \cdot 0.31 + 0 \cdot 0.54 + 0.4 \cdot (-0.04) - 0.11 = -0.002$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Once the feature map is filled

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Activation function should be applied such as the ReLU function

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

ReLU

$$f(x) = \max(0, x)$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

The ReLU function takes a value, and returns the maximum value between zero and the value we input

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

ReLU

$$f(x) = \max(0, x)$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

If this value is input to the ReLU function, zero will be returned because zero is the maximum value out of negative 0.11 and zero.

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

ReLU

$$f(x) = \max(0, x)$$

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

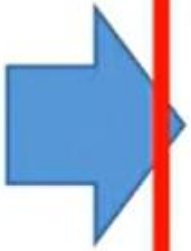
This will set all negative values to zero and positive values unchanged.

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11



Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

How a U-Net Works?

1.45	0.31
0.54	-0.04

Filter

Next, 2×2 **MAX** pooling with $\boldsymbol{stride} = \mathbf{2}$ is performed

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

How a U-Net Works?

The output is input into **pooled feature map**

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

Feature map (ReLU)

Pooled feature map

0.96		

How a U-Net Works?

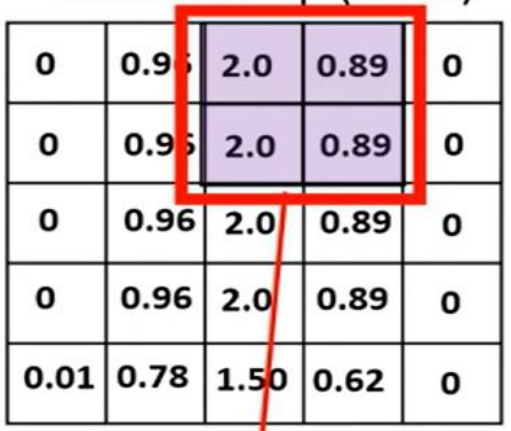
Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

Feature map (ReLU)



Pooled feature map

0.96	2.0	

How a U-Net Works?

Padding can be done to use all values in the feature map

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

Feature map (ReLU)

0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0.01	0.78	1.50	0.62	0	0
0	0	0	0	0	0

Padding

Pooled feature map

0.96	2.0	0

How a U-Net Works?

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

Feature map (ReLU)

0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0.01	0.78	1.50	0.62	0	0
0	0	0	0	0	0

Pooled feature map

0.96	2.0	0
0.96		

How a U-Net Works?

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

Feature map (ReLU)

0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0.01	0.78	1.50	0.62	0	0
0	0	0	0	0	0

Pooled feature map

0.96	2.0	0
0.96	2.0	

How a U-Net Works?

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

Feature map (ReLU)

0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.8	0	0
0	0.96	2.0	0.8	0	0
0.01	0.78	1.50	0.62	0	0
0	0	0	0	0	0

Pooled feature map

0.96	2.0	0
0.96	2.0	0

How a U-Net Works?

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

Feature map (ReLU)

0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0.01	0.78	1.50	0.62	0	0
0	0	0	0	0	0

Pooled feature map

0.96	2.0	0
0.96	2.0	0
0.78		

How a U-Net Works?

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

Feature map (ReLU)

0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0.01	0.7	1.50	0.62	0	0
0	0	0	0	0	0

Pooled feature map

0.96	2.0	0
0.96	2.0	0
0.78	1.5	

How a U-Net Works?

Image 5 x 5

0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0.4	1	0.5	0	0
0	0	0	0	0	0

Feature map

0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0	0.96	2.0	0.89	-0.11
0.01	0.78	1.50	0.62	-0.11

Feature map (ReLU)

0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0	0.96	2.0	0.89	0	0
0.01	0.78	1.50	0.62	0	0
0	0	0	0	0	0

Pooled feature map

0.96	2.0	0
0.96	2.0	0
0.78	1.5	0

How a U-Net Works?

- The final pooled feature map is now complete
- Now **Upsampling** will be performed using nearest neighbor method

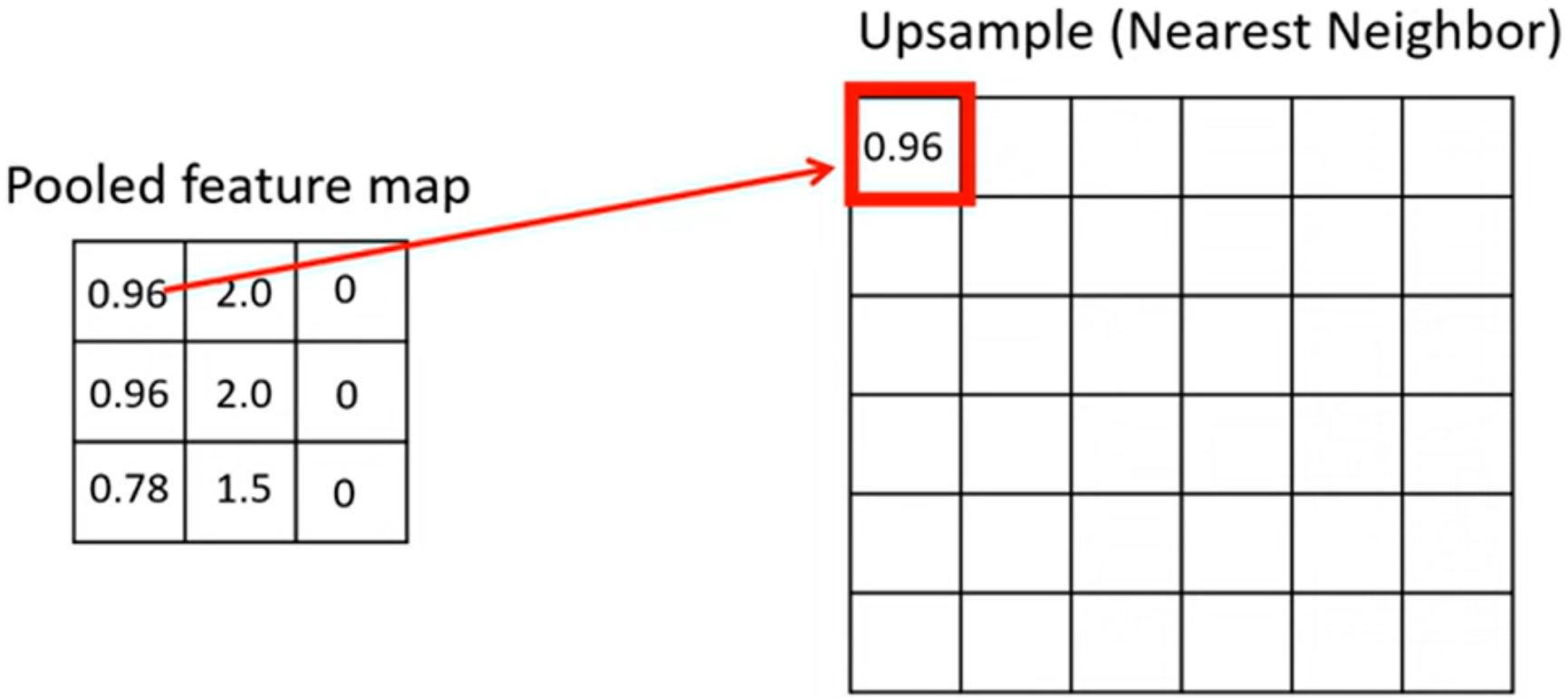
Pooled feature map

0.96	2.0	0
0.96	2.0	0
0.78	1.5	0

Upsample (Nearest Neighbor)

How a U-Net Works?

In the **nearest neighbor method**, the **first** value from the **pooled feature map** is input into the **target matrix**



How a U-Net Works?

And then copy this value to its nearest neighbors

Pooled feature map

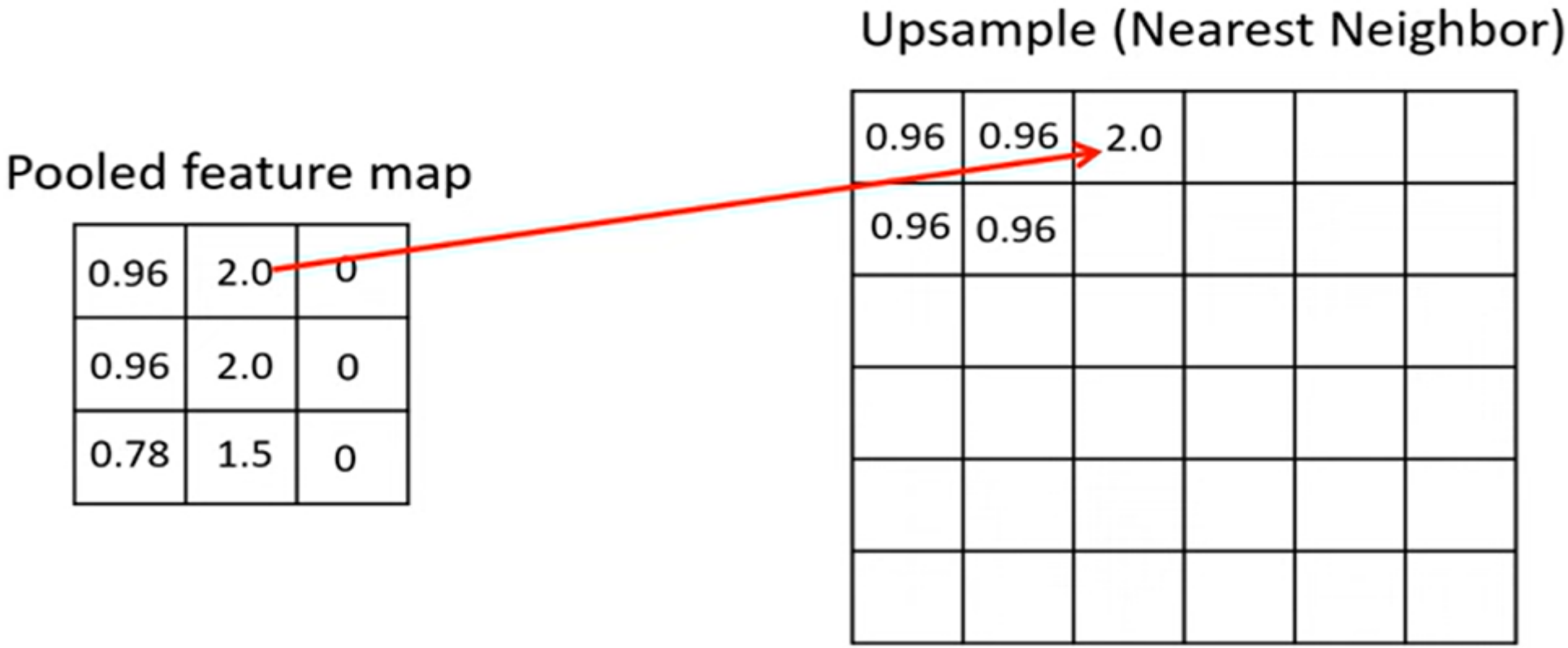
0.96	2.0	0
0.96	2.0	0
0.78	1.5	0

Upsample (Nearest Neighbor)

0.96	0.96				
0.96	0.96				

How a U-Net Works?

Then the next value is plugged in



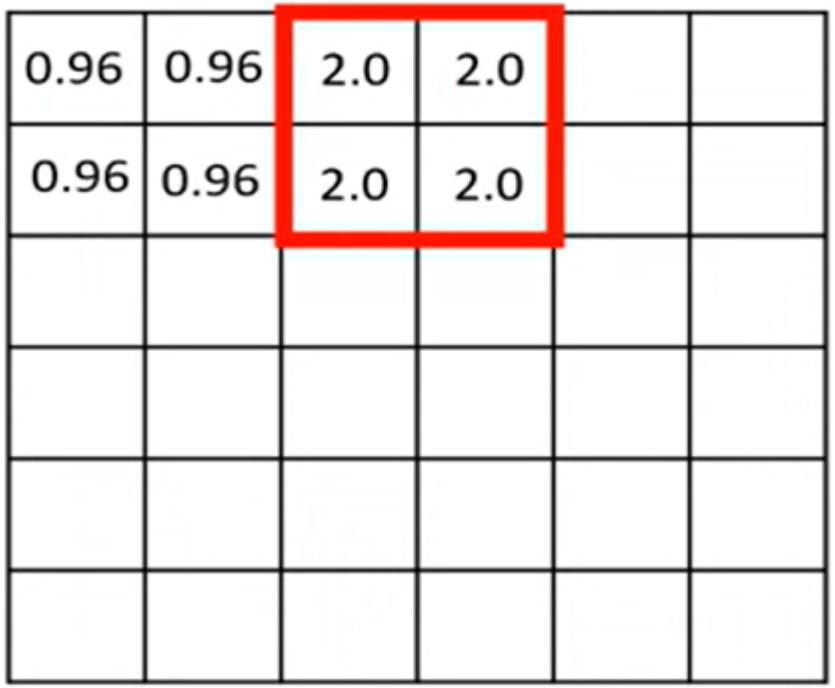
How a U-Net Works?

And copy its values

Pooled feature map

0.96	2.0	0
0.96	2.0	0
0.78	1.5	0

Upsample (Nearest Neighbor)



How a U-Net Works?

This is the final output

Pooled feature map

0.96	2.0	0
0.96	2.0	0
0.78	1.5	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0	0
0.96	0.96	2.0	2.0	0	0
0.96	0.96	2.0	2.0	0	0
0.96	0.96	2.0	2.0	0	0
0.78	0.78	1.5	1.5	0	0
0.78	0.78	1.5	1.5	0	0

How a U-Net Works?

We started with this image

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0	0
0.96	0.96	2.0	2.0	0	0
0.96	0.96	2.0	2.0	0	0
0.96	0.96	2.0	2.0	0	0
0.78	0.78	1.5	1.5	0	0
0.78	0.78	1.5	1.5	0	0

Pooled feature map

0.96	2.0	0
0.96	2.0	0
0.78	1.5	0

How a U-Net Works?

Created this feature map after applying the ReLU activation function

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0	0
0.96	0.96	2.0	2.0	0	0
0.96	0.96	2.0	2.0	0	0
0.96	0.96	2.0	2.0	0	0
0.78	0.78	1.5	1.5	0	0
0.78	0.78	1.5	1.5	0	0

Pooled feature map

0.96	2.0	0
0.96	2.0	0
0.78	1.5	0

How a U-Net Works?

Then we perform **downsampling** with MAX pooling, which **reduced** the size

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0	0
0.96	0.96	2.0	2.0	0	0
0.96	0.96	2.0	2.0	0	0
0.96	0.96	2.0	2.0	0	0
0.78	0.78	1.5	1.5	0	0
0.78	0.78	1.5	1.5	0	0

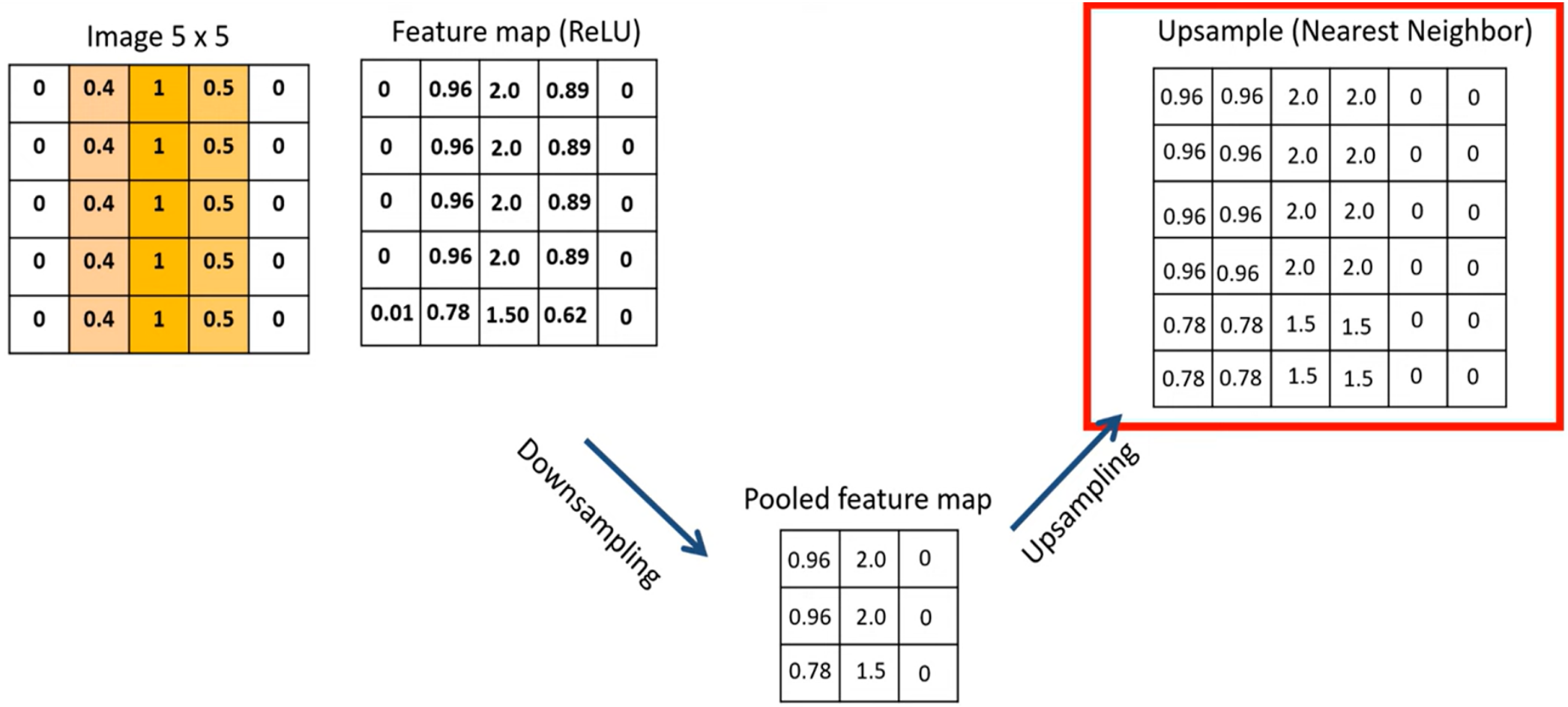
Downsampling

Pooled feature map

0.96	2.0	0
0.96	2.0	0
0.78	1.5	0

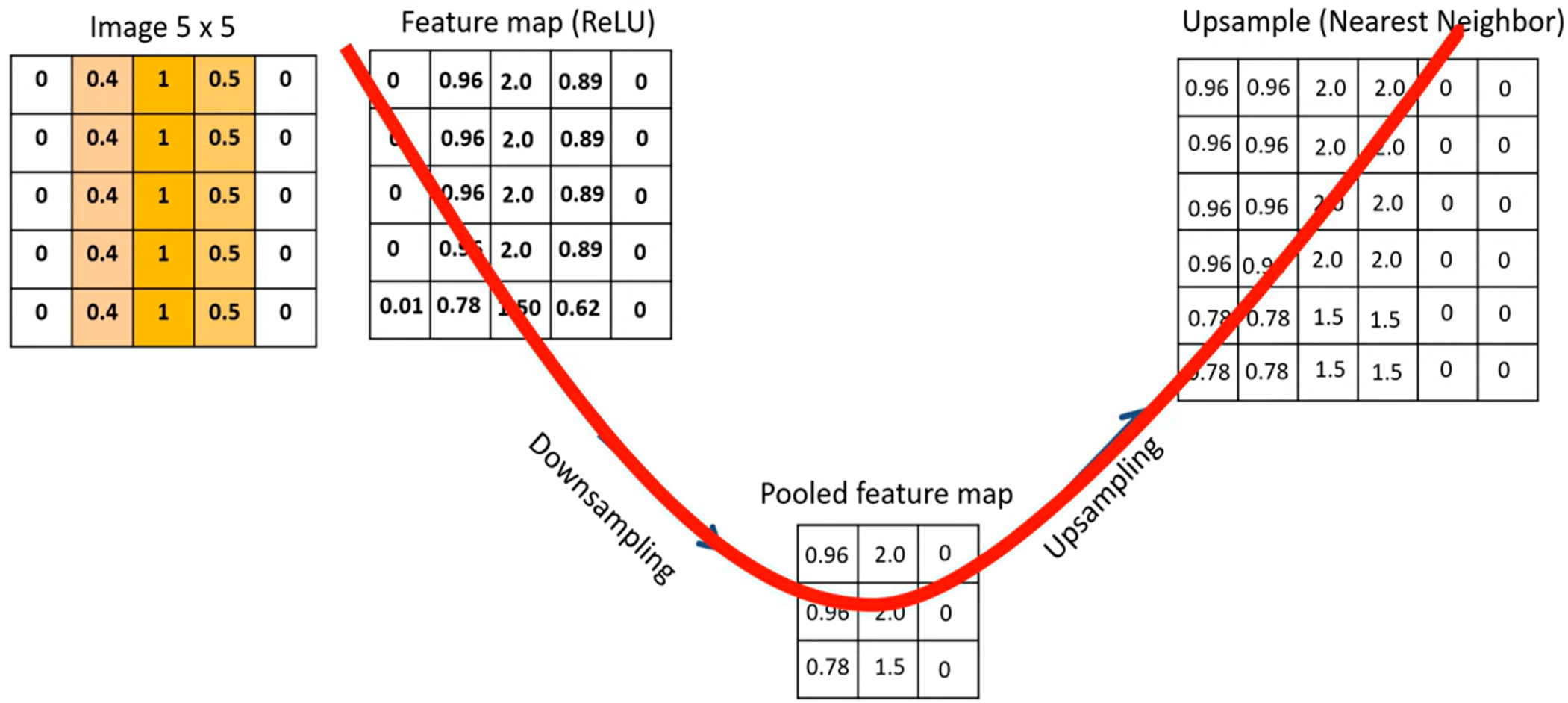
How a U-Net Works?

And then performed **upsampling**, which **increased** the size of the matrix



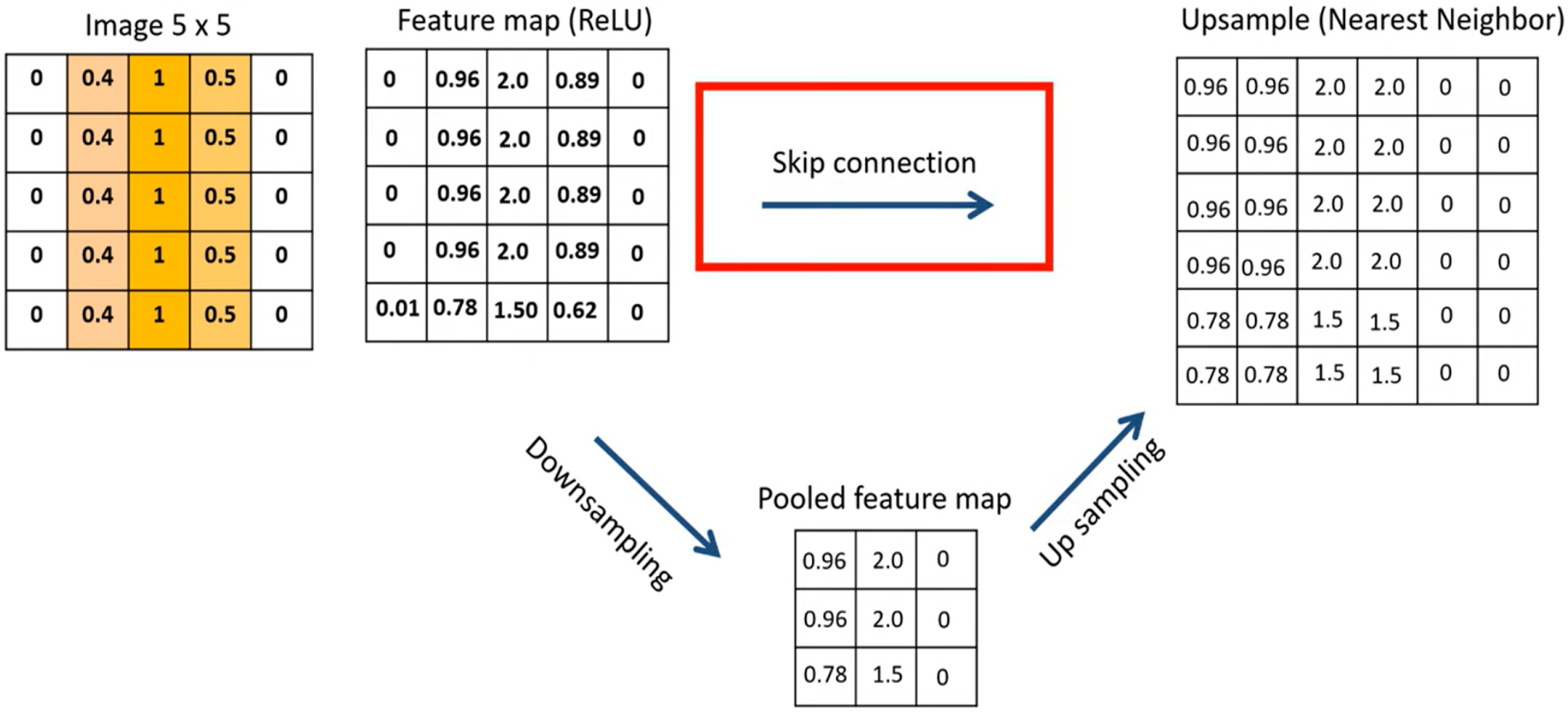
How a U-Net Works?

This looks like a U, which explains the name U-Net.



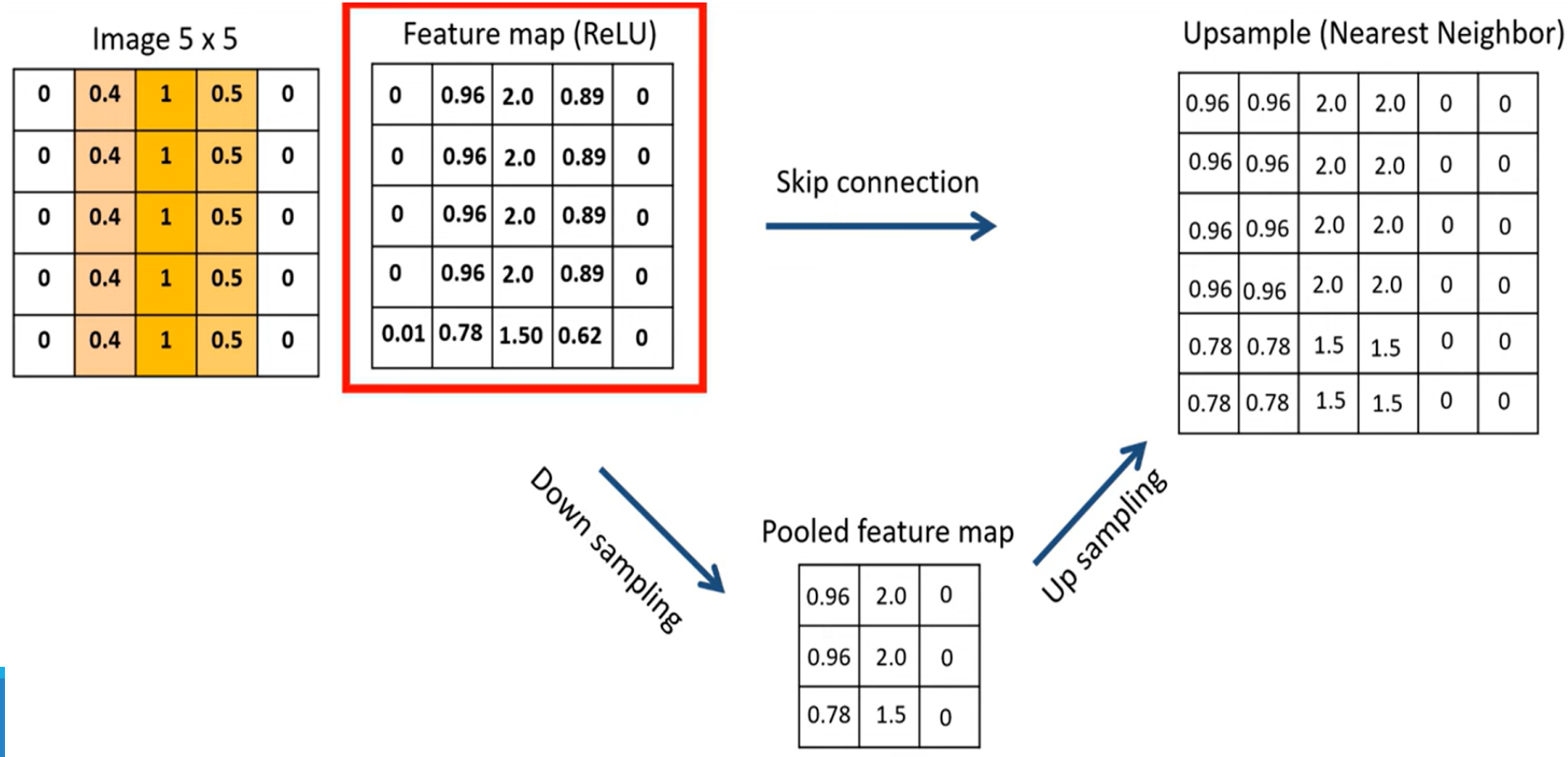
How a U-Net Works?

Another step in U-Net is the **skip connection**



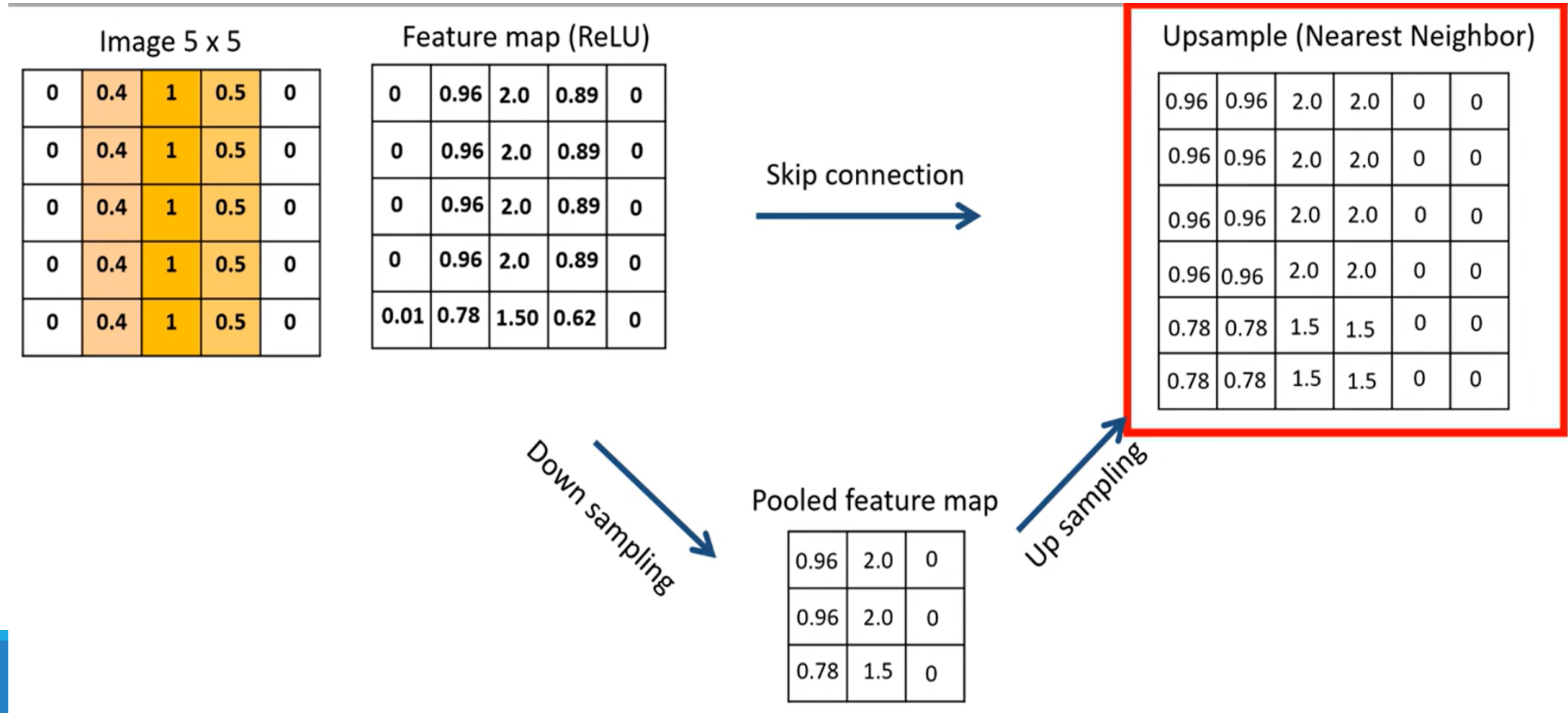
How a U-Net Works?

Which means that we concatenate the feature map



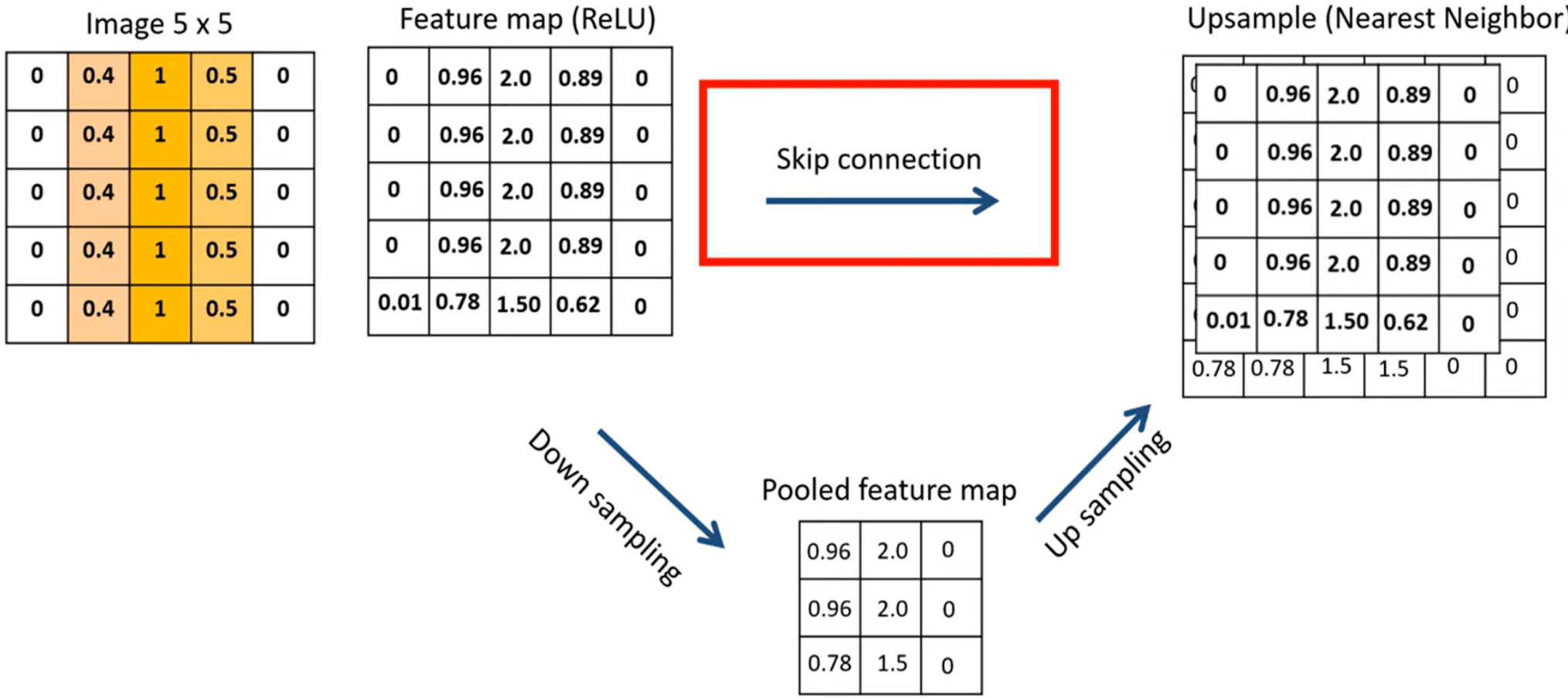
How a U-Net Works?

With the upsampled feature map



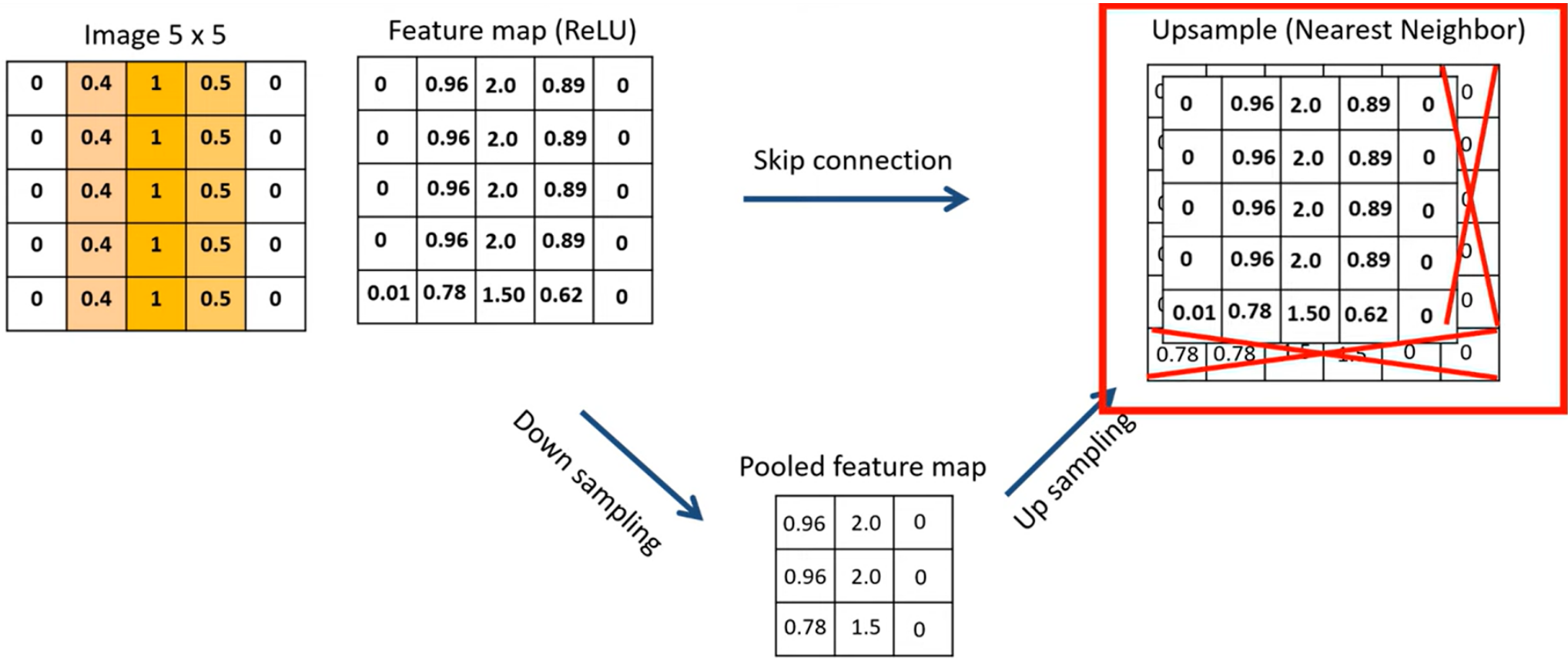
How a U-Net Works?

Concatenation of the feature map with the upsampled feature map



How a U-Net Works?

We then need to **crop** the upsampled feature map so that the two feature maps have the same size



How a U-Net Works?

The two feature maps are side-by-side for reference

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

How a U-Net Works?

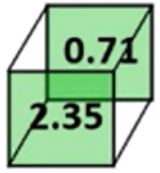
- Final Convolution is computed using $\mathbf{1} \times \mathbf{1}$ filter with depth **Two**
- This filter will be sliding over the two **overlapping** feature maps

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

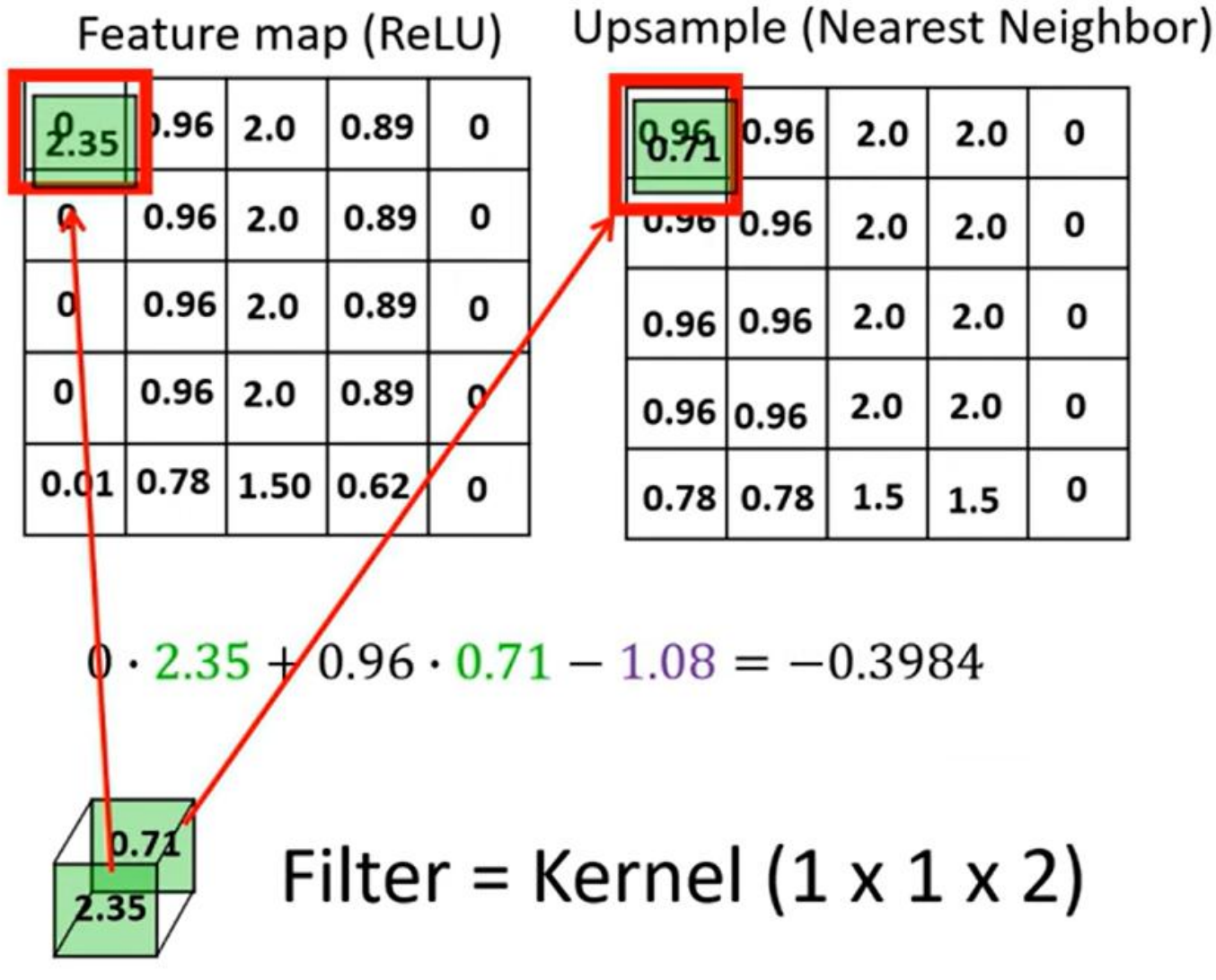
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0



Filter = Kernel (1 x 1 x 2)

How a U-Net Works?

- The calculations are same as we did before



How a U-Net Works?

- And input this sum into the **sigmoid activation** function to **constrain** the output values between **zero** and **one**

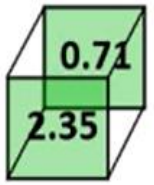
Feature map (ReLU)

0 2.35	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96 0.71	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

$$0 \cdot 2.35 + 0.96 \cdot 0.71 - 1.08 = -0.3984 \qquad \frac{1}{(1 + e^{-x})} = 0.40$$



Filter = Kernel (1 x 1 x 2)

How a U-Net Works?

- Then this output is plugged into the **final predicted output matrix**.
- This is called a **probability map**.

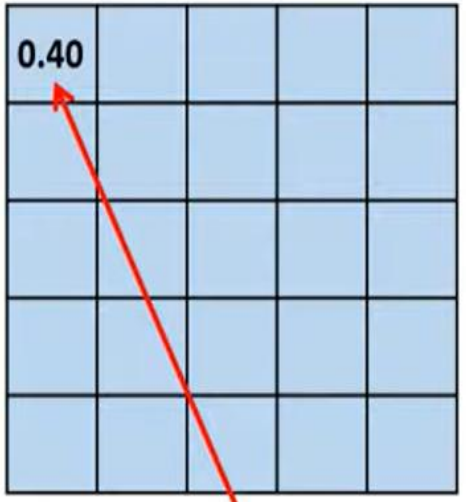
Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

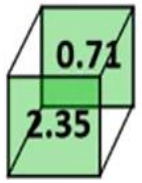
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)



$$0 \cdot 2.35 + 0.96 \cdot 0.71 - 1.08 = -0.3984$$

$$\frac{1}{(1+e^{-x})} = 0.40$$



Filter = Kernel (1 x 1 x 2)

How a U-Net Works?

- Then we slide the filter one step to the right

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

2.35

Upsample (Nearest Neighbor)

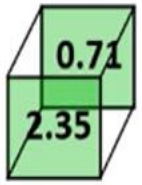
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

0.71

Final predictions (Sigmoid)

0.40				

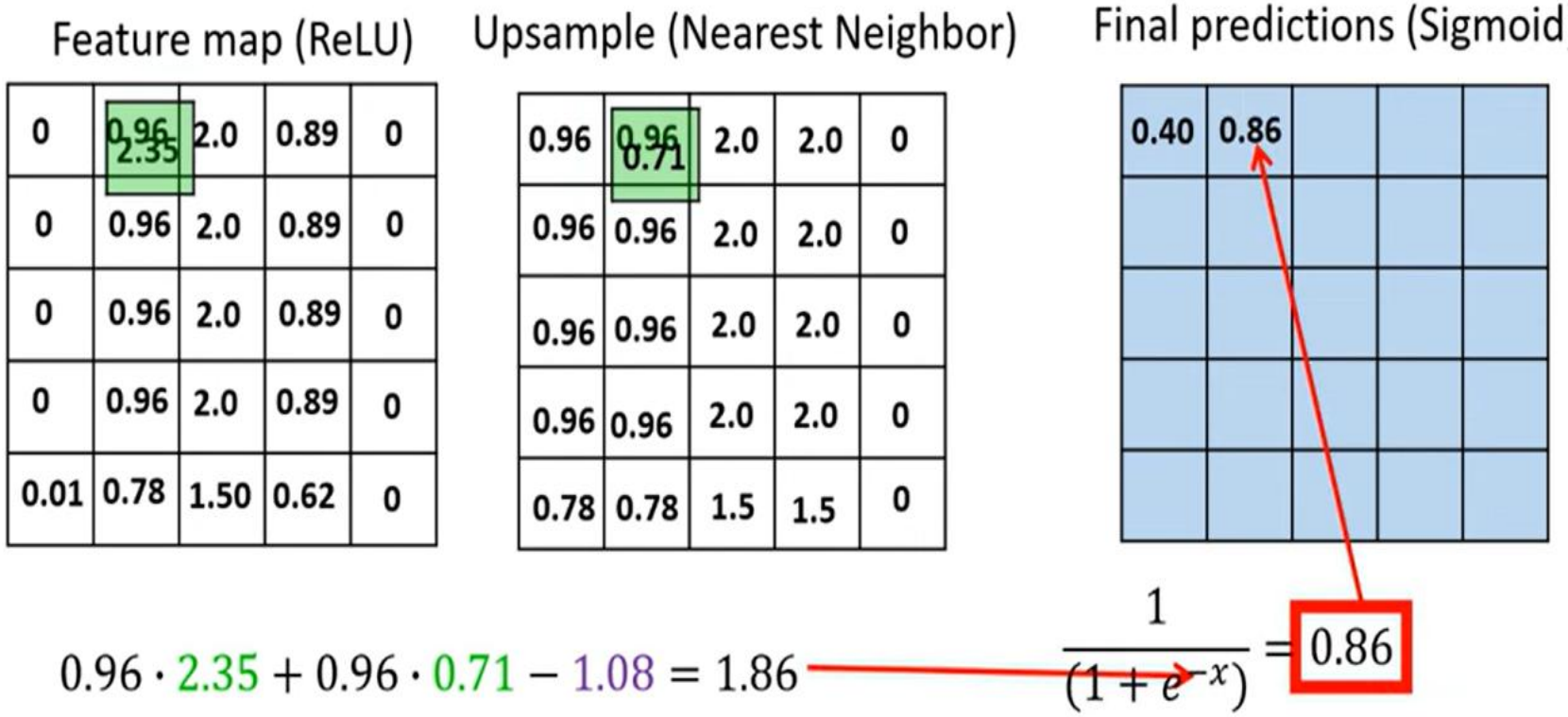
$$0.96 \cdot 2.35 + 0.96 \cdot 0.71 - 1.08 = 1.86$$

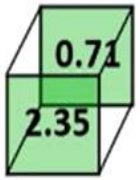


Filter = Kernel (1 x 1 x 2)

How a U-Net Works?

- The computed value is input to the probability map





Filter = Kernel (1 x 1 x 2)

How a U-Net Works?

These values can be seen as the **probability** that the corresponding pixel in the input image **belongs to the foreground**

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

How a U-Net Works?

For example, the probability that this pixel belongs to the foreground is 40%.

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

How a U-Net Works?

If a given **probability is larger than 0.5**, we usually classify the pixel as belonging to the **foreground** in the image

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

If $p > 0.5$ -> 1

If $p < 0.5$ -> 0

How a U-Net Works?

And if it is **less than 0.5**, we classify the pixel as belonging to the **background**.

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

If $p > 0.5$ -> 1

If $p < 0.5$ -> 0

How a U-Net Works?

If this rule is applied on these probabilities

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

If $p > 0.5$ -> 1

If $p < 0.5$ -> 0

How a U-Net Works?

We will get the **final predicted segmentation map**

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

Predicted segmentation map

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

How a U-Net Works?

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

Predicted segmentation map

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

We have gone from this image

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0

Segmented image

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

How a U-Net Works?

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

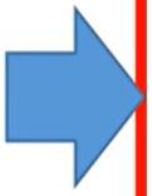
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

Predicted segmentation map

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

Image 5 x 5

0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0
0	0.4	1	0.5	0



Segmented image

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

To this image

How a U-Net Works?

During the training of U-Net, the eight weights that are associated with the filters are optimized

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

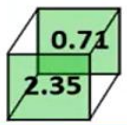
Predicted segmentation map

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

Filters

1.45	0.31
0.54	-0.04

Bias $= -0.11$



Bias $- 1.08$

How a U-Net Works?

So that the **predicted segmentation map**,

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

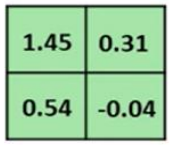
Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

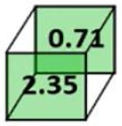
Predicted segmentation map

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

Filters



Bias $= -0.11$



Bias $- 1.08$

How a U-Net Works?

Recreates the **ground-truth image** as good as possible

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

Predicted segmentation map

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

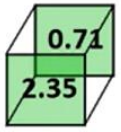
Ground truth segmented image

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

Filters

1.45	0.31
0.54	-0.04

Bias $= -0.11$



Bias $- 1.08$

How a U-Net Works?

This means that the network tries to reduce these **probability** values down to **zero**

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

Predicted segmentation map

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

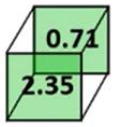
Ground truth segmented image

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

Filters

1.45	0.31
0.54	-0.04

Bias $= -0.11$



Bias $- 1.08$

How a U-Net Works?

And these **probability** values up to **one**

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

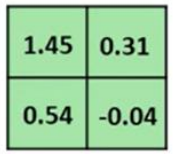
Predicted segmentation map

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

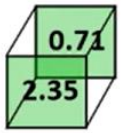
Ground truth segmented image

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

Filters



Bias $= -0.11$



Bias $- 1.08$

How a U-Net Works?

By changing the values of the weights

Feature map (ReLU)

0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0	0.96	2.0	0.89	0
0.01	0.78	1.50	0.62	0

Upsample (Nearest Neighbor)

0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.96	0.96	2.0	2.0	0
0.78	0.78	1.5	1.5	0

Final predictions (Sigmoid)

0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.40	0.86	0.99	0.92	0.25
0.38	0.79	0.97	0.81	0.25

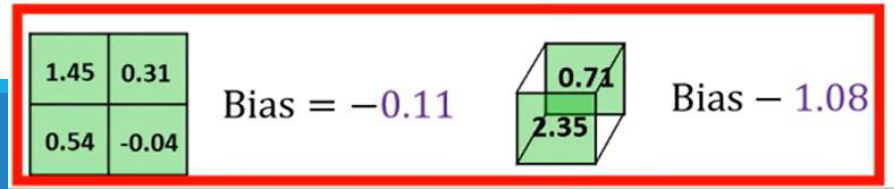
Predicted segmentation map

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

Ground truth segmented image

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

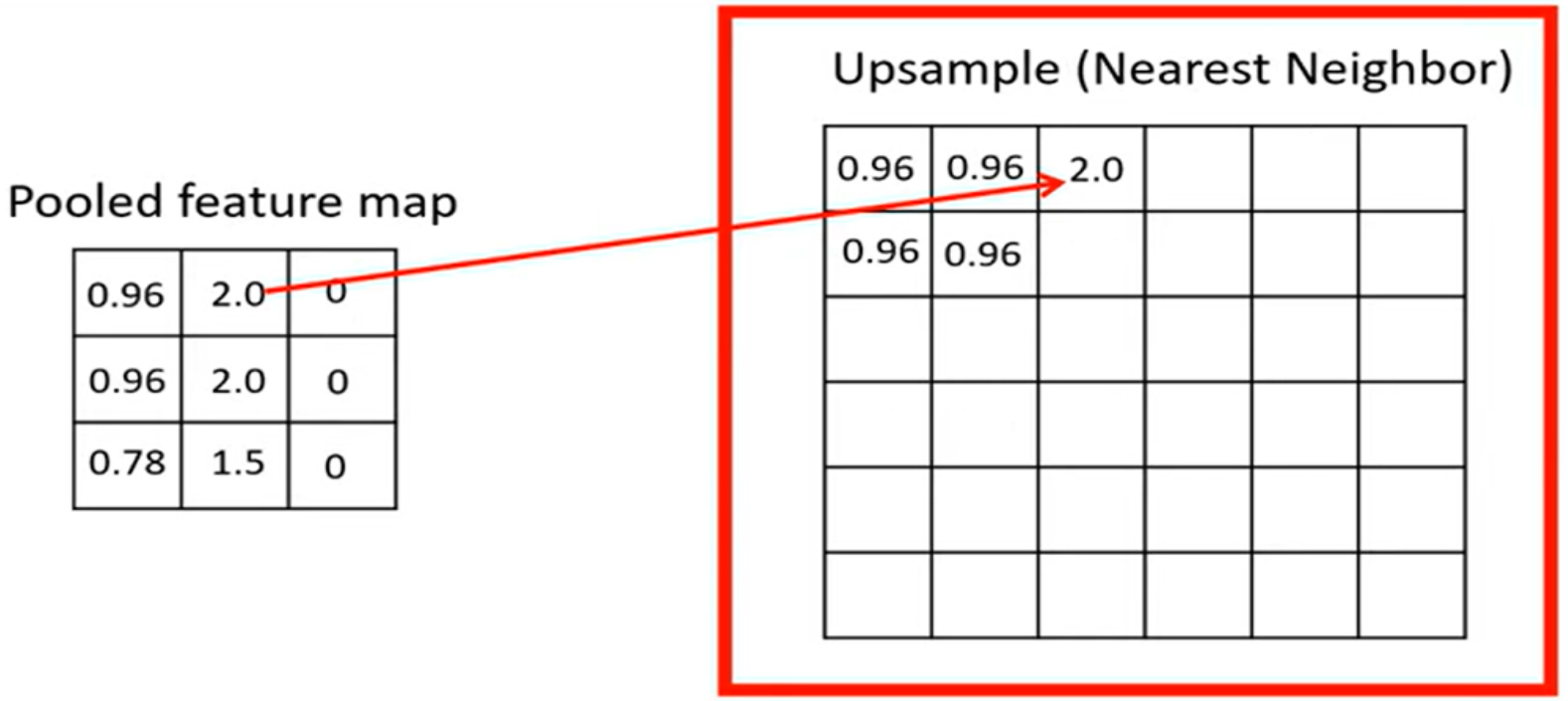
Filters



Transposed Convolution

Transposed Convolution

- We previously **upsampled** the image by copying a value from the **pooled feature map** and placed these values in the nearest neighboring elements.
- One **drawback** of this method is that it **does not involve any trainable weights**

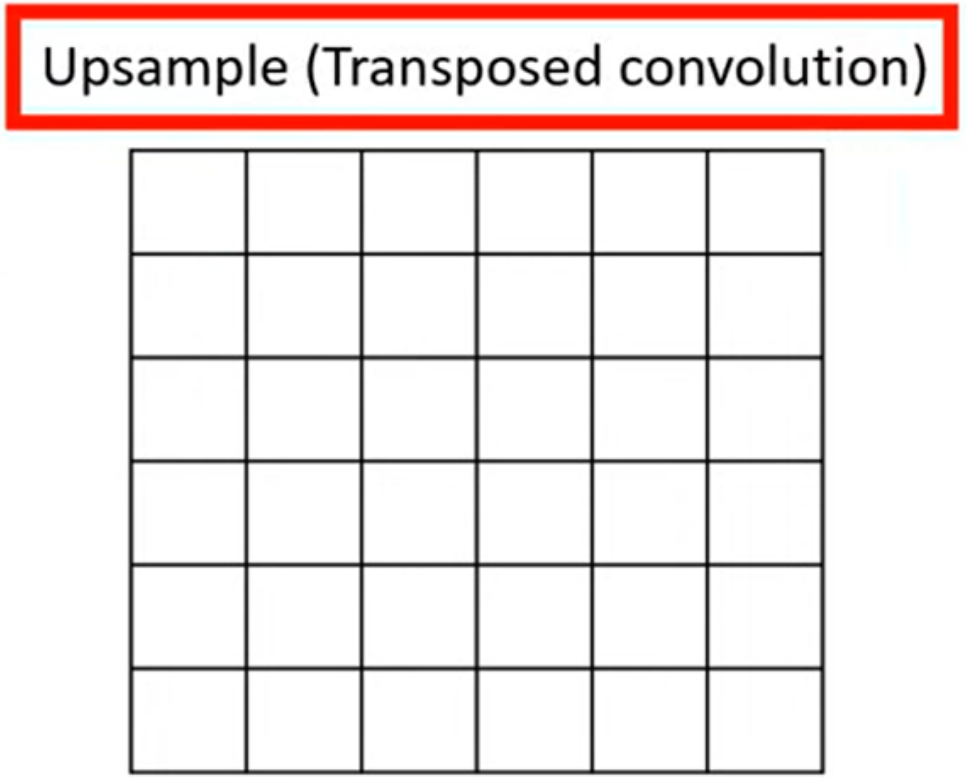


Transposed Convolution

- Here is another method for upsampling that is called **transposed convolution**.

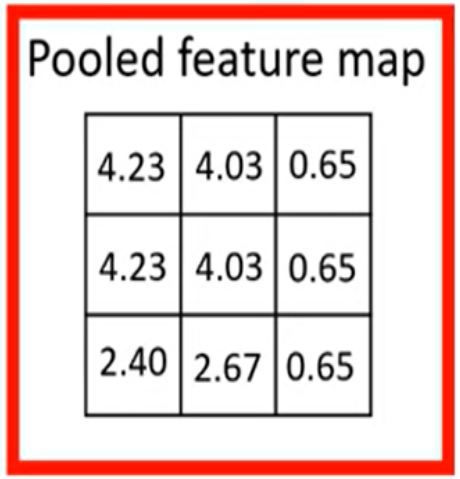
Pooled feature map

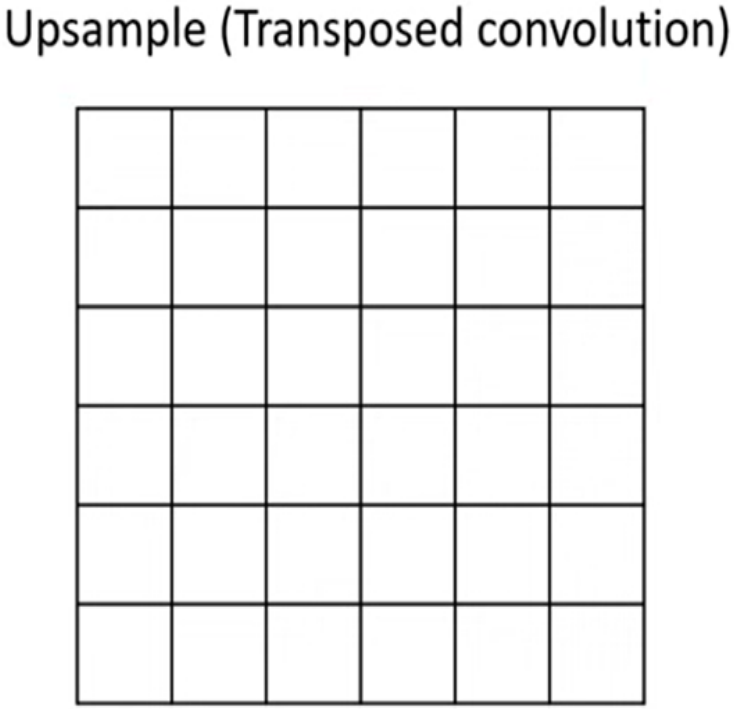
4.23	4.03	0.65
4.23	4.03	0.65
2.40	2.67	0.65



Transposed Convolution

- If we use this method, we will get other values in our pooled feature map because the previous weights in the filter will be optimized to other values when we add additional weights to our network.





Transposed Convolution

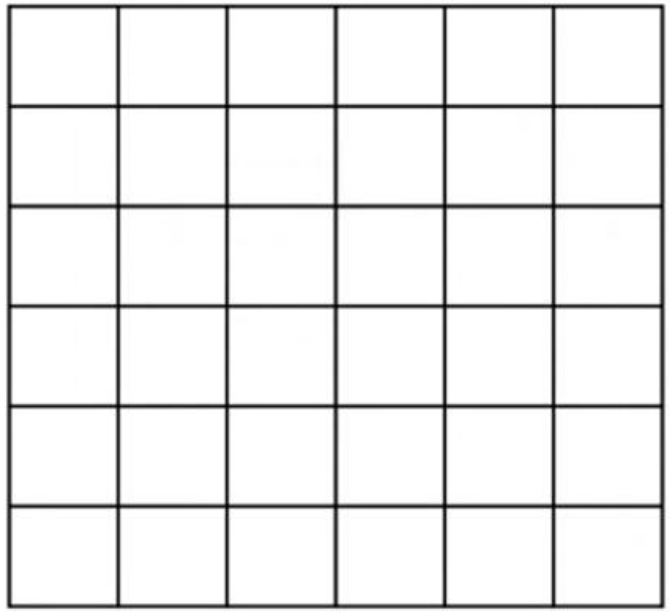
- This is the **additional filter** that we will use to perform upsampling.
- The filter has the following optimized weights,

Pooled feature map

4.23	4.03	0.65
4.23	4.03	0.65
2.40	2.67	0.65

0.84	-1.0
0.94	-0.65

Upsample (Transposed convolution)



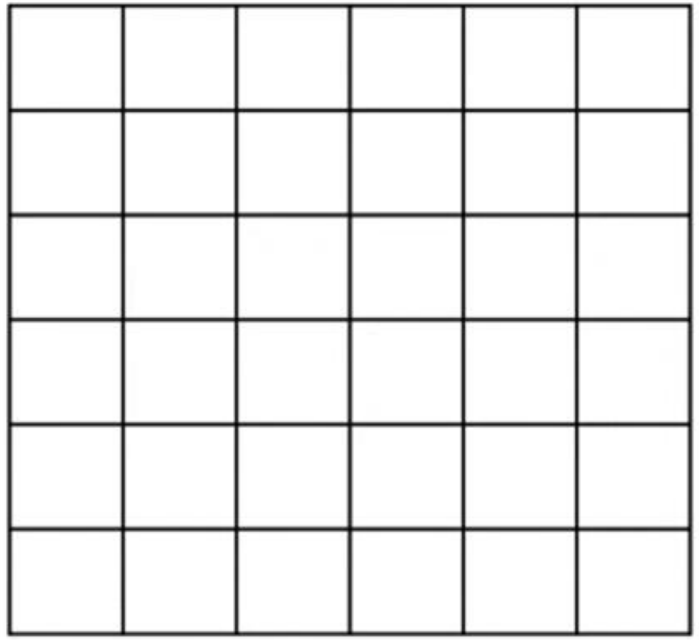
Transposed Convolution

- And the following optimized value of the bias weight

Pooled feature map

4.23	4.03	0.65
4.23	4.03	0.65
2.40	2.67	0.65

Upsample (Transposed convolution)



0.84	-1.0
0.94	-0.65

Bias = 1.43

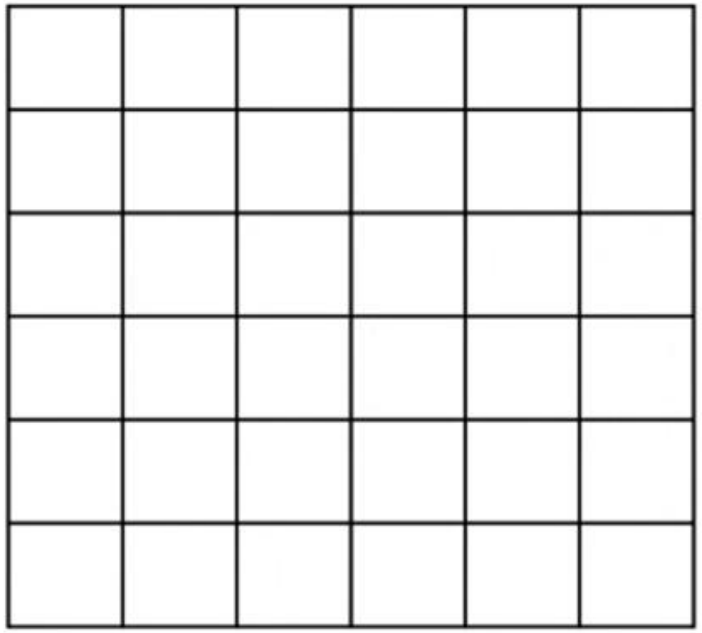
Transposed Convolution

We start by multiplying these values

Pooled feature map

4.23	4.03	0.65
4.23	4.03	0.65
2.40	2.67	0.65

Upsample (Transposed convolution)



0.84	-1.0
0.94	-0.65

Bias = 1.43

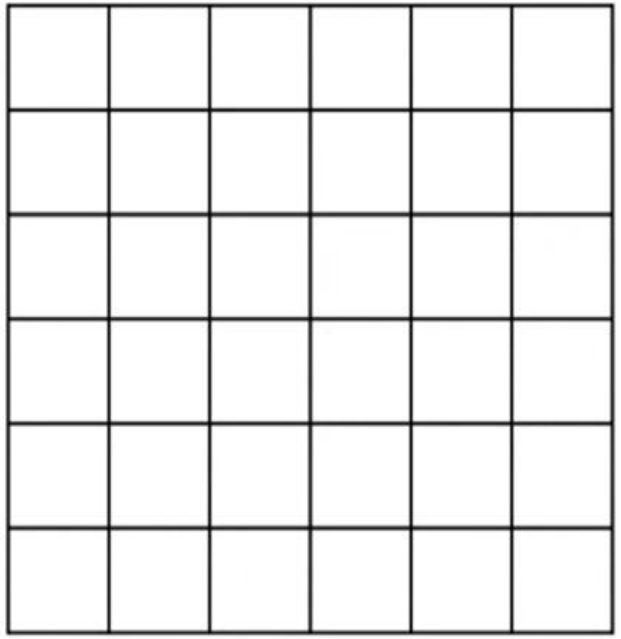
Transposed Convolution

And add the bias weight to this product

Pooled feature map

4.23	4.03	0.65
4.23	4.03	0.65
2.40	2.67	0.65

Upsample (Transposed convolution)



0.84	-1.0
0.94	-0.65

Bias = 1.43

$$4.23 \cdot 0.84 + 1.43 = 4.98$$

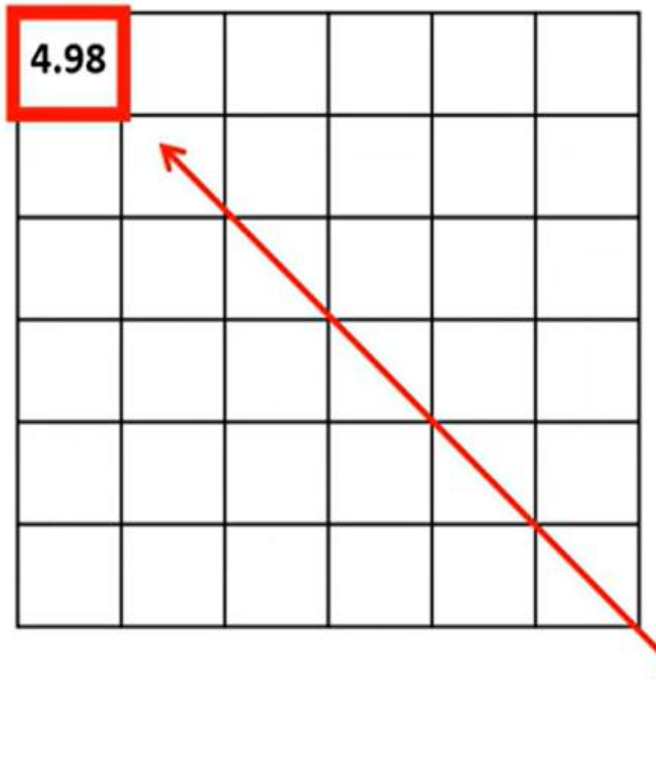
Transposed Convolution

The final sum is input as given:

Pooled feature map

4.23	4.03	0.65
4.23	4.03	0.65
2.40	2.67	0.65

Upsample (Transposed convolution)



0.84	-1.0
0.94	-0.65

Bias = 1.43

$4.23 \cdot 0.84 + 1.43 = 4.98$

Transposed Convolution

Then we multiply these values

Pooled feature map

4.23	4.03	0.65
4.23	4.03	0.65
2.40	2.67	0.65

Upsample (Transposed convolution)

4.98					

0.84	-1.0
0.94	-0.65

Bias = 1.43

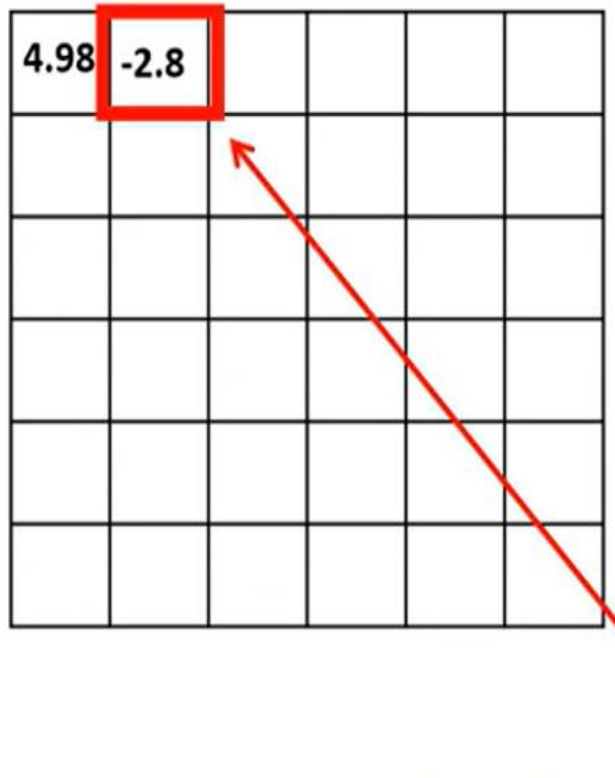
Transposed Convolution

Fill in the computed value as given:

Pooled feature map

4.23	4.03	0.65
4.23	4.03	0.65
2.40	2.67	0.65

Upsample (Transposed convolution)



0.84	-1.0
0.94	-0.65

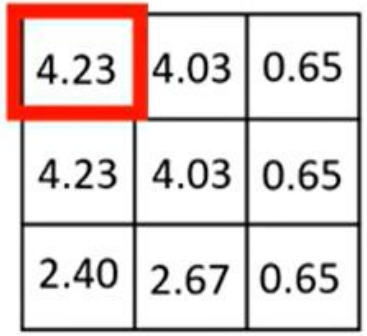
Bias = 1.43

$4.23 \cdot (-1.0) + 1.43 = -2.8$

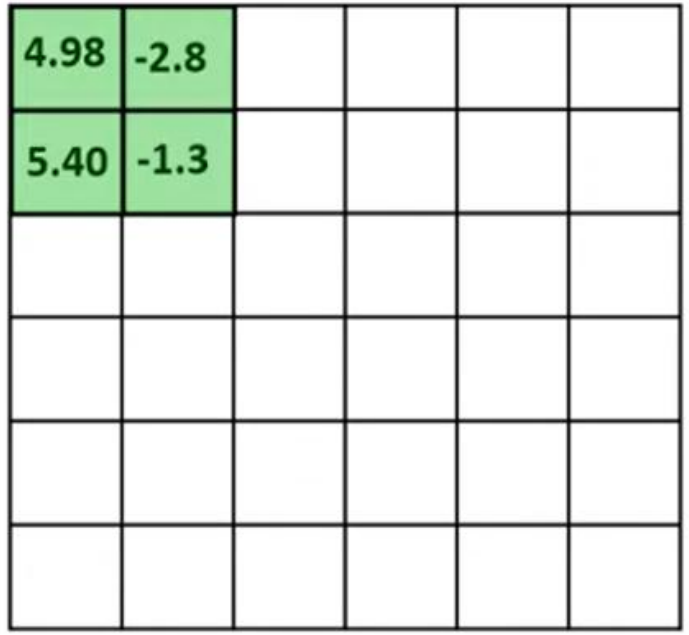
Transposed Convolution

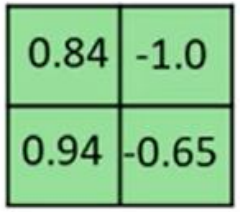
Once we are done with this value

Pooled feature map



Upsample (Transposed convolution)





Bias = 1.43

Transposed Convolution

We move to this value and do the same calculations

Pooled feature map

4.23	4.03	0.65
4.23	4.03	0.65
2.40	2.67	0.65

Upsample (Transposed convolution)

4.98	-2.8				
5.40	-1.3				

0.84	-1.0
0.94	-0.65

Bias = 1.43

Transposed Convolution

Which will result
in these values

Pooled feature map

4.23	4.03	0.65
4.23	4.03	0.65
2.40	2.67	0.65

Upsample (Transposed convolution)

4.98	**-2.8**	**4.81**	**-2.6**		
5.40	**-1.3**	**5.21**	**-1.2**		

0.84	-1.0
0.94	-0.65

Bias = 1.43

Transposed Convolution

- Once, the upsampled feature map is filled, we proceed with the exact same steps as before such as cropping and concatenation

Pooled feature map

4.23	4.03	0.65
4.23	4.03	0.65
2.40	2.67	0.65

Upsample (Transposed convolution)

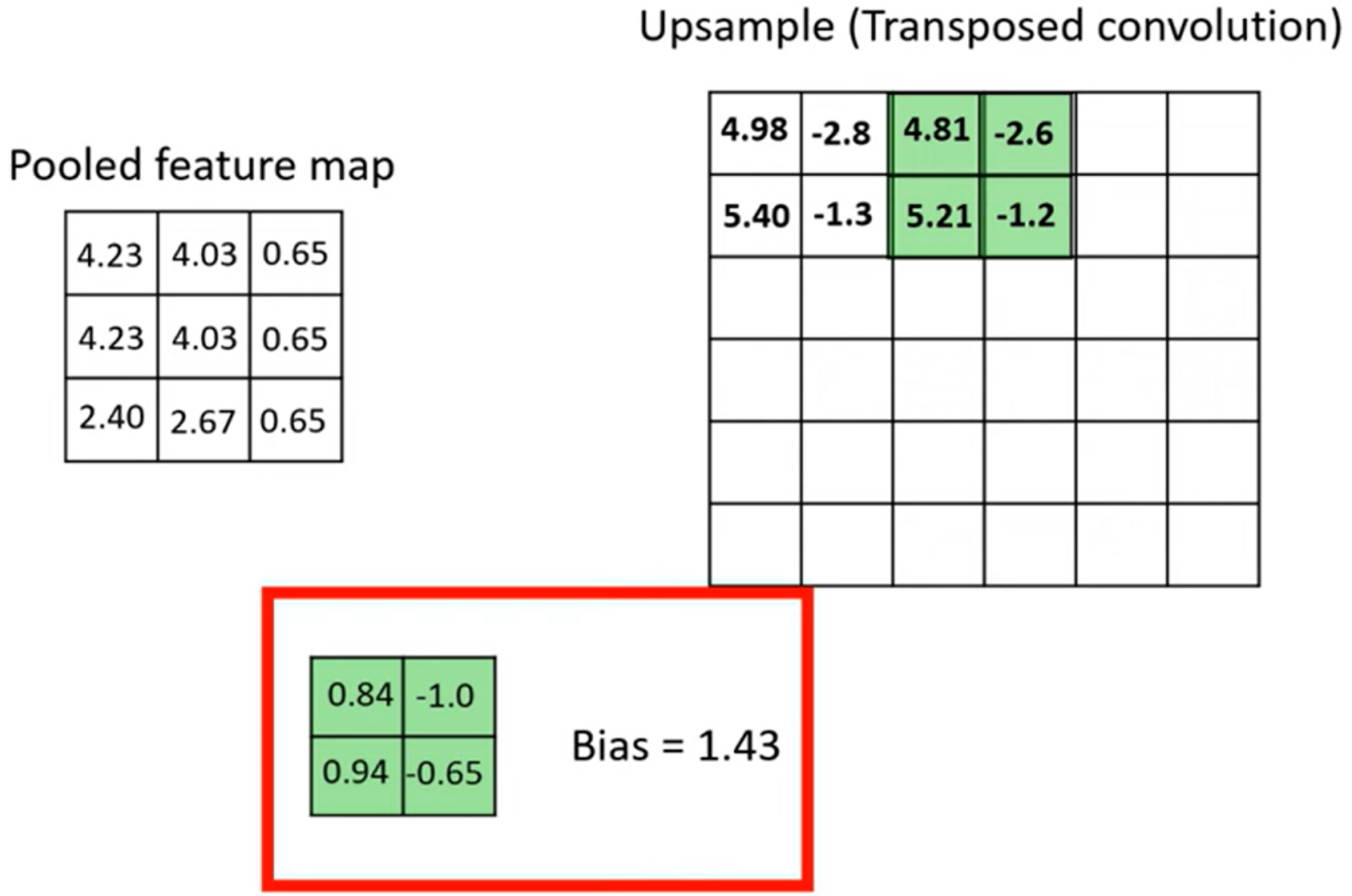
4.98	**-2.8**	**4.81**	**-2.6**		
5.40	**-1.3**	**5.21**	**-1.2**		

0.84	-1.0
0.94	-0.65

Bias = 1.43

Transposed Convolution

- Note that, when we use **transposed convolution**, the network will have **five additional weights** to tune during the training compared to when we used the **nearest neighbour method**.



Thank you