U-NET

FOR SEMANTIC SEGMENTATION

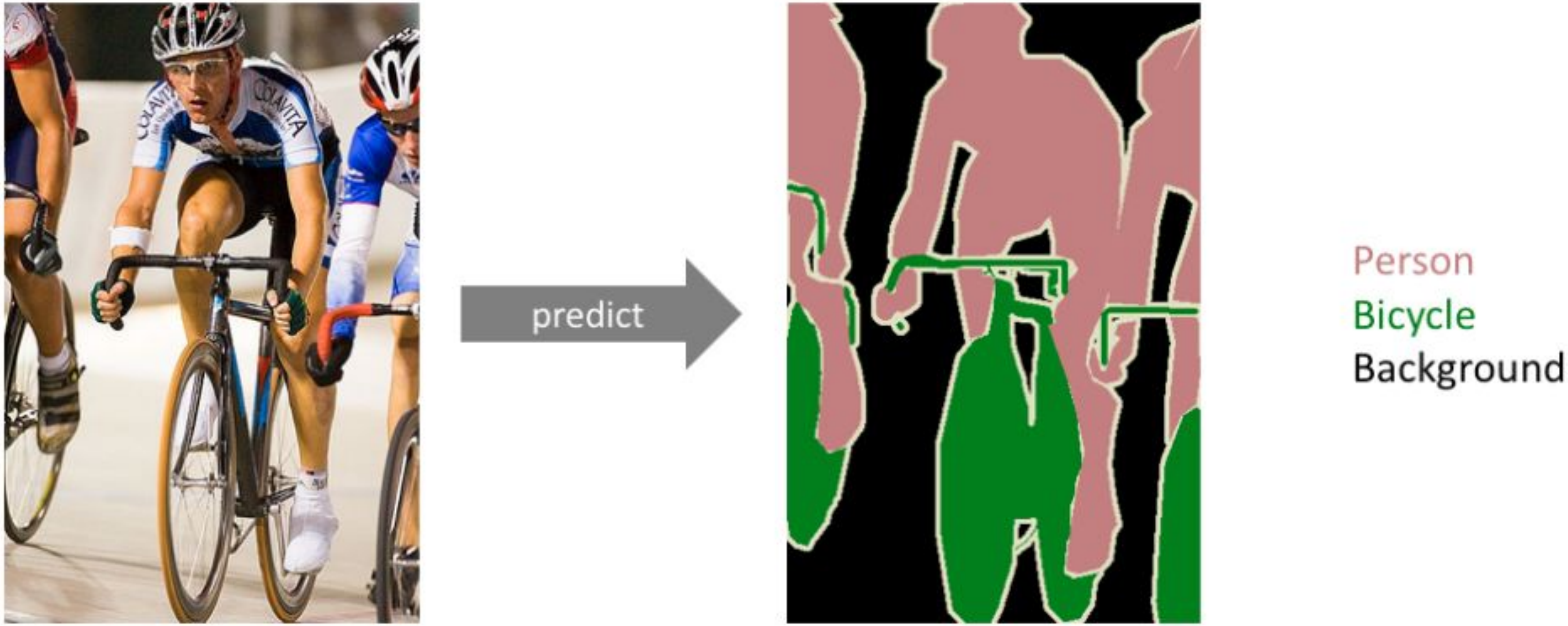
Topics to be covered

➔ Semantic Segmentation

➔ Transposed Convolution

➔ U-Net Architecture

Semantic Segmentation



Semantic Segmentation

Input

segmented

1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5	5
5	5	3	3	3	3	1	1	3	3	5	5	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	4	4	5	5	5
4	4	4	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4

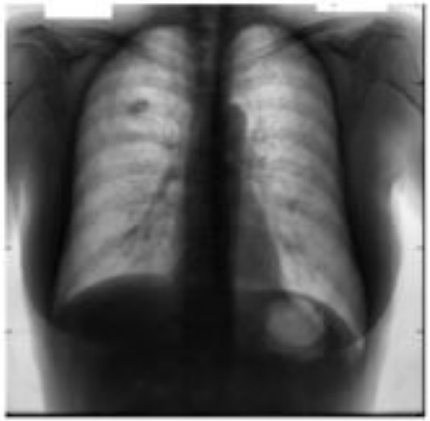
Semantic Labels

Semantic Segmentation

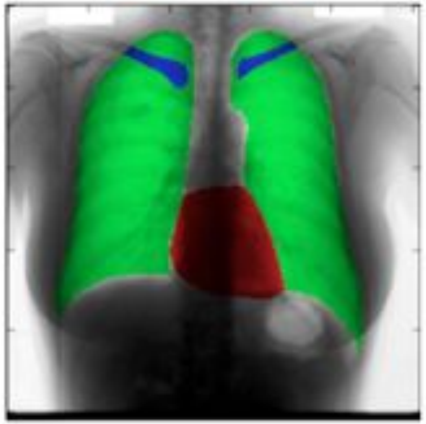
0: Background/Unknown
1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

Semantic Segmentation

Applications



Input Image



Segmented Image

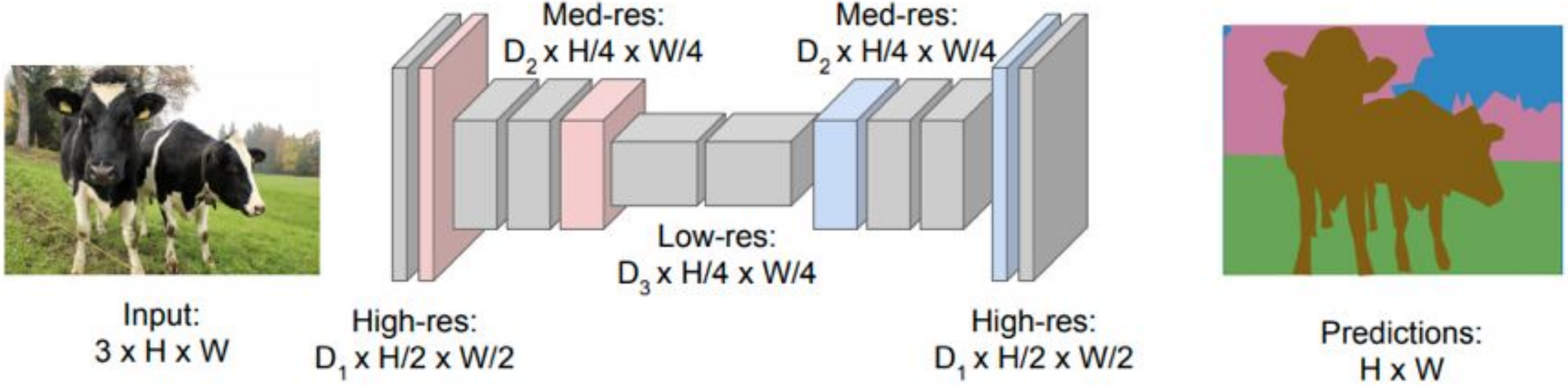
Semantic Segmentation

Applications

Semantic Segmentation

Approach

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



How can we perform **UPSAMPLING**?

How can we perform **UPSAMPLING**?

Interpolation

Interpolation

- ➔ Nearest-Neighbor Interpolation
- ➔ Bi-linear Interpolation
- ➔ Bi-cubic Interpolation

Interpolation

- ➔ Nearest-Neighbor Interpolation
- ➔ Bi-linear Interpolation
- ➔ Bi-cubic Interpolation

- ➔ What's the problem with interpolation in case of CNNs?

Interpolation

- ➔ Nearest-Neighbor Interpolation
- ➔ Bi-linear Interpolation
- ➔ Bi-cubic Interpolation

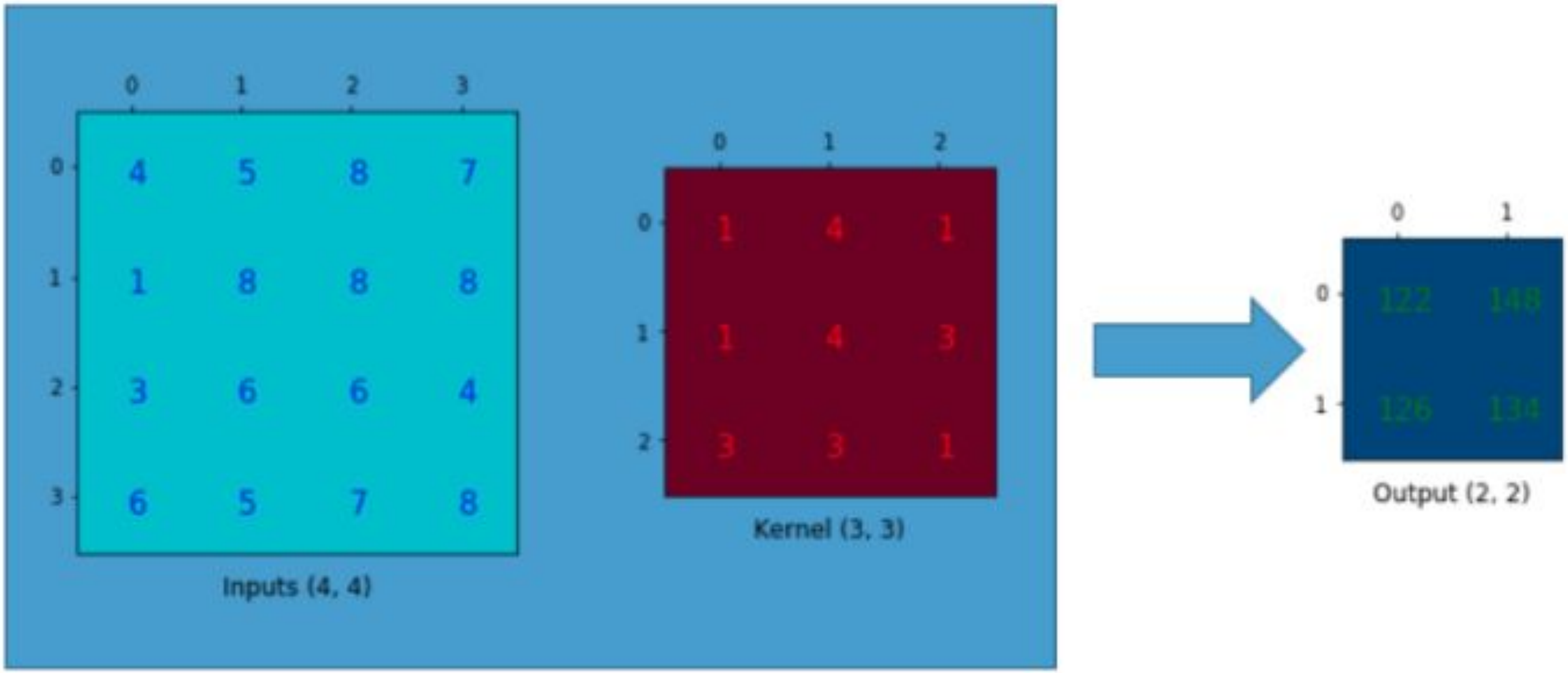
- ➔ What’s the problem with interpolation in case of CNNs?
- ➔ We have to manually choose the type of interpolation, and we can’t apply feature engineering here. The interpolation is not learnable.

How can we **learn** this **UPSAMPLING**?

How can we **learn** this **UPSAMPLING**?

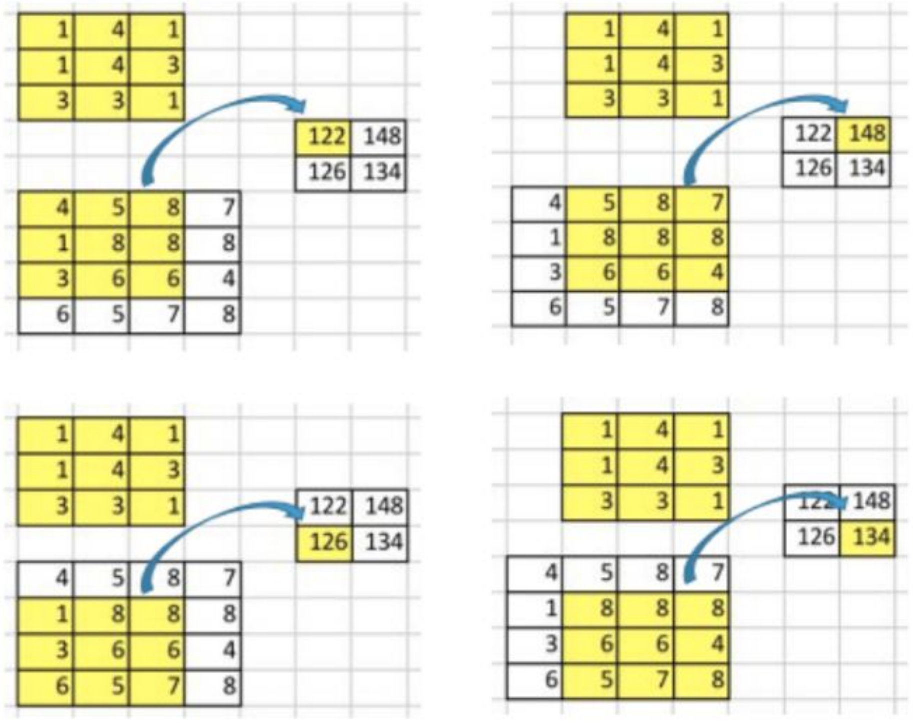
Transposed Convolution

Convolution



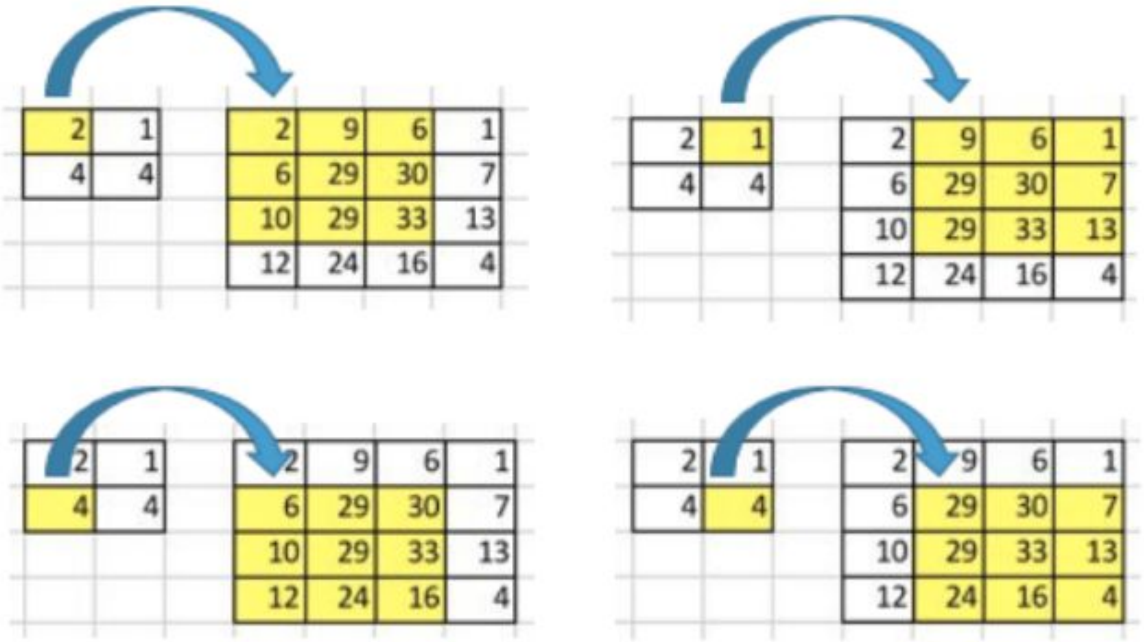
Convolution Operation

Convolution : Many-to-One



Sum of Element-wise Multiplication

Backward Convolution : One-to-Many

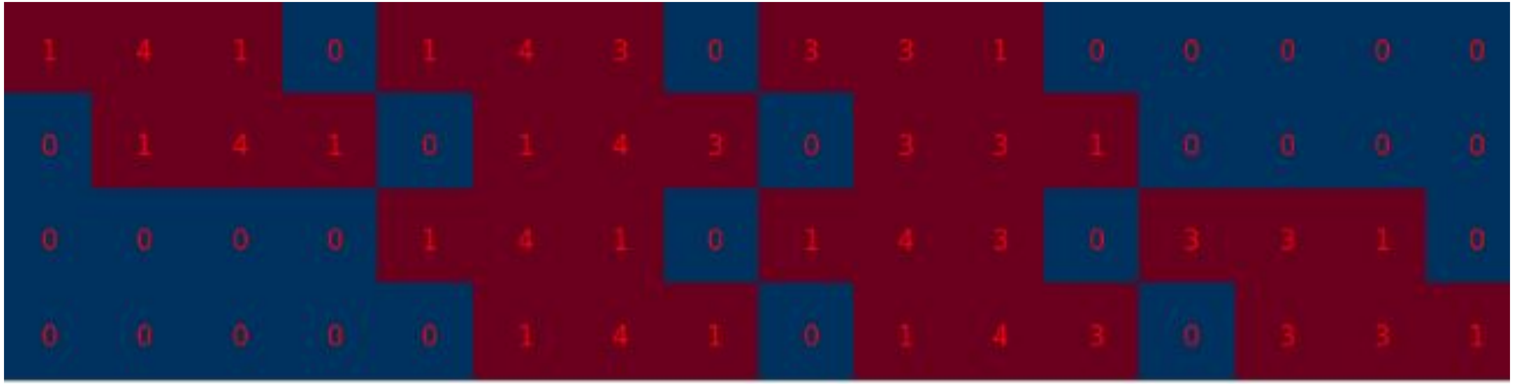


Going Backward of Convolution

Convolution Matrix

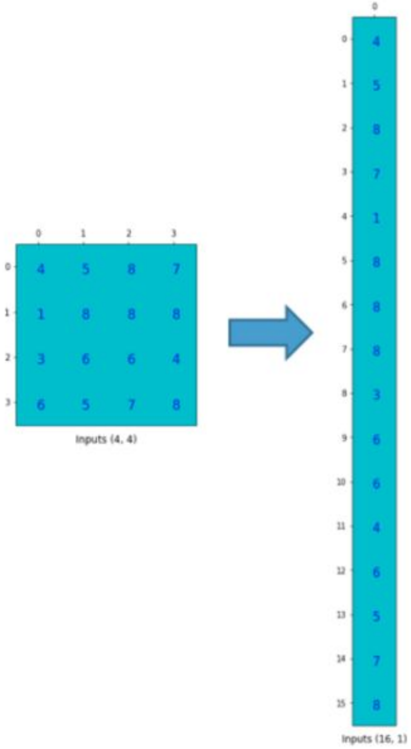
Kernel (3, 3)

We converted **3x3** kernel into **4x16** matrix



Convolution Matrix (4, 16)

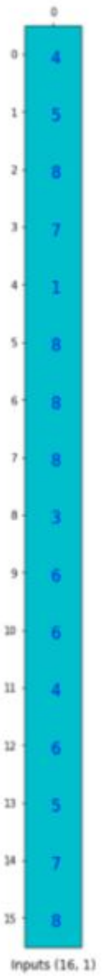
Convolution Matrix

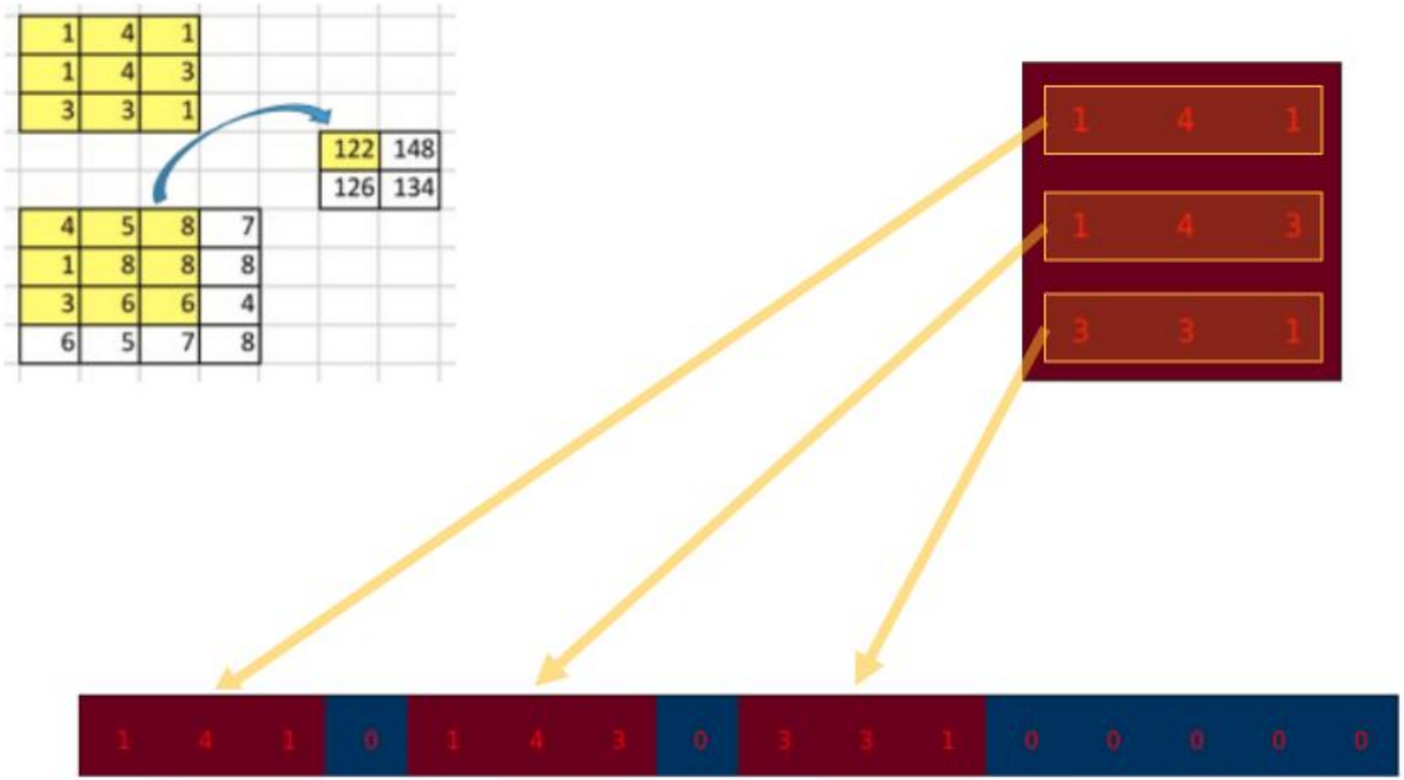


Flattened Input Matrix

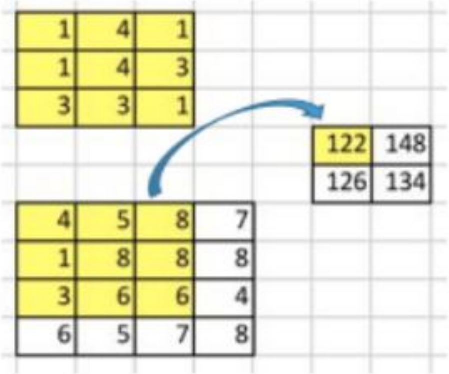
We flattened **4x4** input to **16x1** vector

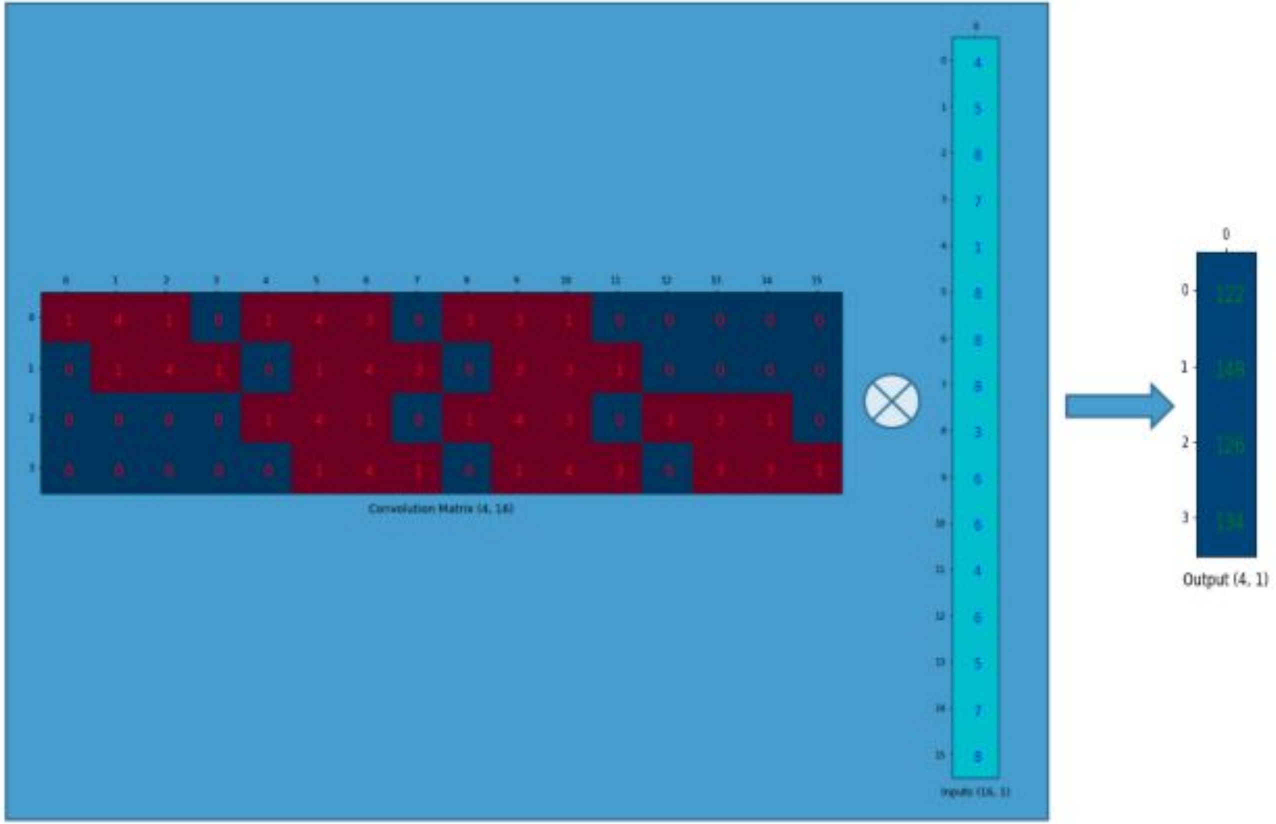
Convolution Matrix



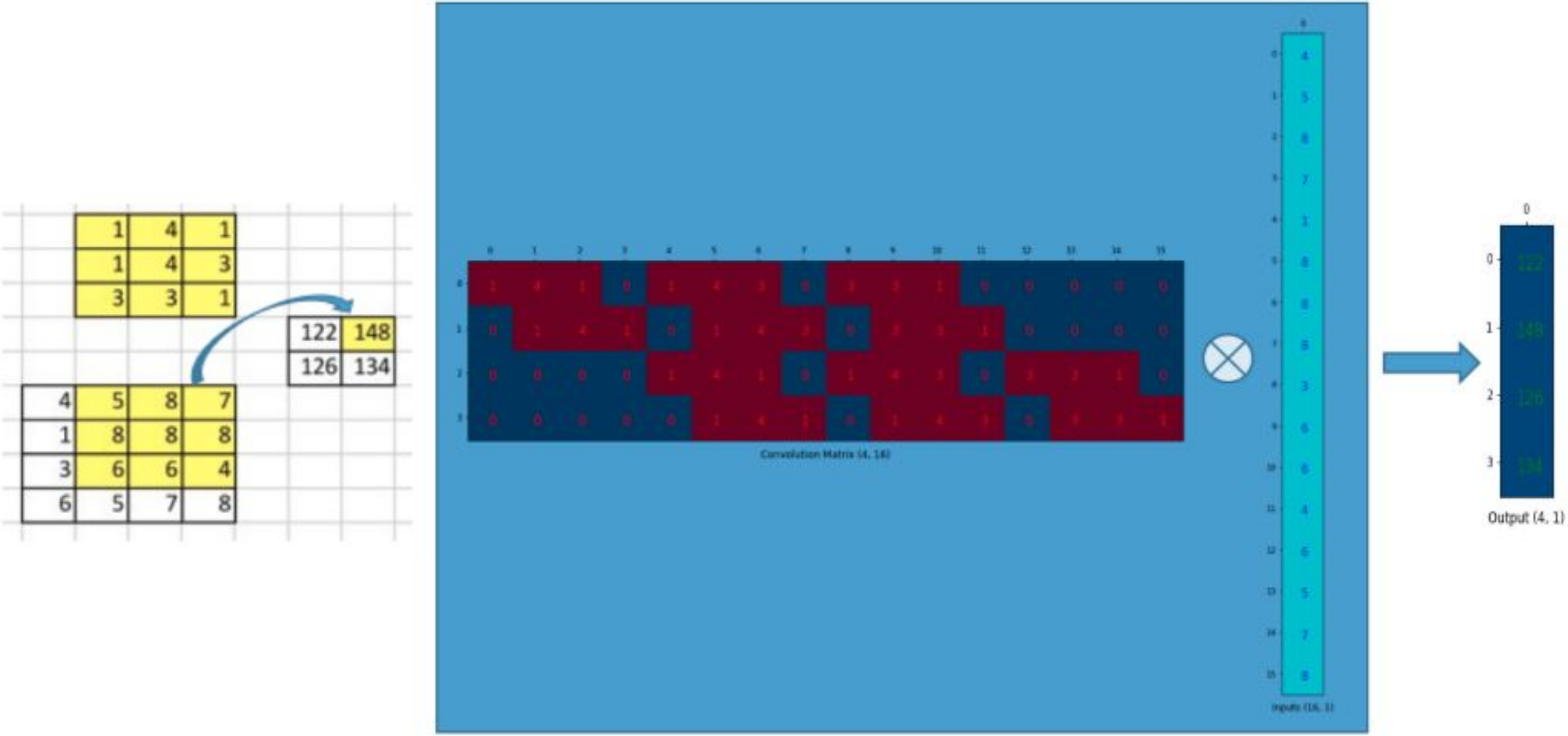


Convolution Matrix

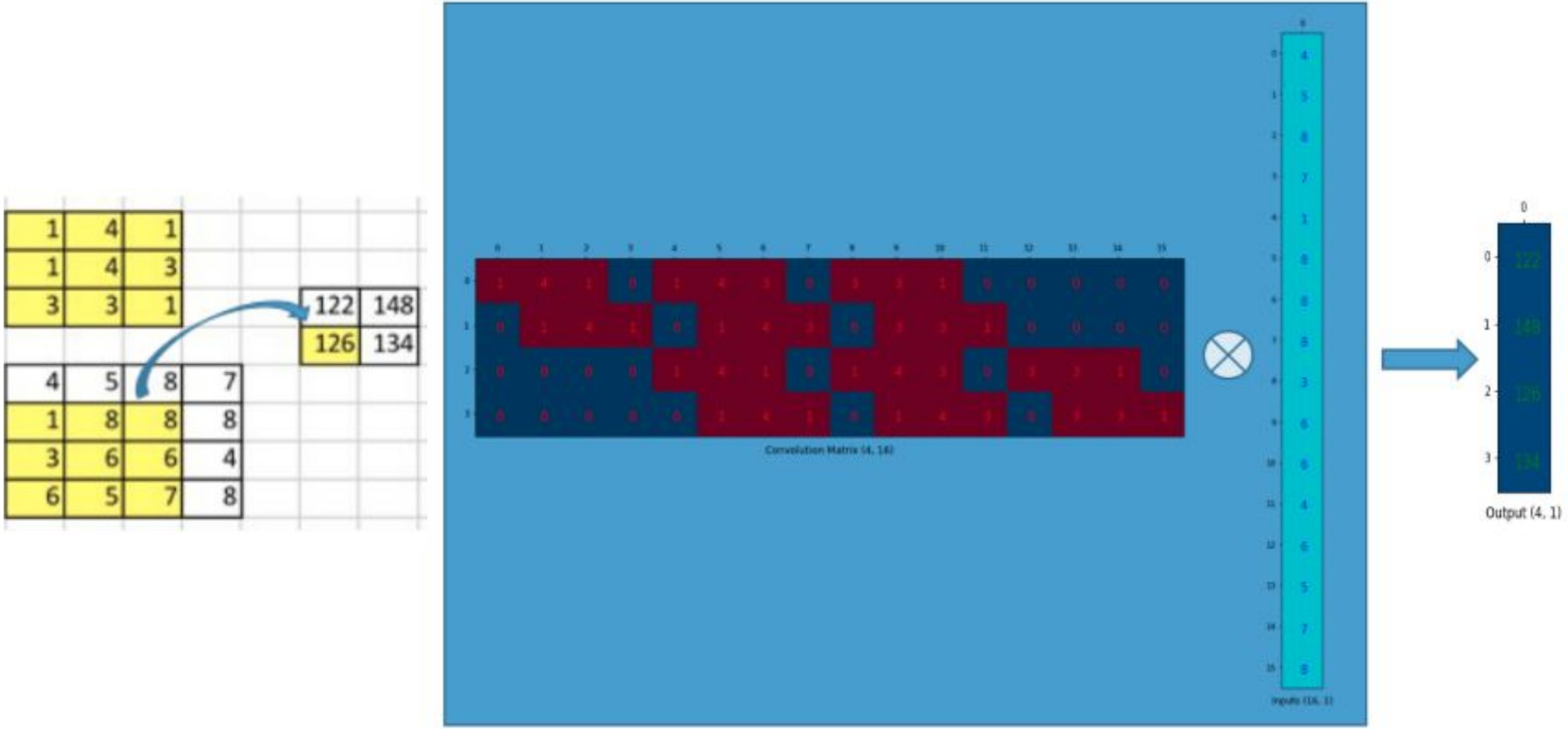




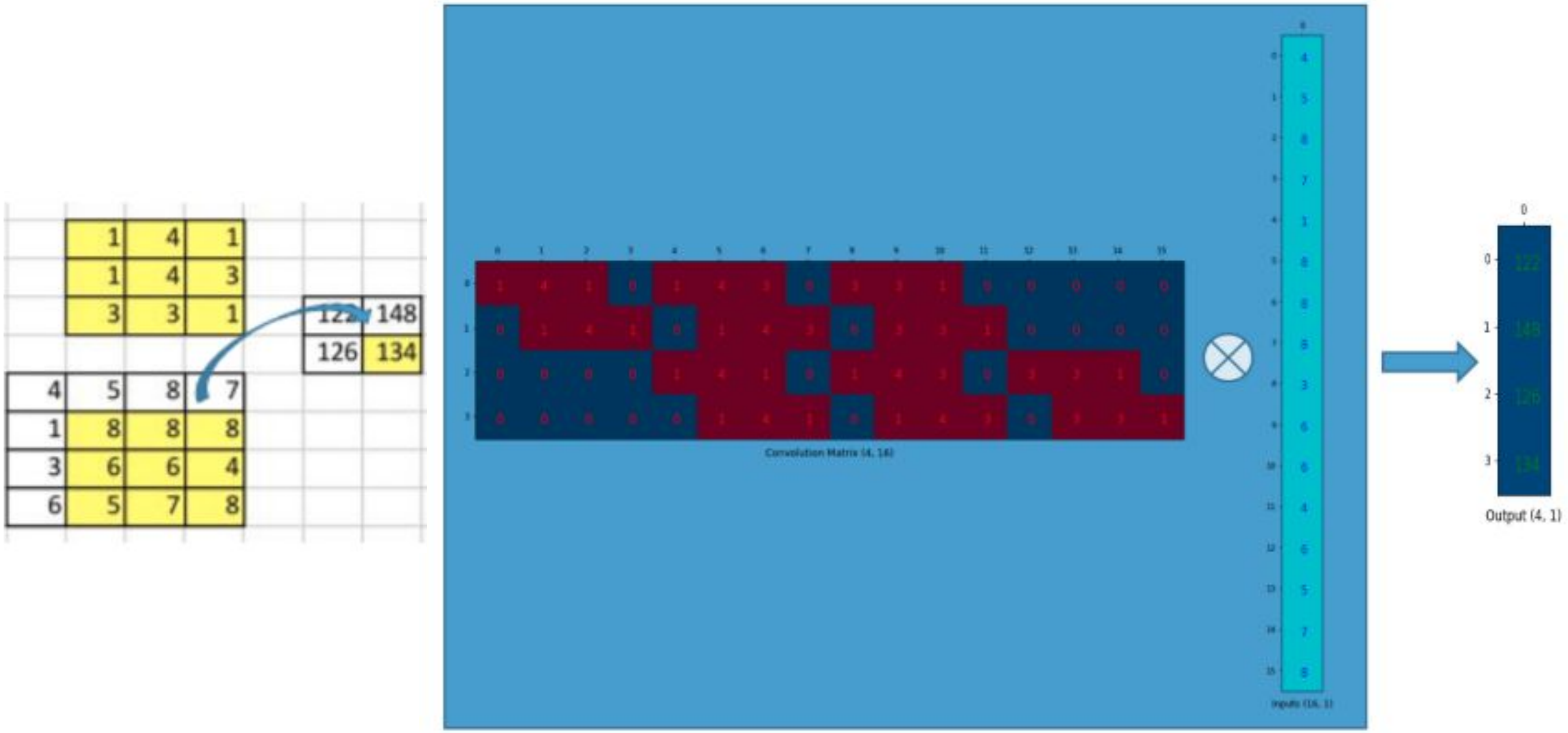
Convolution Matrix



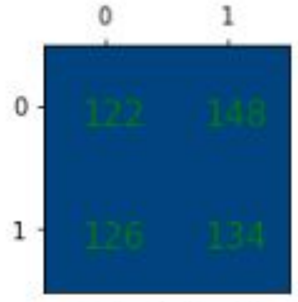
Convolution Matrix



Convolution Matrix

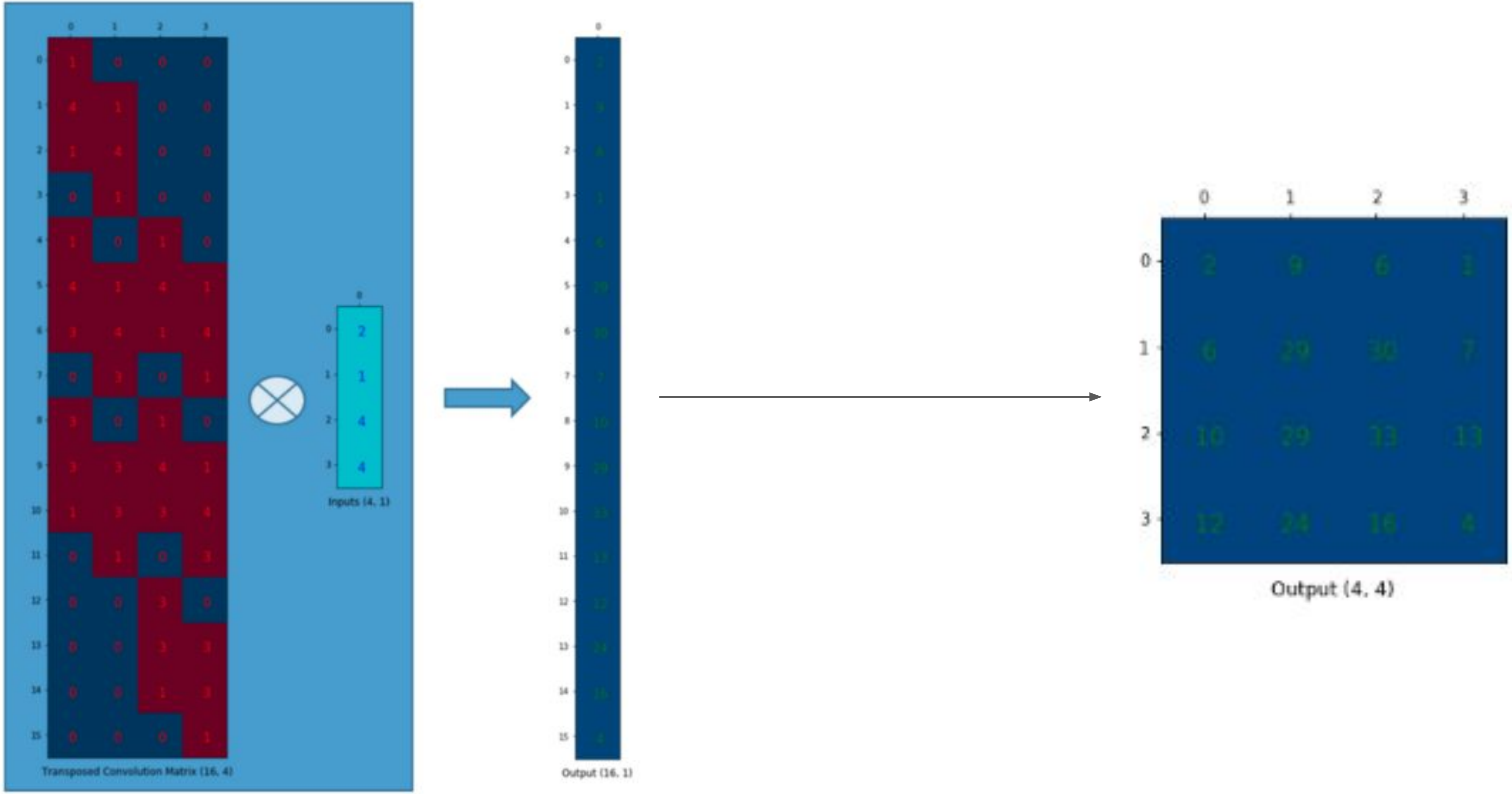


Convolution Matrix

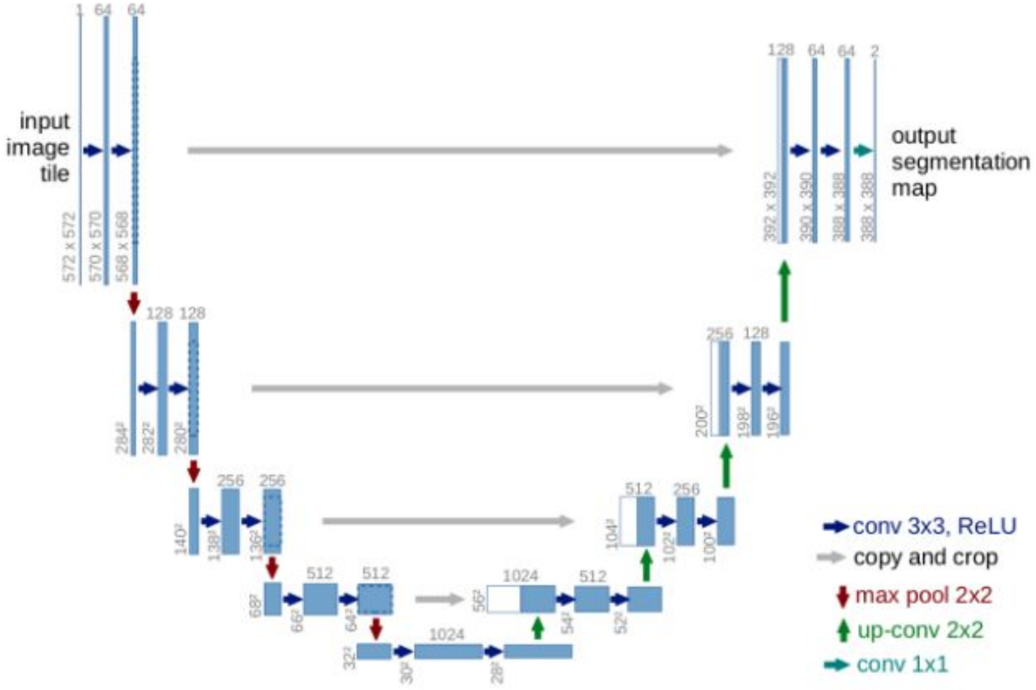


- ➔ We can convert **16x1** (4x4) input to **4x1** (2x2) output using **4x16** convolution matrix.
- ➔ So, we can also convert **4x1** (2x2) input to **16x1** (4x4) output using **16x4 'transposed'** convolution matrix.
- ➔ Let's visualize.

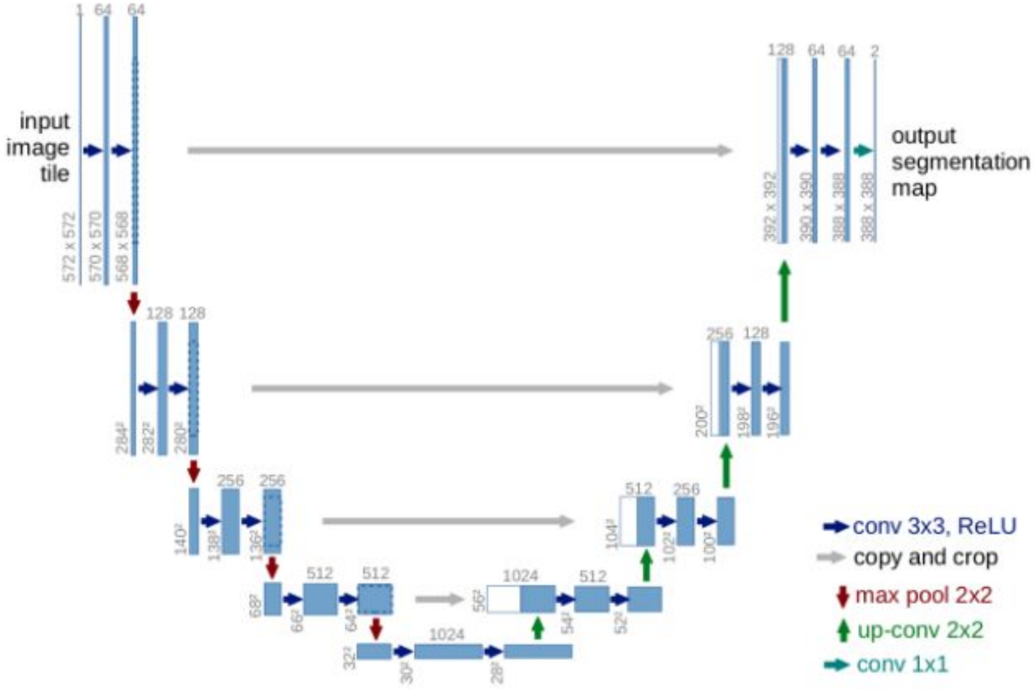
Transposed Convolution



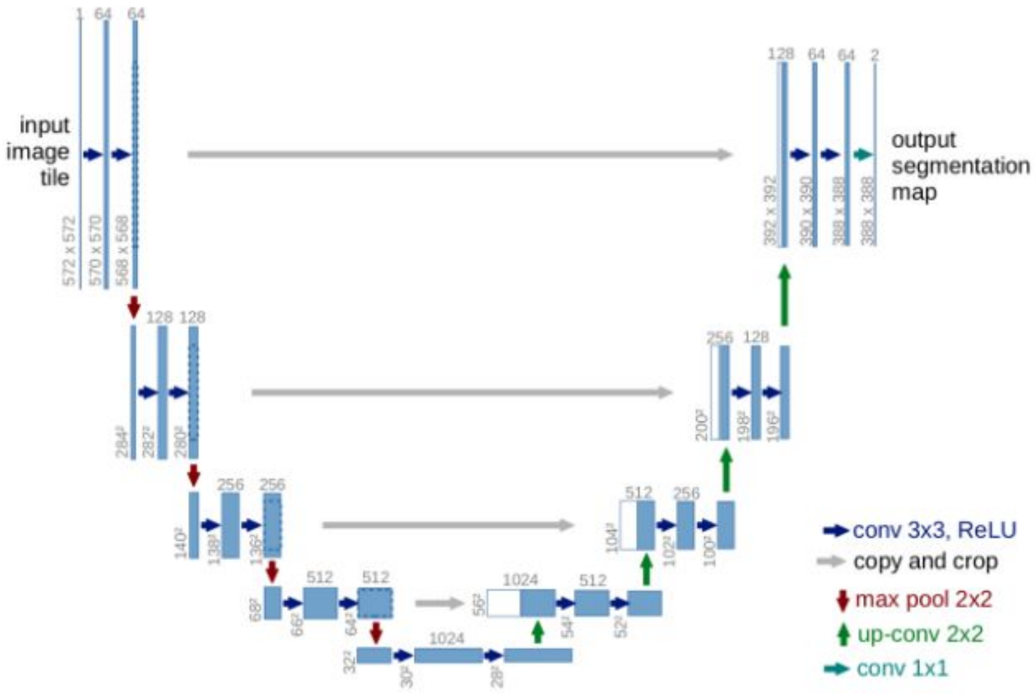
U-Net Architecture



U-Net Architecture



U-Net Architecture

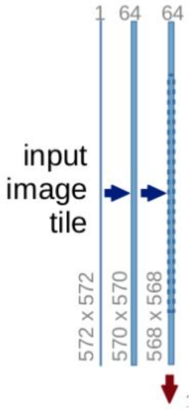


- Symmetric Architecture
- Two main parts :
  - Contractive Path
  - Expansive Path
- **Contractive Path ->** Downsampling using max pooling.
- **Expansive Path ->** Upsampling using transposed convolution.

U-Net Architecture

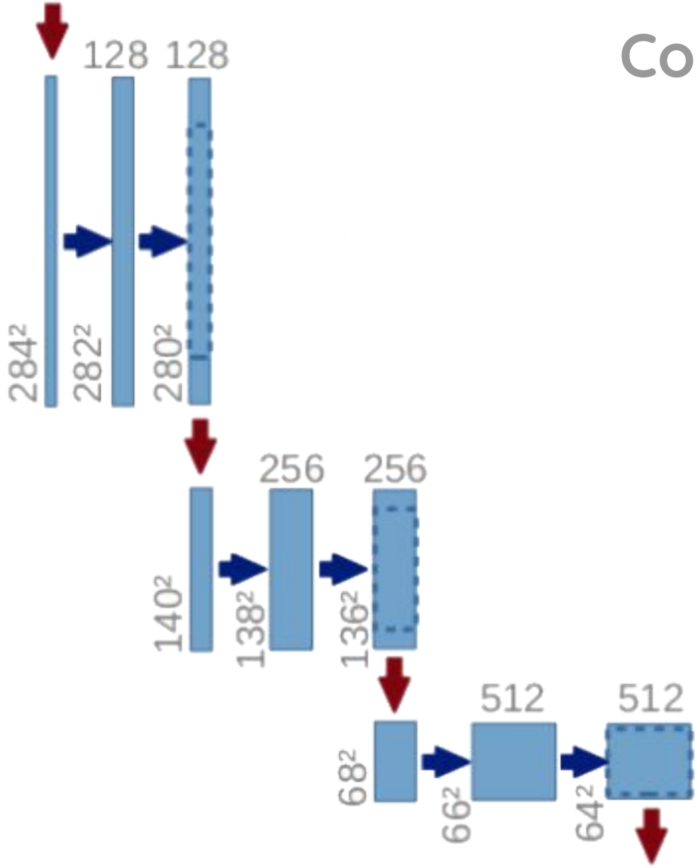
Contractive Path

Conv_Layer → Conv_Layer → Max_Pooling_Layer → Dropout(optional)



U-Net Architecture

Contractive Path



- ➔ The basic process is repeated 3 more times
- ➔ 3x3 filters with ReLu activation function are used in convolutional layers.
- ➔ 2x2 max-pooling is applied with a stride of 2.
- ➔ Output of the 2nd convolutional layer is stored at each step.

U-Net Architecture

- The bottom most layer also has 2 convolutional layers.
- No max-pooling is applied.
- The output of 2nd convolutional layer is not stored and simply passed to the next(upper) layer.

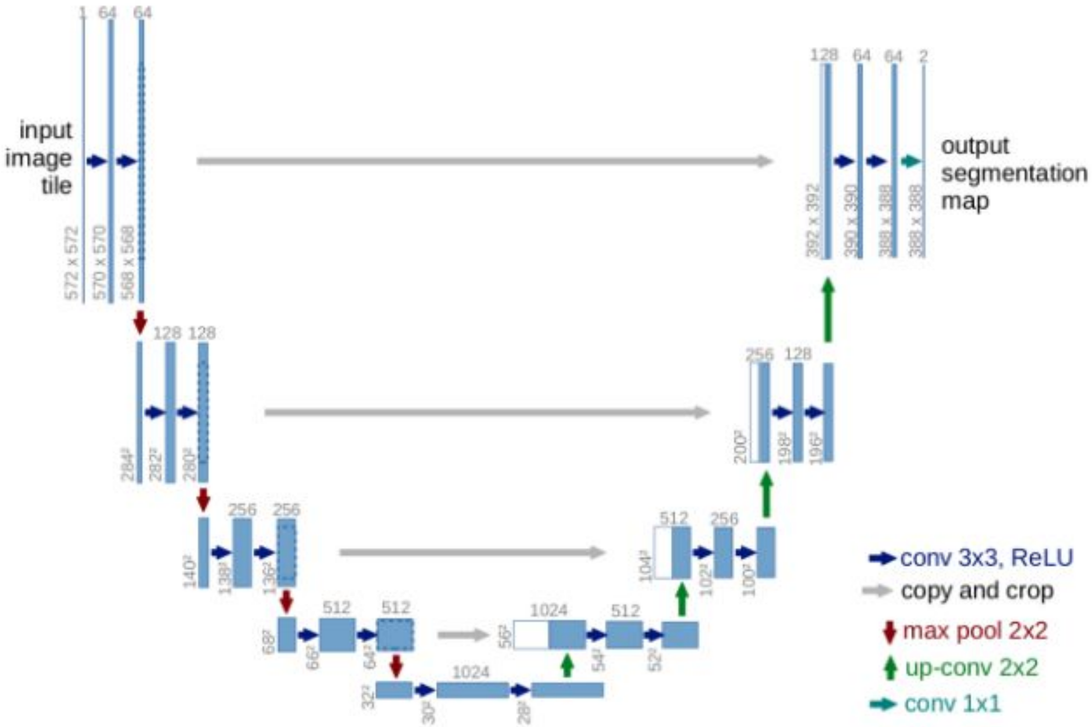
U-Net Architecture

Expansive Path

Transposed_Convolution → Copy_and_Crop → Conv_Layer → Conv_Layer → Dropout(optional)

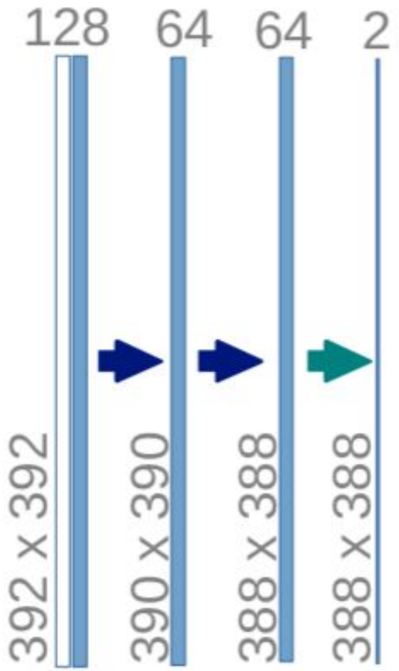
U-Net Architecture

Expansive Path



- ➔ The basic process is repeated same as in contraction path.
- ➔ In the last layer, image segmentation map is obtained.

U-Net Architecture



The image segmentation map can be reshaped according to the prediction requirements.

U-Net : Results

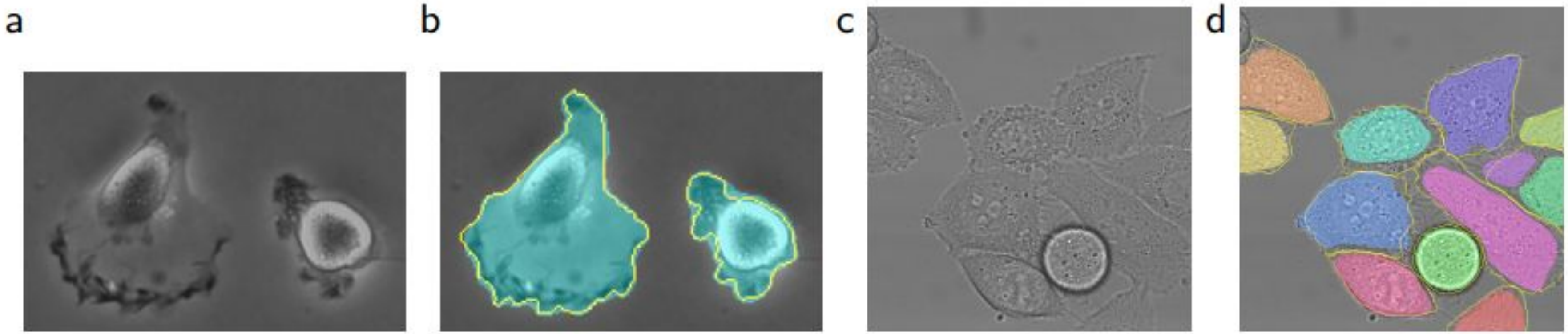


Fig. 4. Result on the ISBI cell tracking challenge. (**a**) part of an input image of the "PhC-U373" data set. (**b**) Segmentation result (cyan mask) with manual ground truth (yellow border) (**c**) input image of the "DIC-HeLa" data set. (**d**) Segmentation result (random colored masks) with manual ground truth (yellow border).

U-Net : Results

Segmentation results (IOU) on the ISBI cell tracking challenge 2015.

Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	**0.9203**	**0.7756**

U-Net : Why does it work so well?

➔ To get better precise locations, at every step of the decoder we use skip connections by concatenating the output of the transposed convolution layers with the feature maps from the Encoder at the same level.

➔ Feature maps at different scales helps in capturing the fine local details as well as the global details.

➔ After every concatenation we apply two consecutive regular convolutions so that the model can learn to assemble a more precise output.

U-Net : Limitations

- ➔ Requires a good amount of memory for storing the intermediate feature maps.
- ➔ Relies heavily on data augmentation.


Any Questions?