Billy Herzberg

william.herzberg@marquette.edu

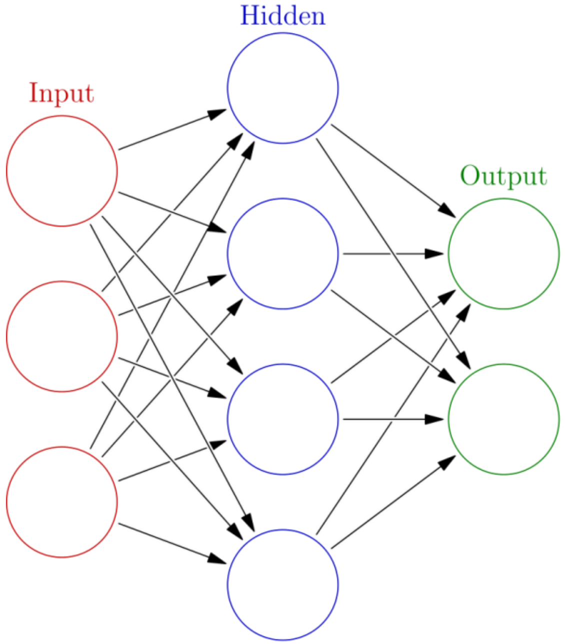
Building Convolutional Neural Networks with TensorFlow 2.x

About Me

- BS Mechanical Engineering 2018 – Marquette University
- MS Applied Statistics 2020 – Marquette University
- PhD Computational Sciences 20?? – Marquette University

- First started with deep learning in Summer 2019
- Convolutional Neural Networks for Electrical Impedance Tomography (EIT) image reconstruction
  - Post processing images with a U-Net CNN
  - Model-based, learned, iterative reconstruction

What Makes a Neural Network?



- Data Set
  - Inputs X
  - Truths Y
  - Split into training and validation
    - $X = [X_{tr} \quad X_{va}]^T$
- Model
  - Has layers
  - Has parameters θ
  - Makes predictions
  - $\hat{Y}^{(k)} = Net_{\theta}\left(X^{(k)}\right)$
- Loss Function
  - Evaluates model
  - $loss = f\left(Y, \hat{Y}\right)$
- Optimizer
  - Adjusts θ during training
  - Ex. gradient descent
- Training the model
  - Split training data into batches
  - Update θ after each batch

MARQUETTE UNIVERSITY | BE THE DIFFERENCE.

Outline

- 1D Convolution
  - Stride, Padding
- 2D Convolution
  - Single channel input
  - Multiple channel input (i.e. RGB)
  - Multiple channel output
- Choosing Weights
  - Finding changes, edges, peaks, features, etc.
- Pooling Layers
- Transposed Convolutions
  - 1D
  - 2D
  - Stride, padding
  - Multiple channel input
  - Multiple channel output
- U-Net Architecture
  - Post-processing images

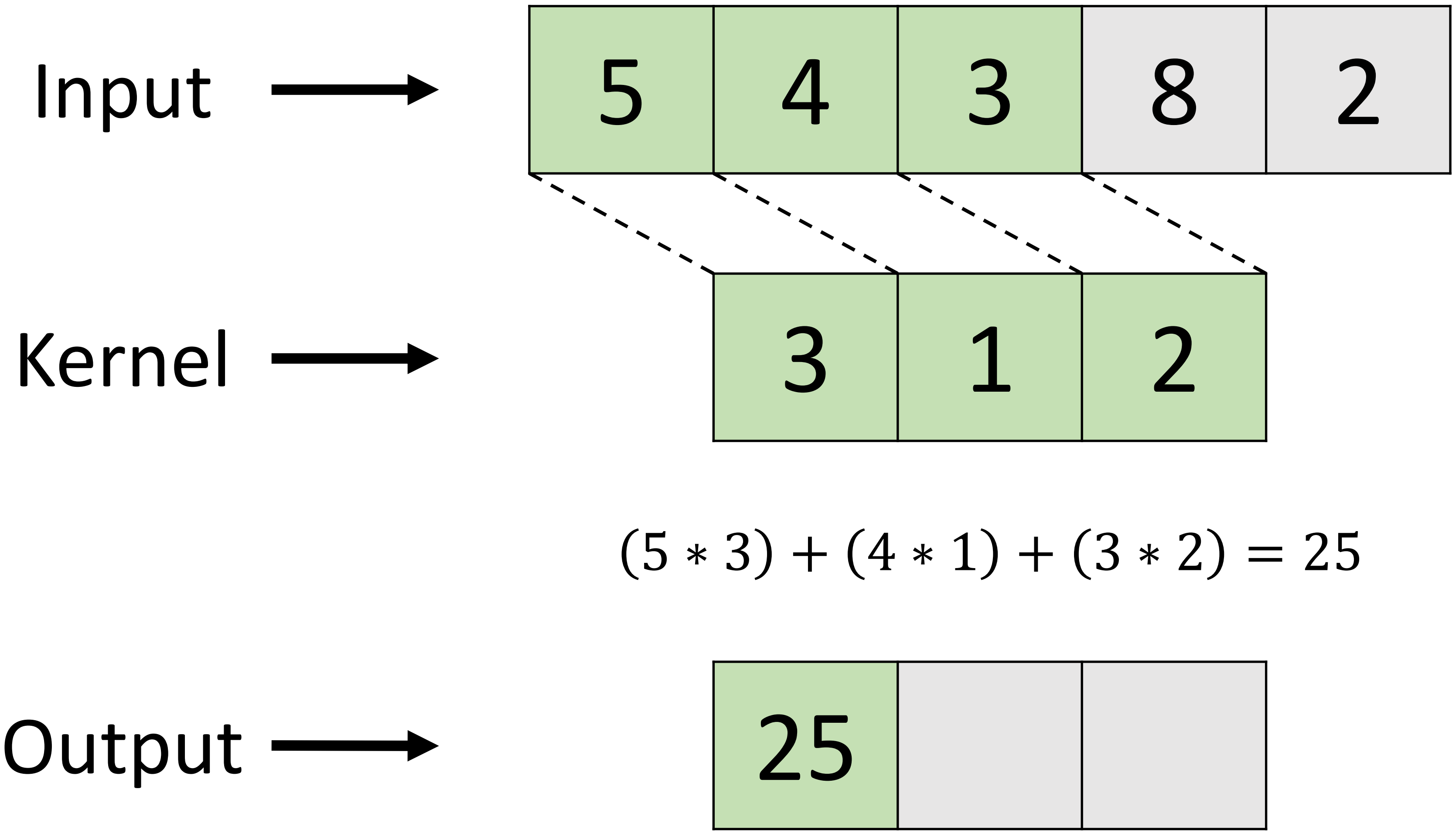
Outline

- **1D Convolution**
  - Stride, Padding
- 2D Convolution
  - Single channel input
  - Multiple channel input (i.e. RGB)
  - Multiple channel output
- Choosing Weights
  - Finding changes, edges, peaks, features, etc.
- Pooling Layers
- Transposed Convolutions
  - 1D
  - 2D
  - Stride, padding
  - Multiple channel input
  - Multiple channel output
- U-Net Architecture
  - Post-processing images

1D Convolutions

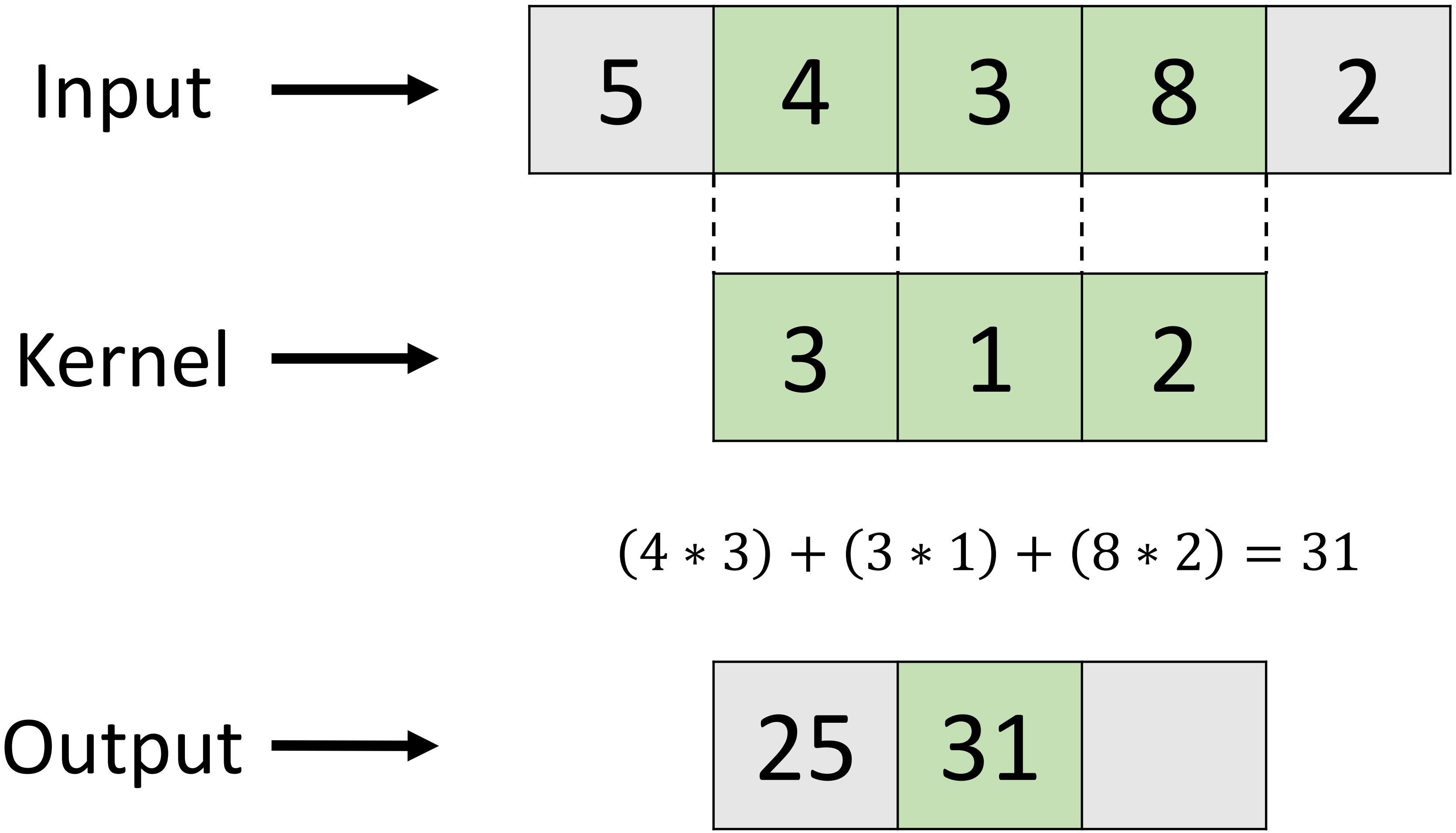
Input	→	5	4	3	8	2
Kernel	→		3	1	2	
Output	→					

1D Convolutions

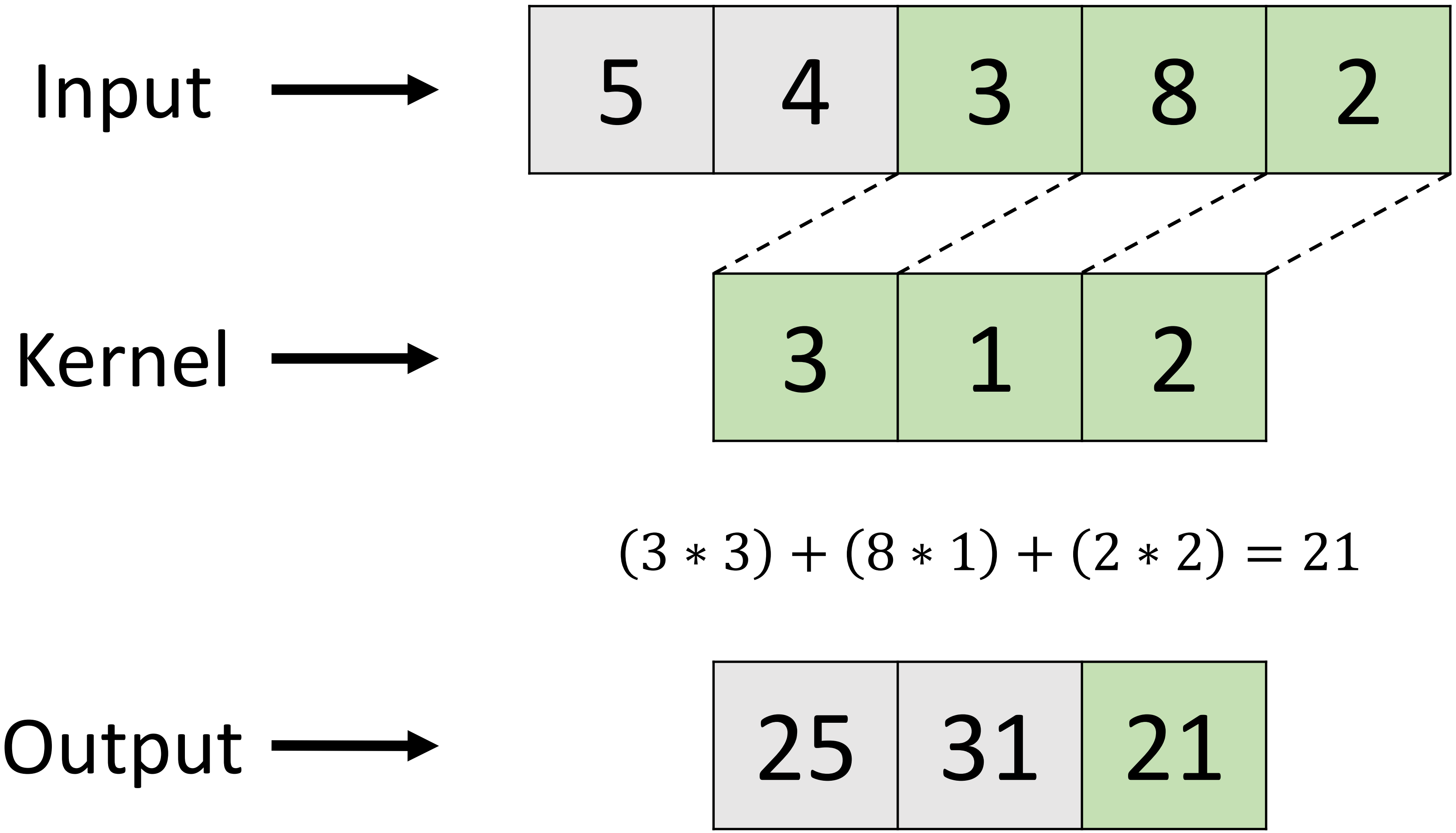


$$(5 * 3) + (4 * 1) + (3 * 2) = 25$$

1D Convolutions



1D Convolutions

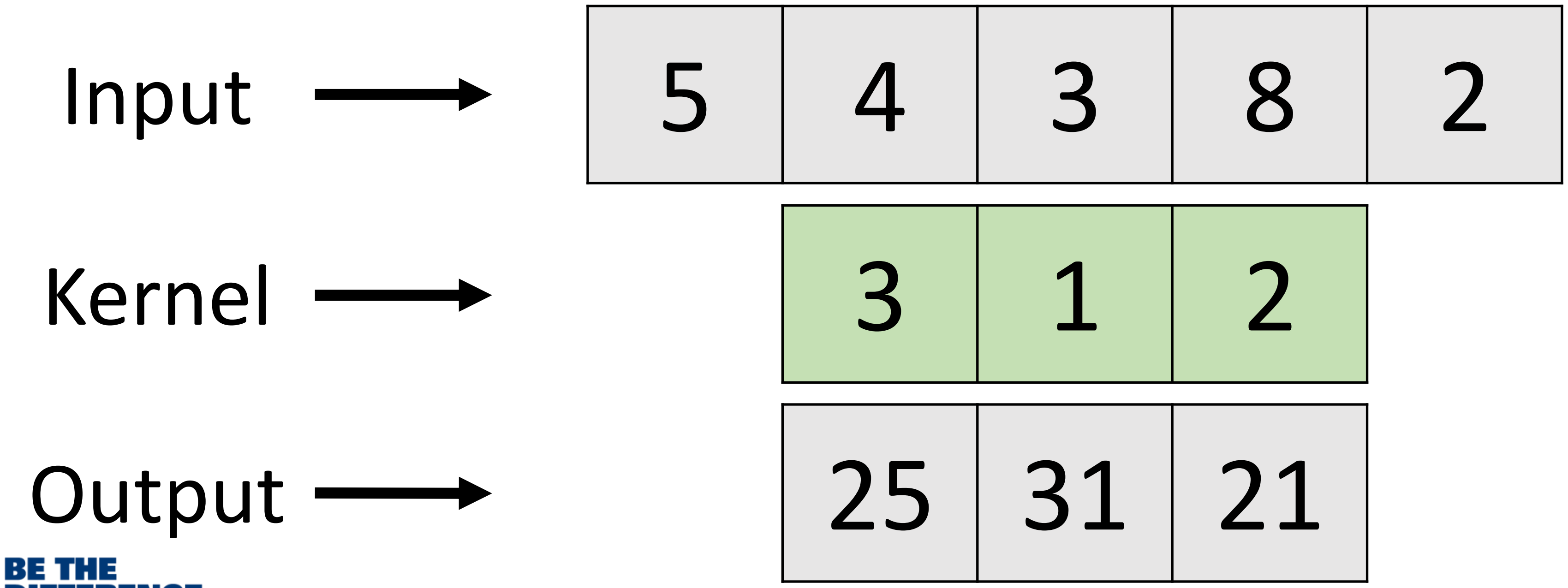


$$(3 * 3) + (8 * 1) + (2 * 2) = 21$$

1D Convolutions

- **Stride** is the distance the kernel moves for each output calculation
- **Padding** can be used to get the desired size for the output

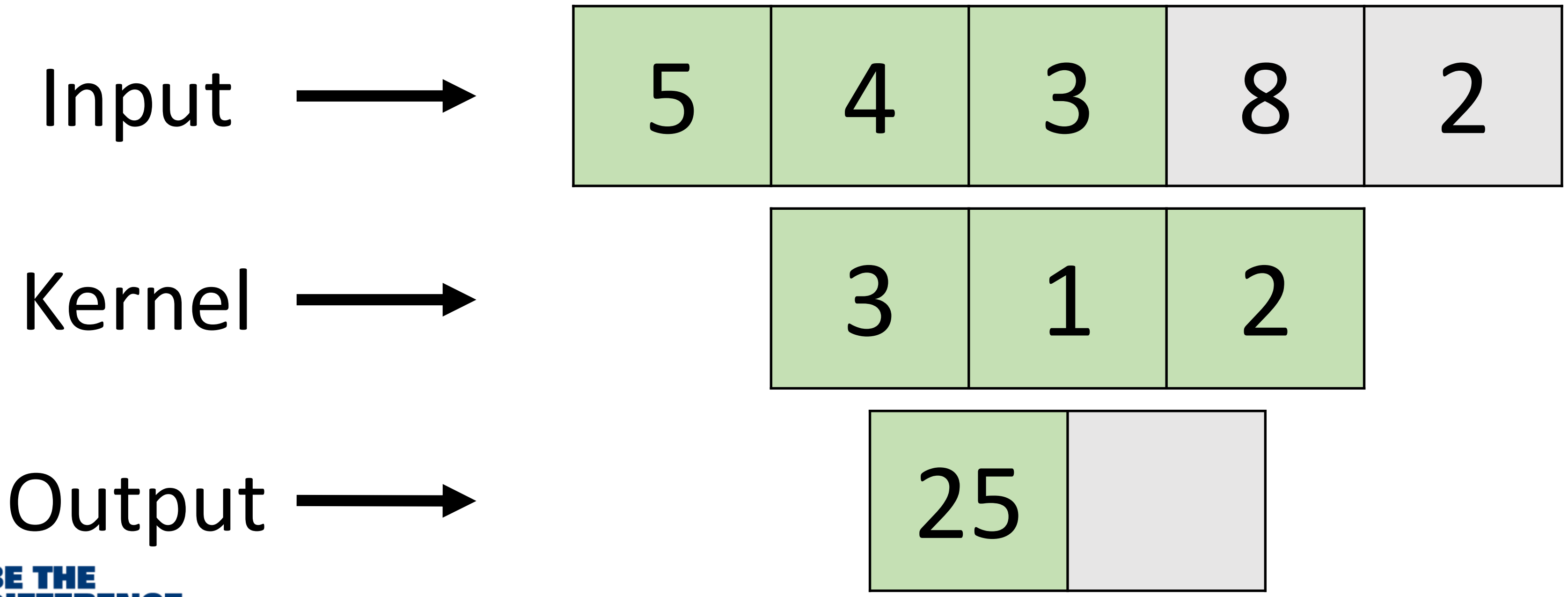
Stride=1, Padding="None"



1D Convolutions

- **Stride** is the distance the kernel moves for each output calculation
- Padding can be used to get the desired size for the output

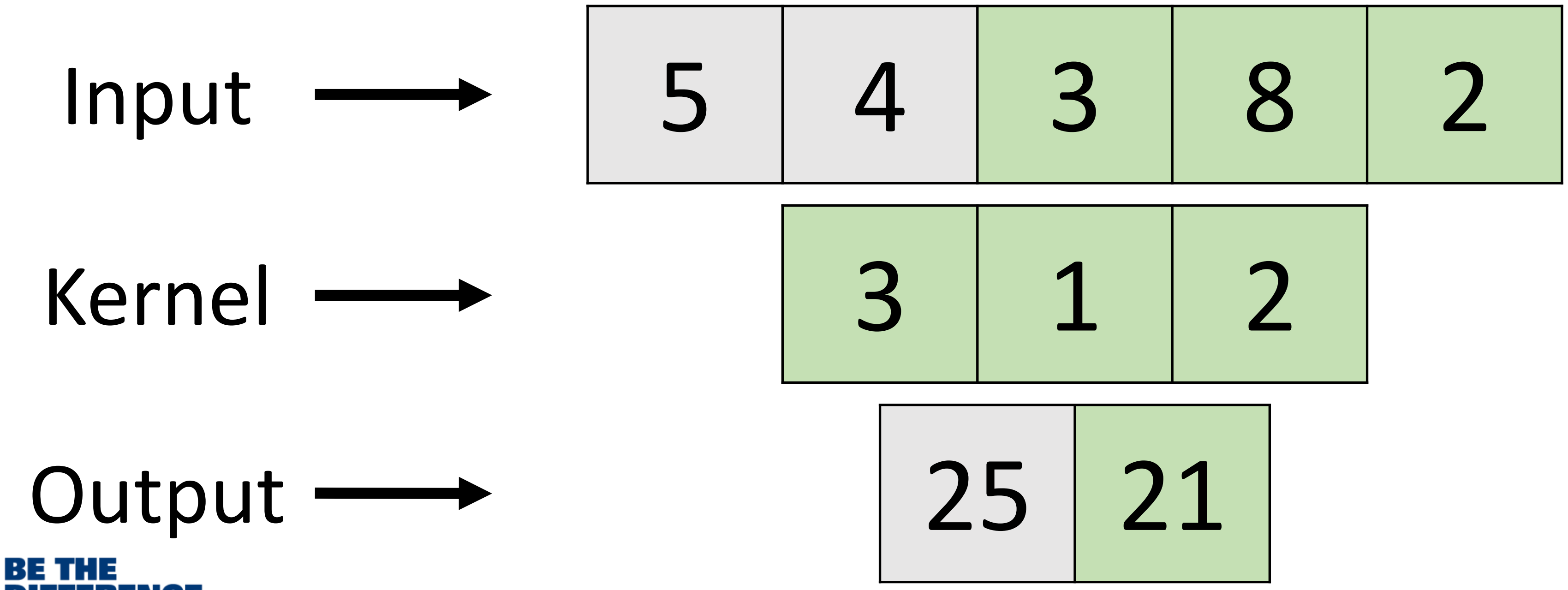
Stride=2, Padding="None"



1D Convolutions

- **Stride** is the distance the kernel moves for each output calculation
- Padding can be used to get the desired size for the output

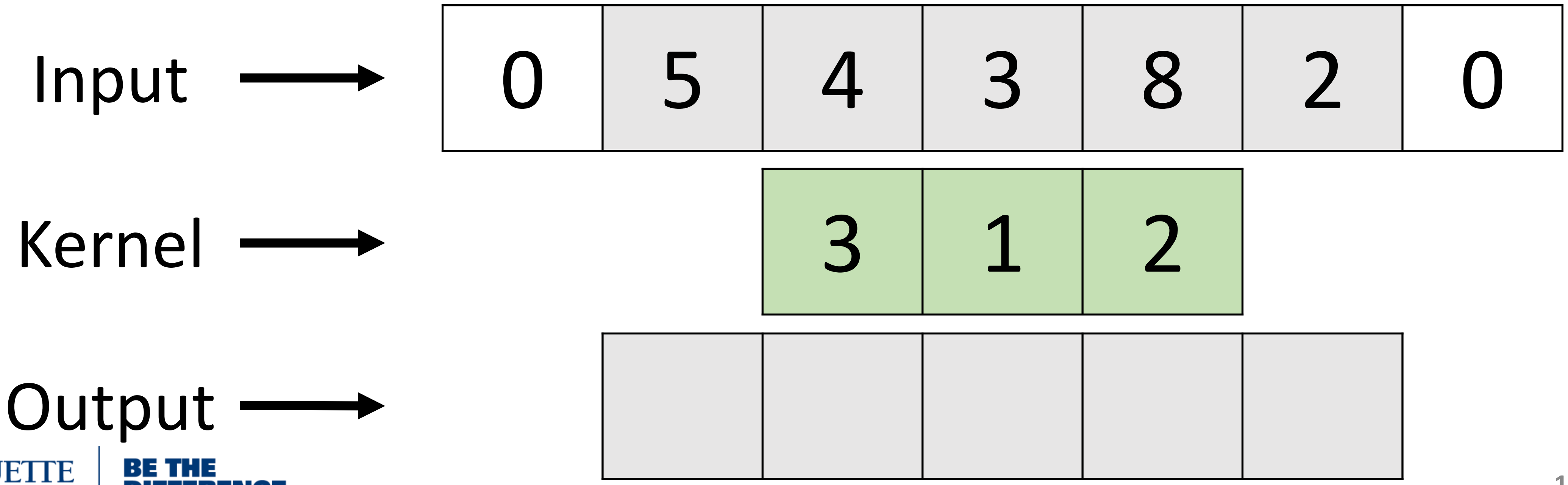
Stride=2, Padding="None"



1D Convolutions

- Stride is the distance the kernel moves for each output calculation
- **Padding** can be used to get the desired size for the output

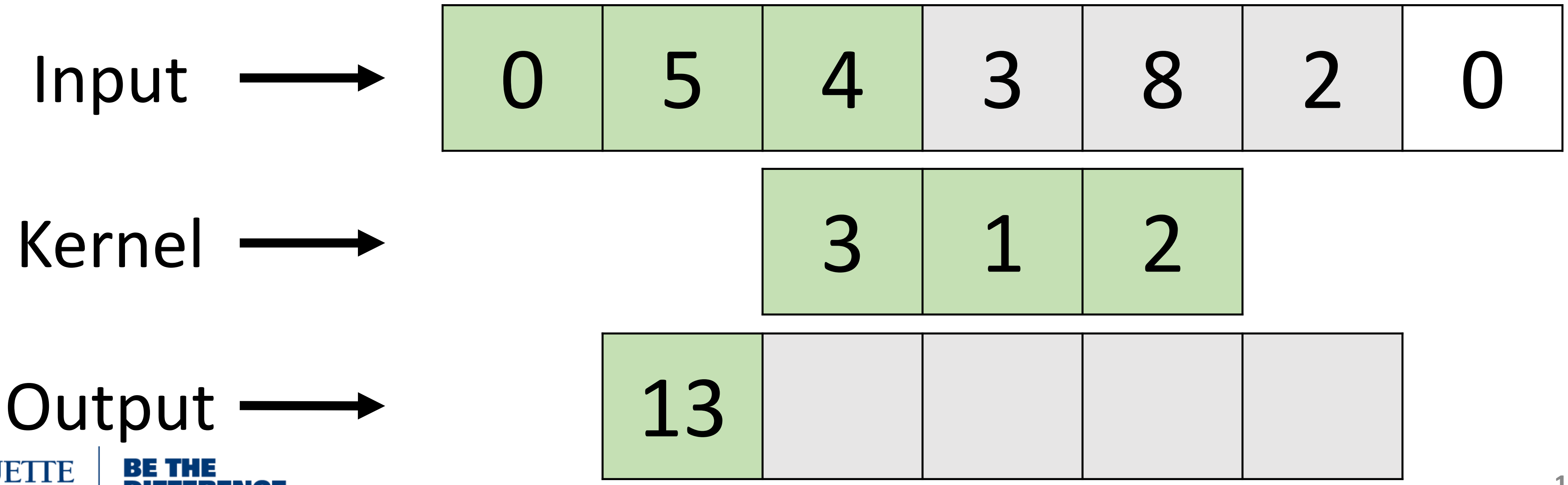
Stride=1, **Padding="Same"**



1D Convolutions

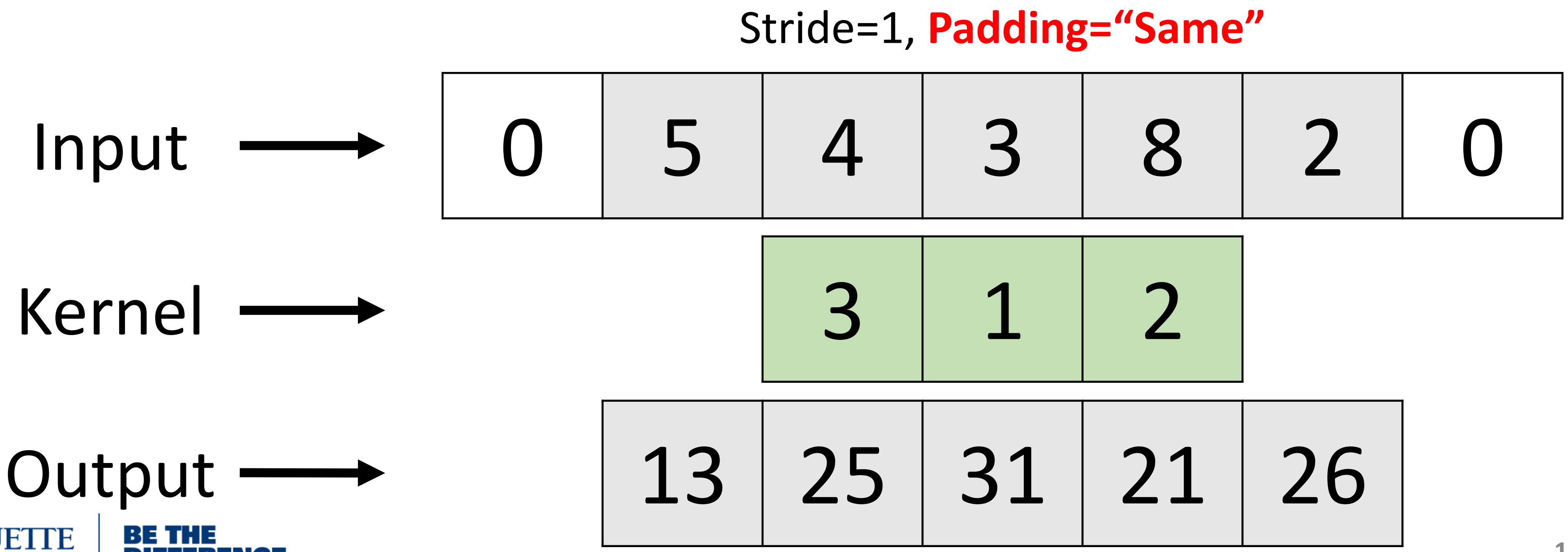
- Stride is the distance the kernel moves for each output calculation
- **Padding** can be used to get the desired size for the output

Stride=1, **Padding="Same"**



1D Convolutions

- Stride is the distance the kernel moves for each output calculation
- **Padding** can be used to get the desired size for the output



Questions?

- 1D Convolutions
- Stride
- Padding

- Up next: 2D Convolutions

Outline

- 1D Convolution
  - Stride, Padding
- **2D Convolution**
  - Single channel input
  - Multiple channel input (i.e. RGB)
  - Multiple channel output
- Choosing Weights
  - Finding changes, edges, peaks, features, etc.
- Pooling Layers
- Transposed Convolutions
  - 1D
  - 2D
  - Stride, padding
  - Multiple channel input
  - Multiple channel output
- U-Net Architecture
  - Post-processing images

2D Convolutions

Input

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

Kernel

1	0	1
0	2	1
0	0	2

Output

2D Convolutions

Input

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

Kernel

1	0	1
0	2	1
0	0	2

Output

18		

$$(5*1)+(4*0)+(3*1)+\cdots$$
$$(8*0)+(1*2)+(2*1)+\cdots$$
$$(1*0)+(7*0)+(3*2)=18$$

2D Convolutions

Input

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

Kernel

1	0	1
0	2	1
0	0	2

Output

18	24	

$$(4*1)+(3*0)+(8*1)+\cdots$$
$$(1*0)+(2*2)+(4*1)+\cdots$$
$$(7*0)+(3*0)+(2*2)=24$$

MARQUETTE UNIVERSITY | BE THE DIFFERENCE.

2D Convolutions

Input

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

Kernel

1	0	1
0	2	1
0	0	2

Output

18	24	26
27	14	21
26	14	19

$$(3 * 1) + (2 * 0) + (6 * 1) + \cdots$$
$$(0 * 0) + (1 * 2) + (4 * 1) + \cdots$$
$$(7 * 0) + (3 * 0) + (2 * 2) = 19$$

2D Convolutions

- Again, stride and padding can be used to manipulate output size

Input

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

Kernel

1	0	1
0	2	1
0	0	2

Output

Stride=(1,1)
Padding="None"

18	24	26
27	14	21
26	14	19

2D Convolutions

- Again, stride and padding can be used to manipulate output size

Input

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

Kernel

1	0	1
0	2	1
0	0	2

Output

Stride=(2,2)
Padding="None"

18	26
26	19

2D Convolutions

- Again, stride and padding can be used to manipulate output size

Input

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

Kernel

1	0	1
0	2	1
0	0	2

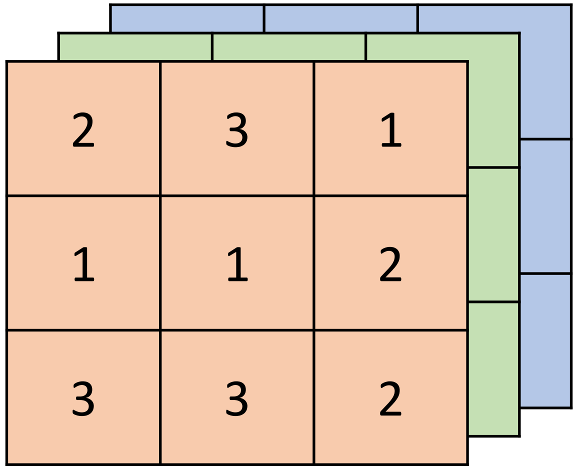
Output

Stride=(1,1)
Padding="Same"

16	15	22	20	4
35	18	24	26	10
18	27	14	21	16
17	26	14	19	10
11	11	22	12	5

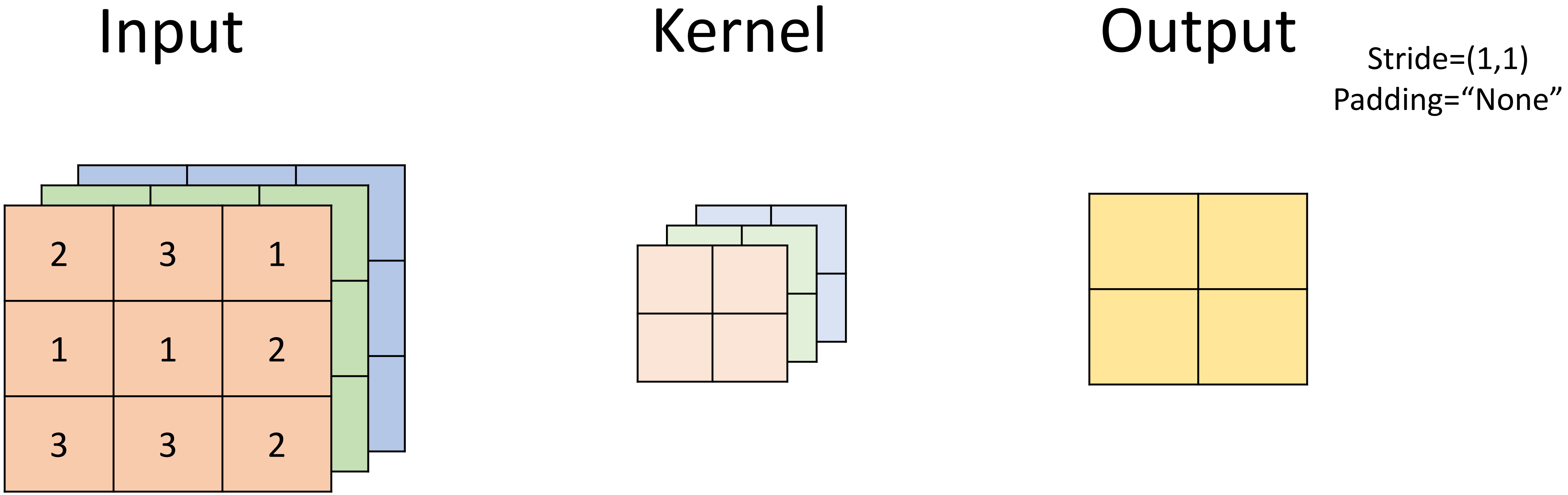
2D Convolutions

- The dimension of the convolution is determined by the stride dimension, NOT the input or output dimension
- RGB images have three channels (one each for Red, Green, and Blue) and are therefor 3D arrays. Still, 2D convolutions are used

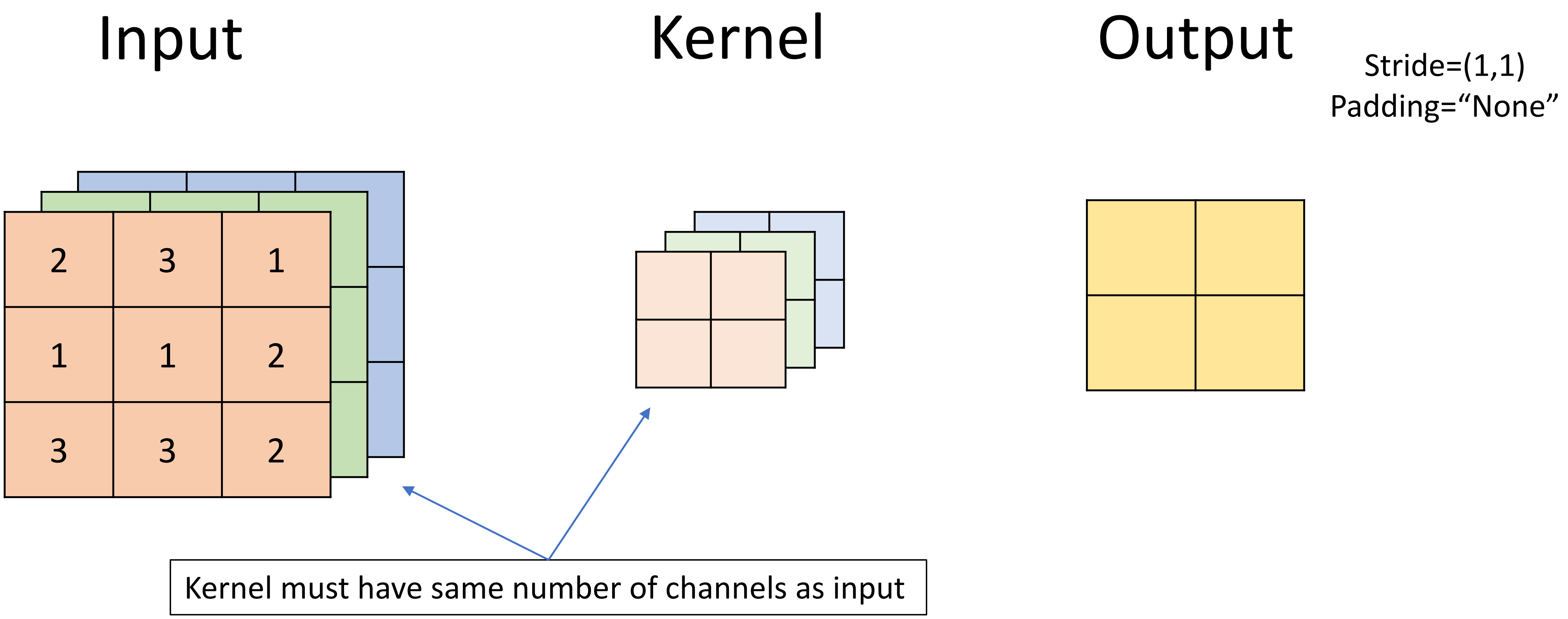


$$= \begin{bmatrix} 2 & 3 & 1 \\ 1 & 1 & 2 \\ 3 & 3 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

2D Convolutions

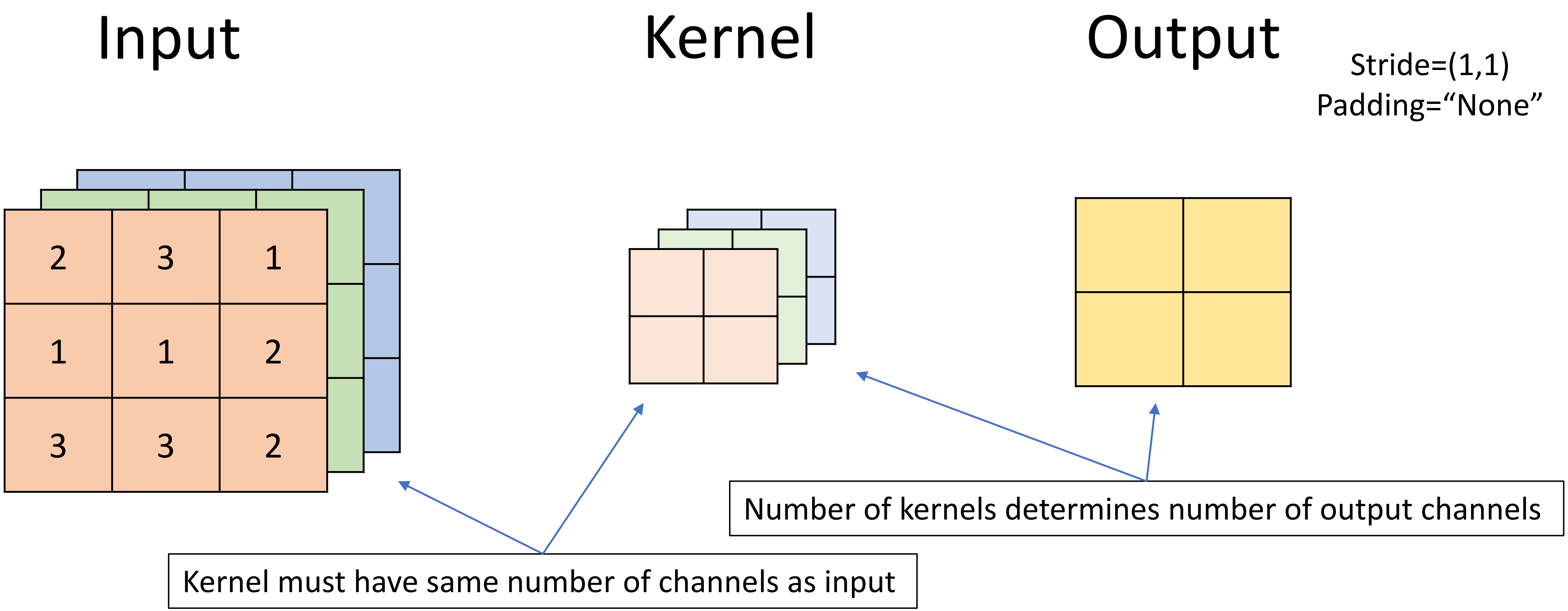


2D Convolutions

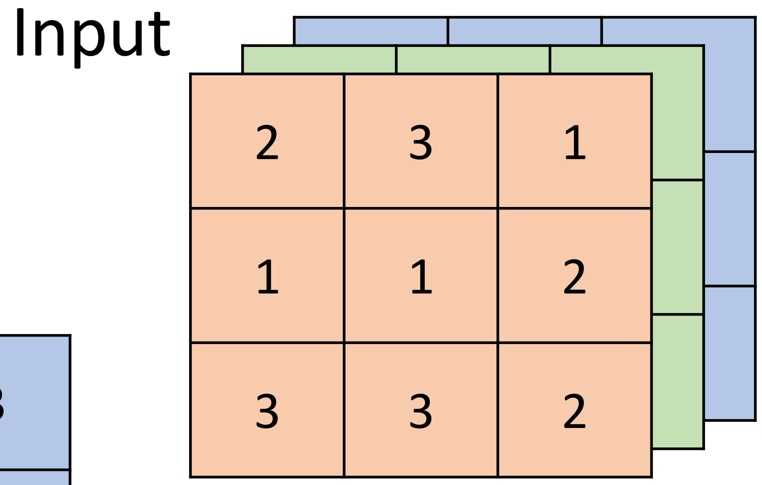


MARQUETTE UNIVERSITY | BE THE DIFFERENCE.

2D Convolutions



2D Convolutions

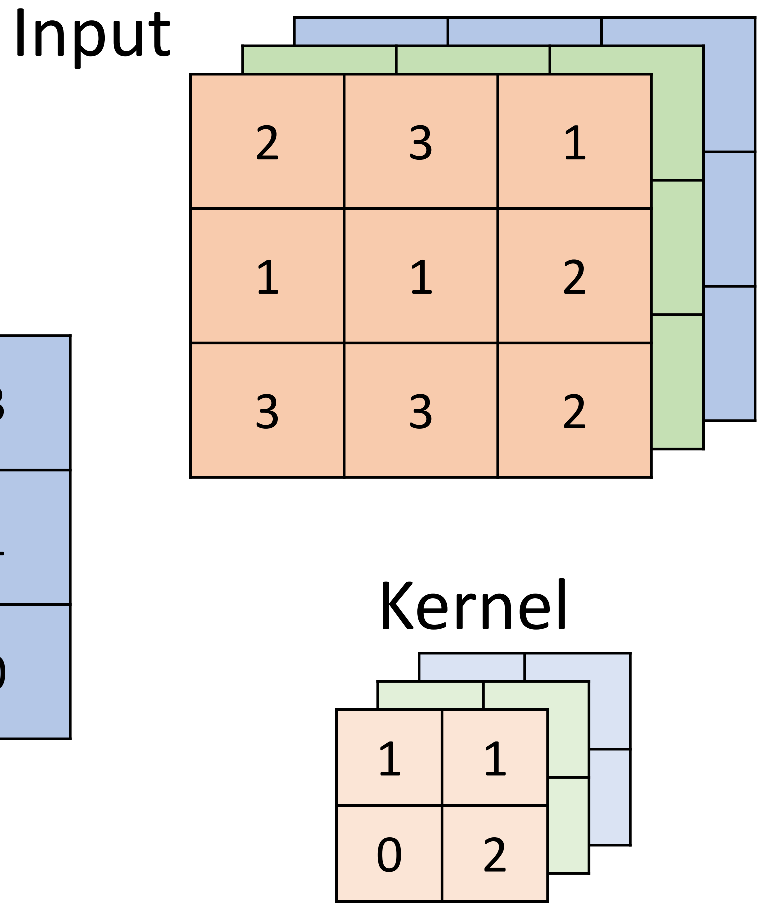


2	3	1
1	1	2
3	3	2

1	0	1
0	2	1
0	0	2

2	1	3
1	2	1
1	0	0

2D Convolutions



2	3	1
1	1	2
3	3	2

1	0	1
0	2	1
0	0	2

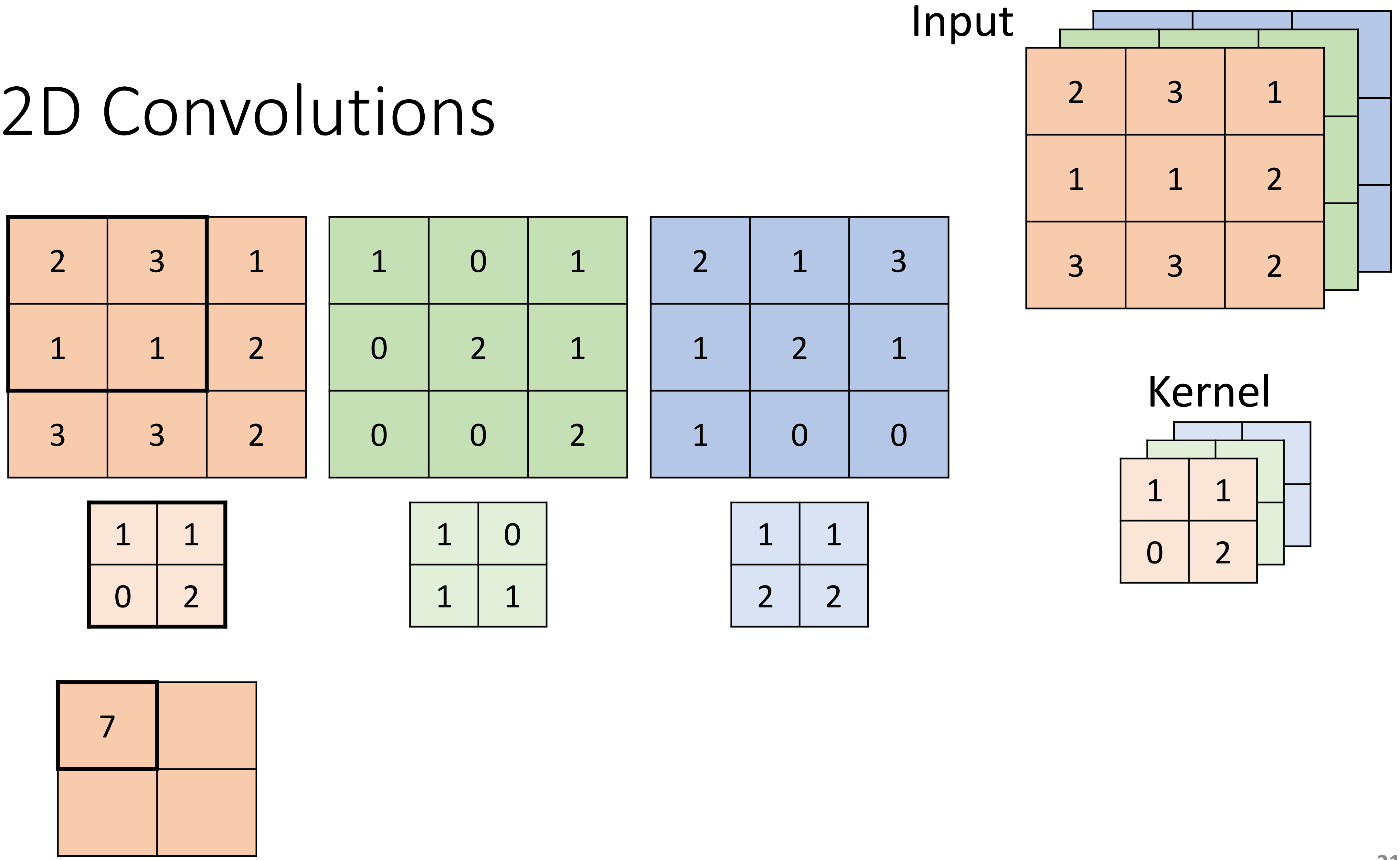
2	1	3
1	2	1
1	0	0

1	1
0	2

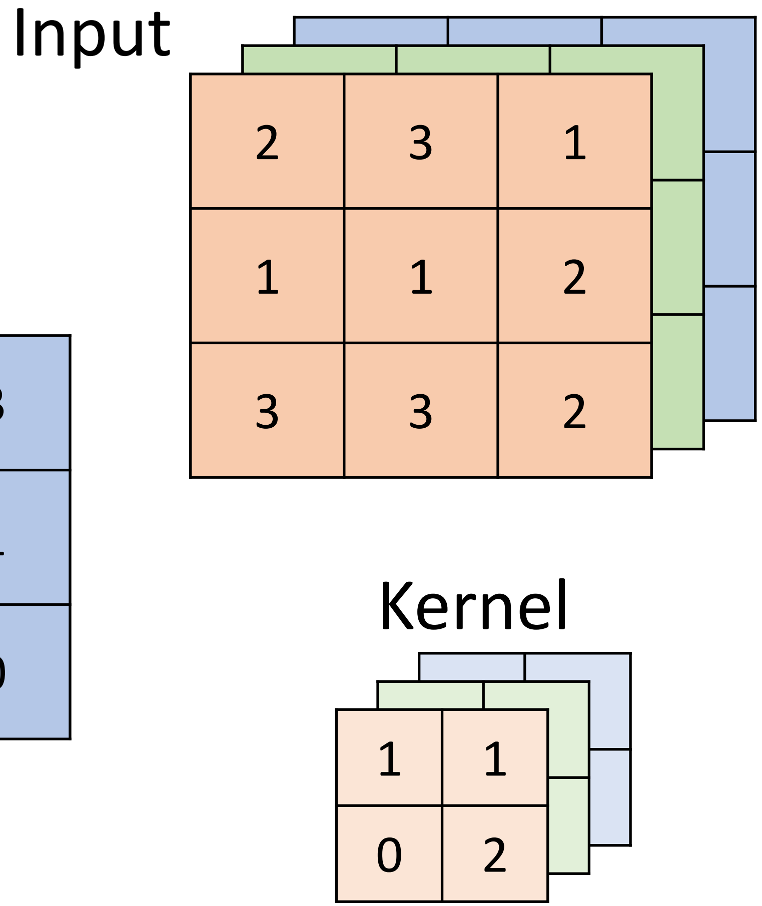
1	0
1	1

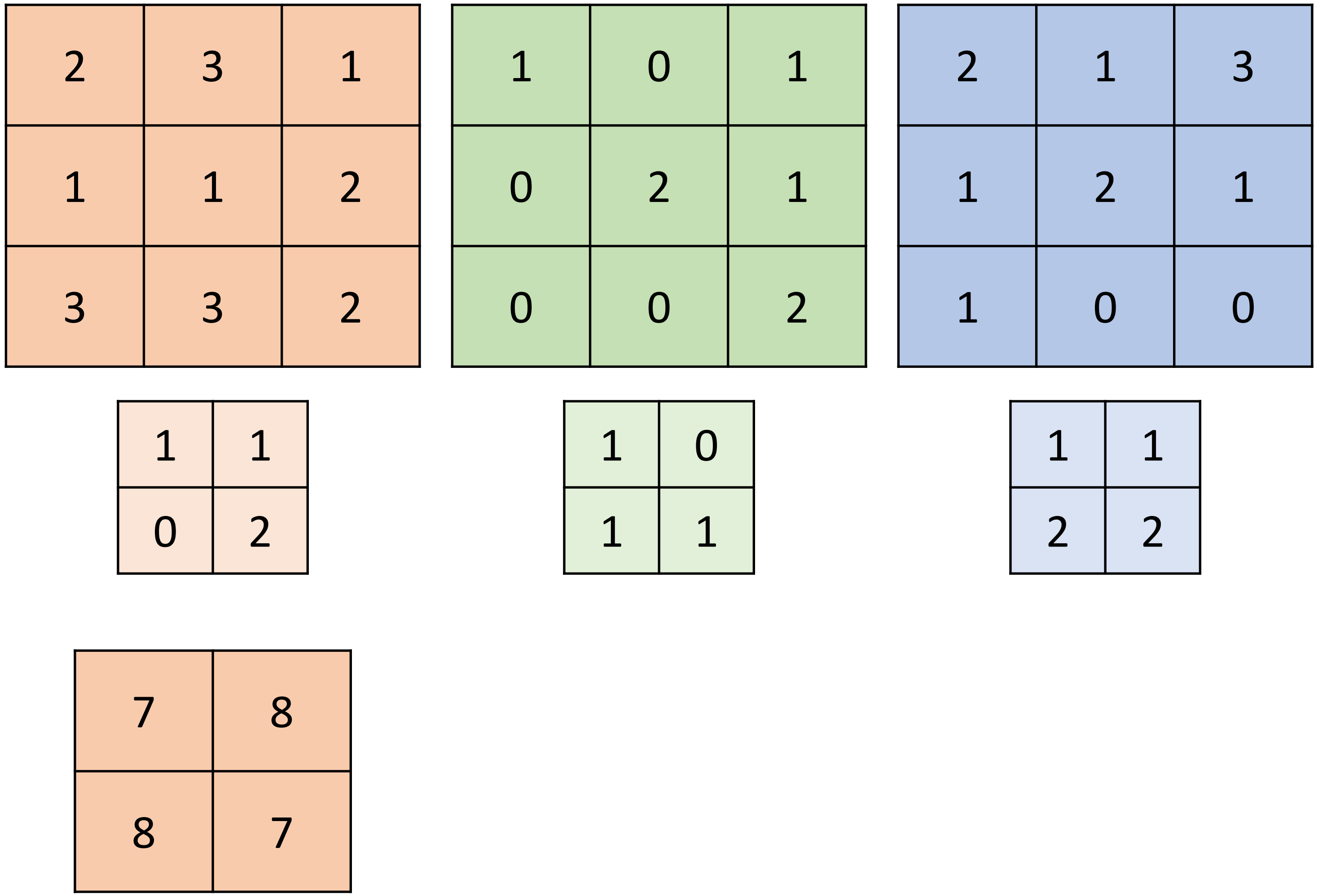
1	1
2	2

2D Convolutions

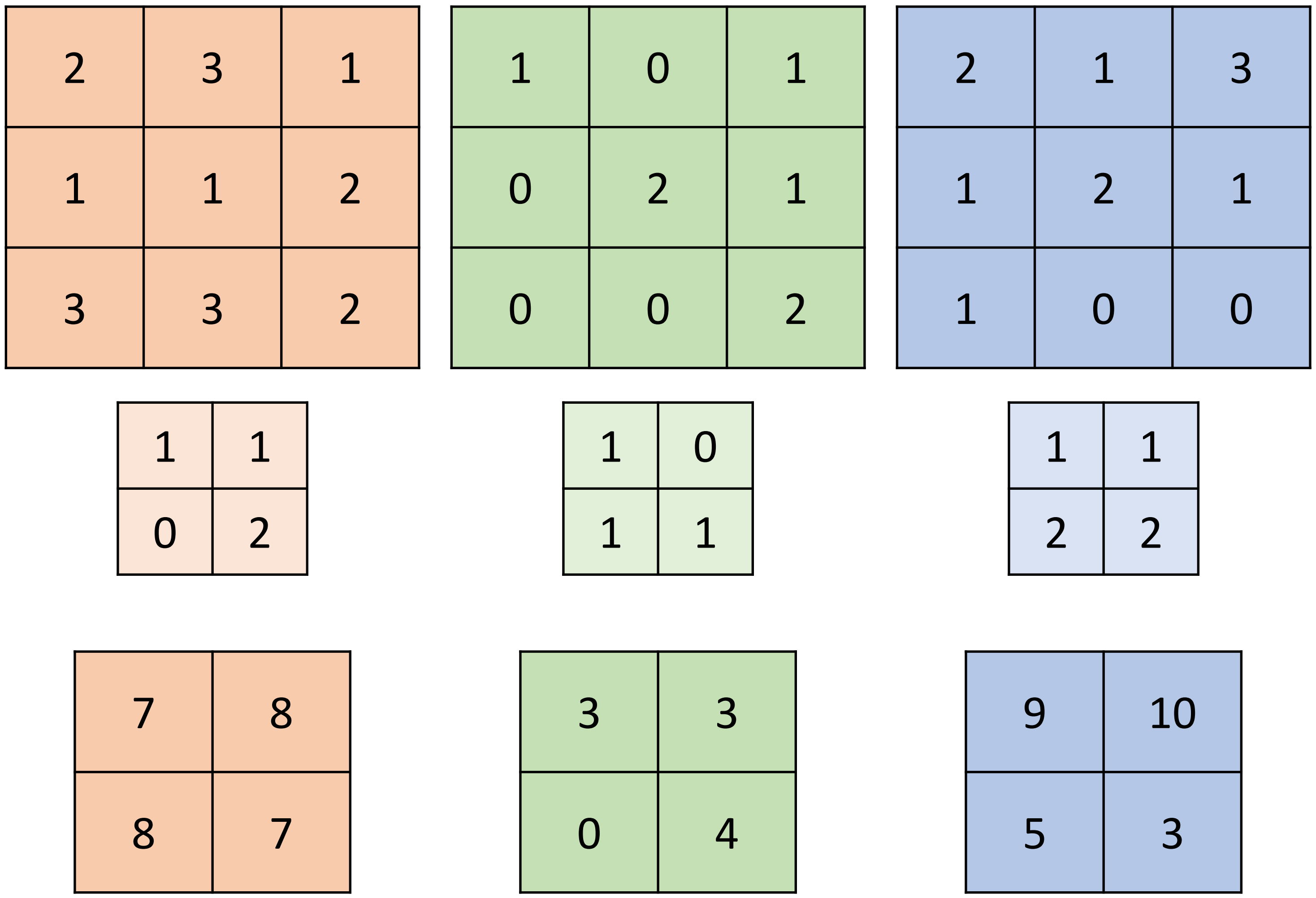


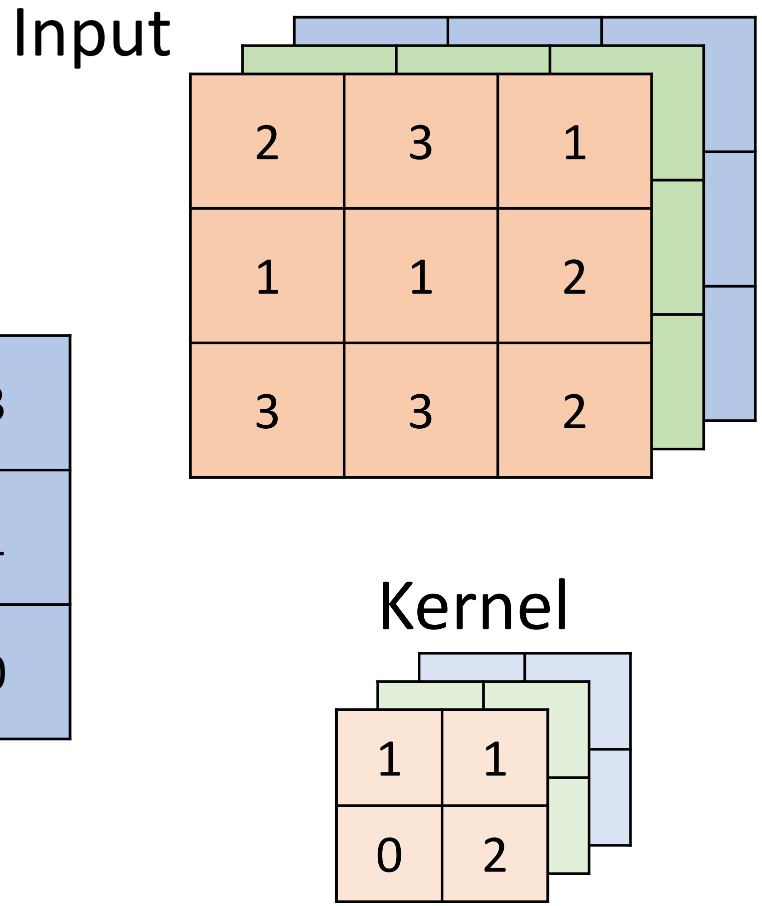
2D Convolutions



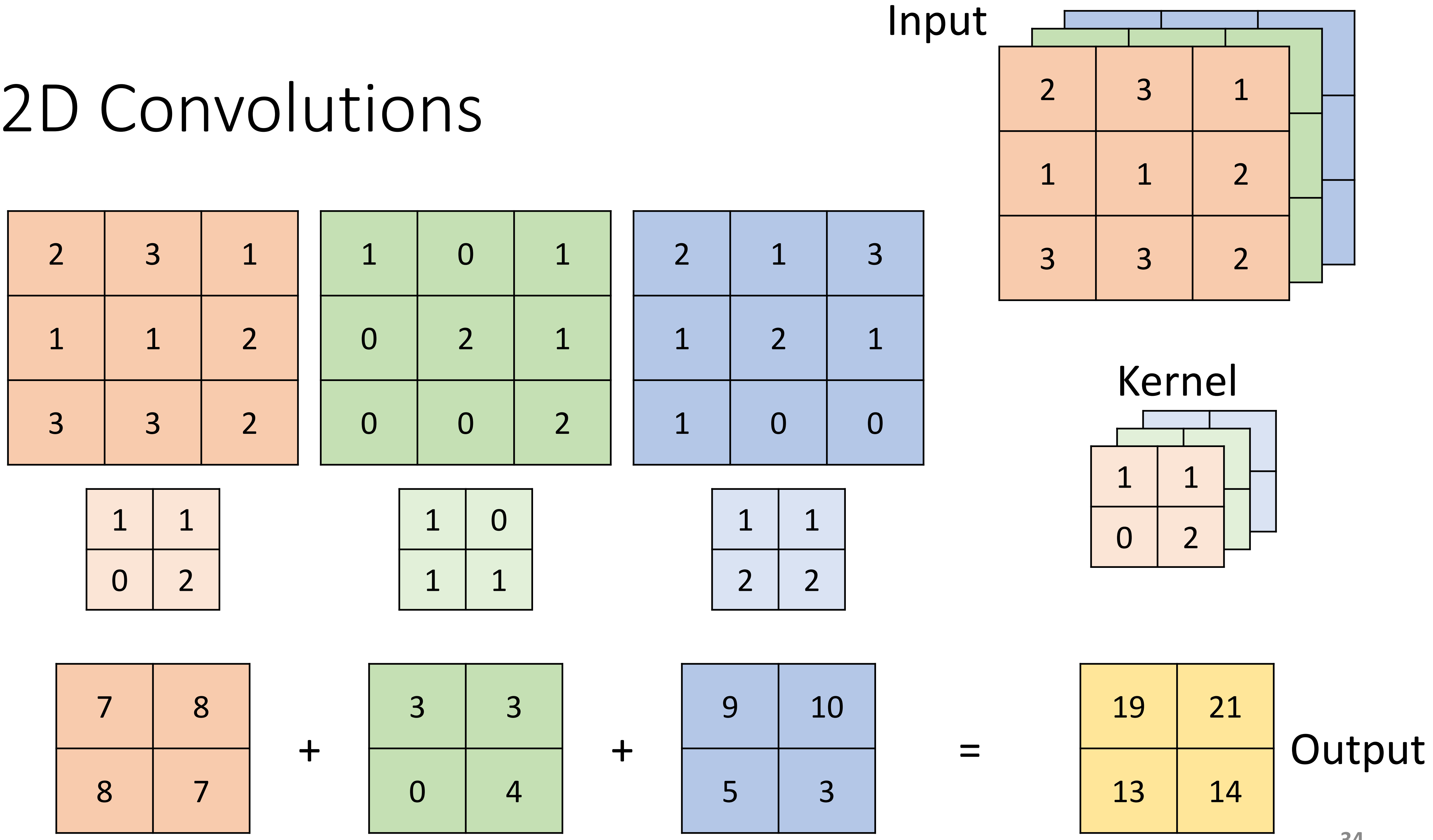


2D Convolutions





2D Convolutions



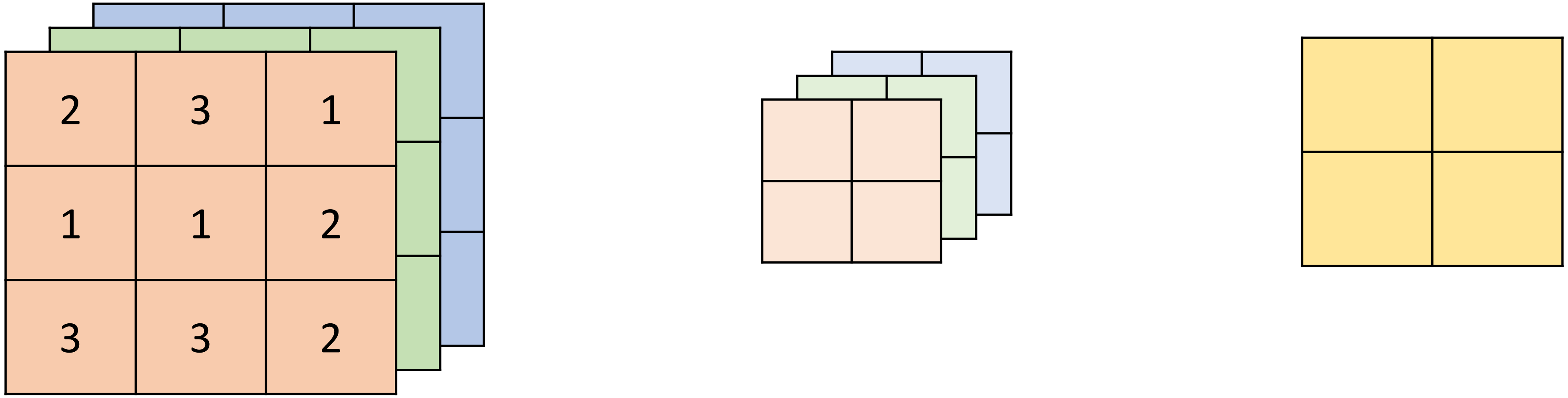
2D Convolutions

Input

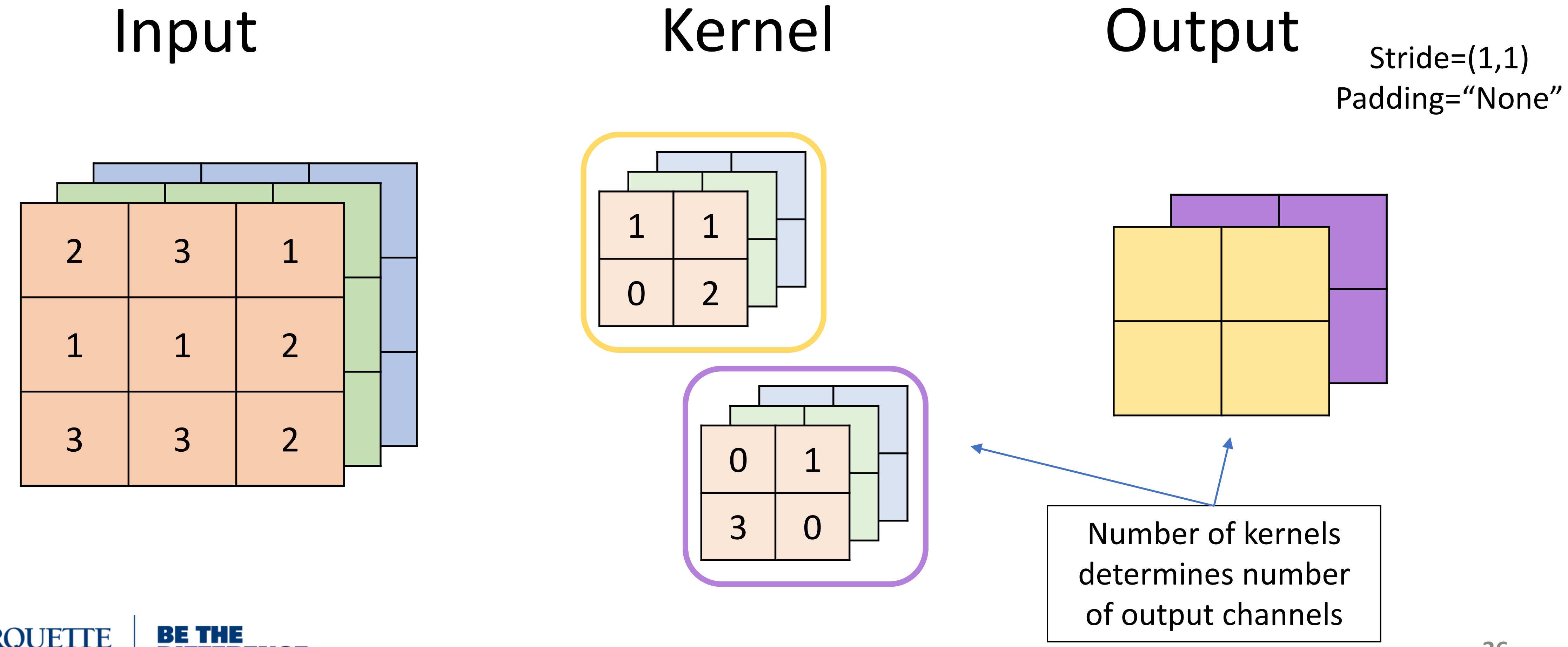
Kernel

Output

Stride=(1,1)
Padding="None"



2D Convolutions



Questions?

- 1D Convolutions
- 2D Convolutions
- Multiple input channels
- Multiple output channels

- Up next: Choosing weights

Outline

- 1D Convolution
  - Stride, Padding
- 2D Convolution
  - Single channel input
  - Multiple channel input (i.e. RGB)
  - Multiple channels on output
- **Choosing Weights**
  - Finding changes, edges, peaks, features, etc.
- Pooling Layers
- Transposed Convolutions
  - 1D
  - 2D
  - Stride, padding
  - Multiple channel input
  - Multiple channel output
- U-Net Architecture
  - Post-processing images

Choosing Weights

- Can we choose a kernel so that the output is the change of the input?

Input

0	0	1	0	2	1	2	0

$+1$

Kernel

w_1	w_2

Output

0	1	-1	2	-1	1	-2

Choosing Weights

- We can! Selecting $(w_1, w_2) = (-1, 1)$ will output the change

Input

0	0	1	0	2	1	2	0

Kernel

-1	1

Output

0	1	-1	2	-1	1	-2

Choosing Weights

- Other kernels can be used to detect **peaks**, valleys, features, etc.
- Same idea works with multidimensional kernels
- Note that padding and stride=1 were used here

Input	0	0	1	0	2	1	2	0	0
Kernel				-1	2	-1			
Output		-1	2	-3	3	-2	3	-2	

Choosing Weights

- So what kernels do we pick for our CNN?
- Luckily, we don't need to pick, the kernels are learned during training!

- For each convolutional layer, how many parameters need to be learned by the algorithm?

$$\underbrace{(num\ elements\ in\ kernel) * (num\ kernels)}_{\text{weights}} + \underbrace{(num\ kernels)}_{\text{biases}}$$

Questions?

- 1D Convolutions
- 2D Convolutions
- Multiple Channels
- Choosing weights to highlight features

- Up next: Pooling layers and a classification example

Outline

- 1D Convolution
  - Stride, Padding
- 2D Convolution
  - Single channel input
  - Multiple channel input (i.e. RGB)
  - Multiple channels on output
- Choosing Weights
  - Finding changes, edges, peaks, features, etc.
- **Pooling Layers**
- Transposed Convolutions
  - 1D
  - 2D
  - Stride, padding
  - Multiple channel input
  - Multiple channel output
- U-Net Architecture
  - Post-processing images

Pooling Layers

- Pooling layers reduce the resolution (height and/or width)
- Extract dominant features that are positionally invariant
- Reduce noise

- Slide a window over the input and take the max (or mean, etc.) of the input values as the output

- Max pooling is typically used in CNN’s because it disregards noise better than other options

Max Pooling Layers

Input

1	2	4	1
7	3	2	6
4	0	1	4
1	7	3	2

Output

7		

Max Pooling Layers

Input

1	2	4	1
7	3	2	6
4	0	1	4
1	7	3	2

Output

7	4	

Max Pooling Layers

Input

1	2	4	1
7	3	2	6
4	0	1	4
1	7	3	2

Output

7	4	6

Max Pooling Layers

Input

1	2	4	1
7	3	2	6
4	0	1	4
1	7	3	2

Output

7	4	6
7	3	6
7	7	4

Max Pooling Layers

- Pooling layers are defined by kernel size and stride
- There are no learned parameters in a pooling layer

Kernel Size=(2,2)
Stride=(1,1)

Input

1	2	4	1
7	3	2	6
4	0	1	4
1	7	3	2

Output

7	4	6
7	3	6
7	7	4

Questions?

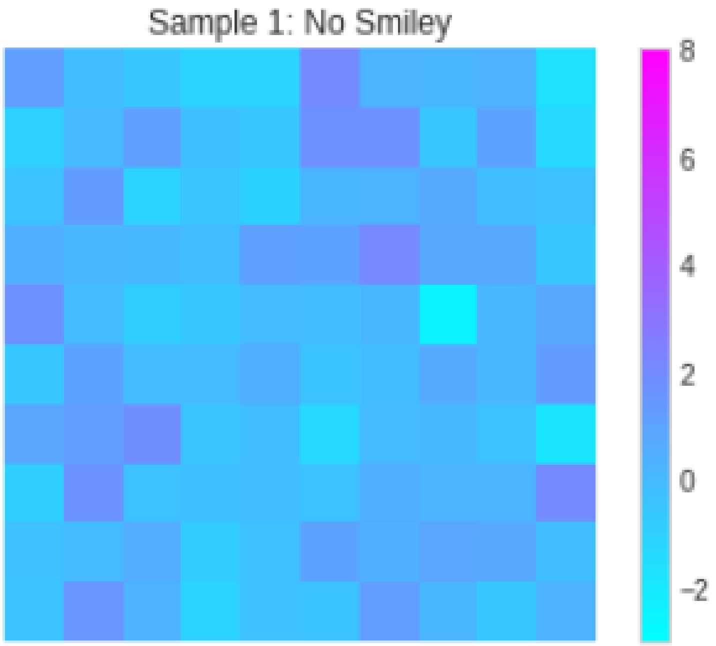
- 1D Convolutions
- 2D Convolutions
- Multiple Channels
- Choosing weights to highlight features
- Max Pooling

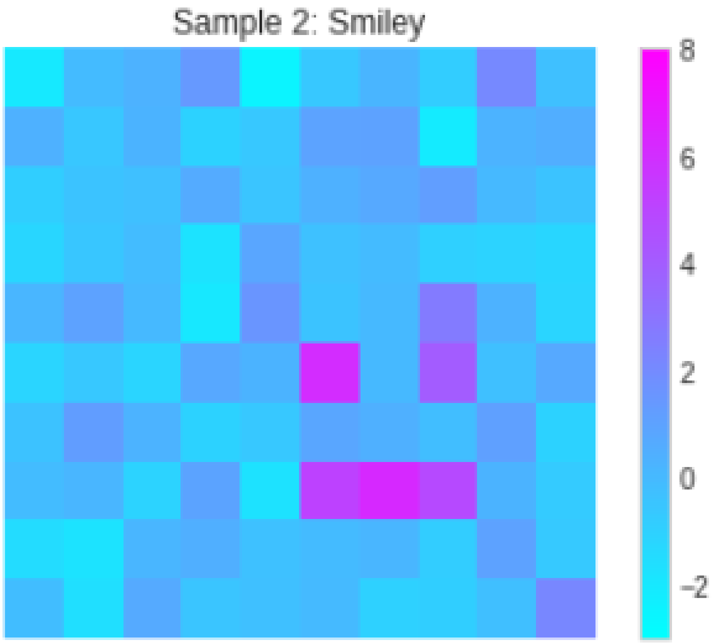
- Up next: Classification Example in Google Colab

Classification Example

- Network Architecture
  - Input Layer
  - Convolutional Layer
    - 1 kernel
  - Pooling Layer
  - Output Layer

- Sigmoid Activation
- Binary Cross Entropy





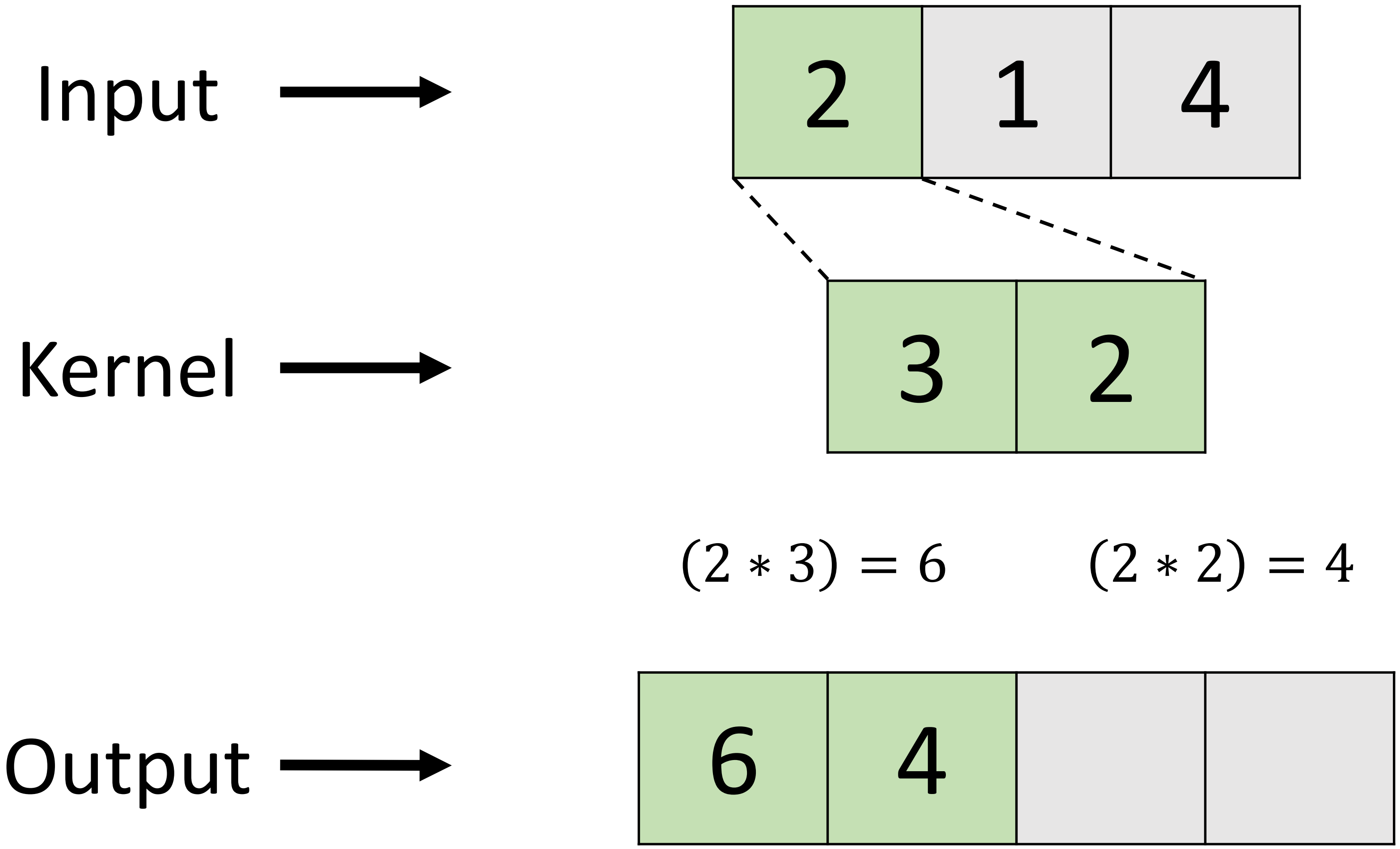
Outline

- 1D Convolution
  - Stride, Padding
- 2D Convolution
  - Single channel input
  - Multiple channel input (i.e. RGB)
  - Multiple channels on output
- Choosing Weights
  - Finding changes, edges, peaks, features, etc.
- Pooling Layers
- **Transposed Convolutions**
  - 1D
  - 2D
  - Stride, padding
  - Multiple channel input
  - Multiple channel output
- U-Net Architecture
  - Post-processing images

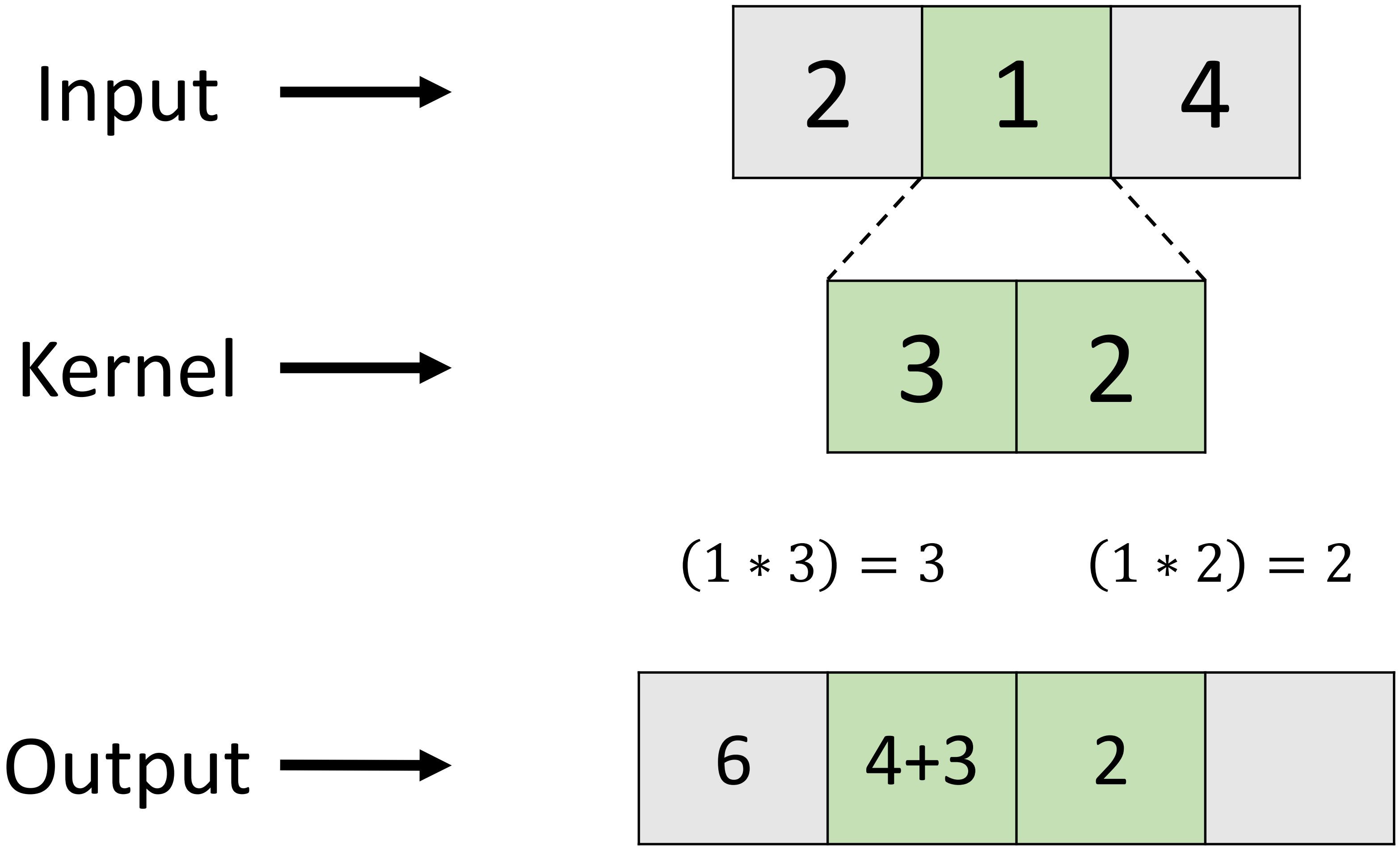
Transposed Convolutions

- Also called backwards convolution or deconvolution among other things (transposed makes the most sense in my opinion)
- A form of up sampling
- Increases the resolution (height and width) of the input
  - Kernel size, stride, and padding help manage output sizes
- Usually used while decreasing the number of channels

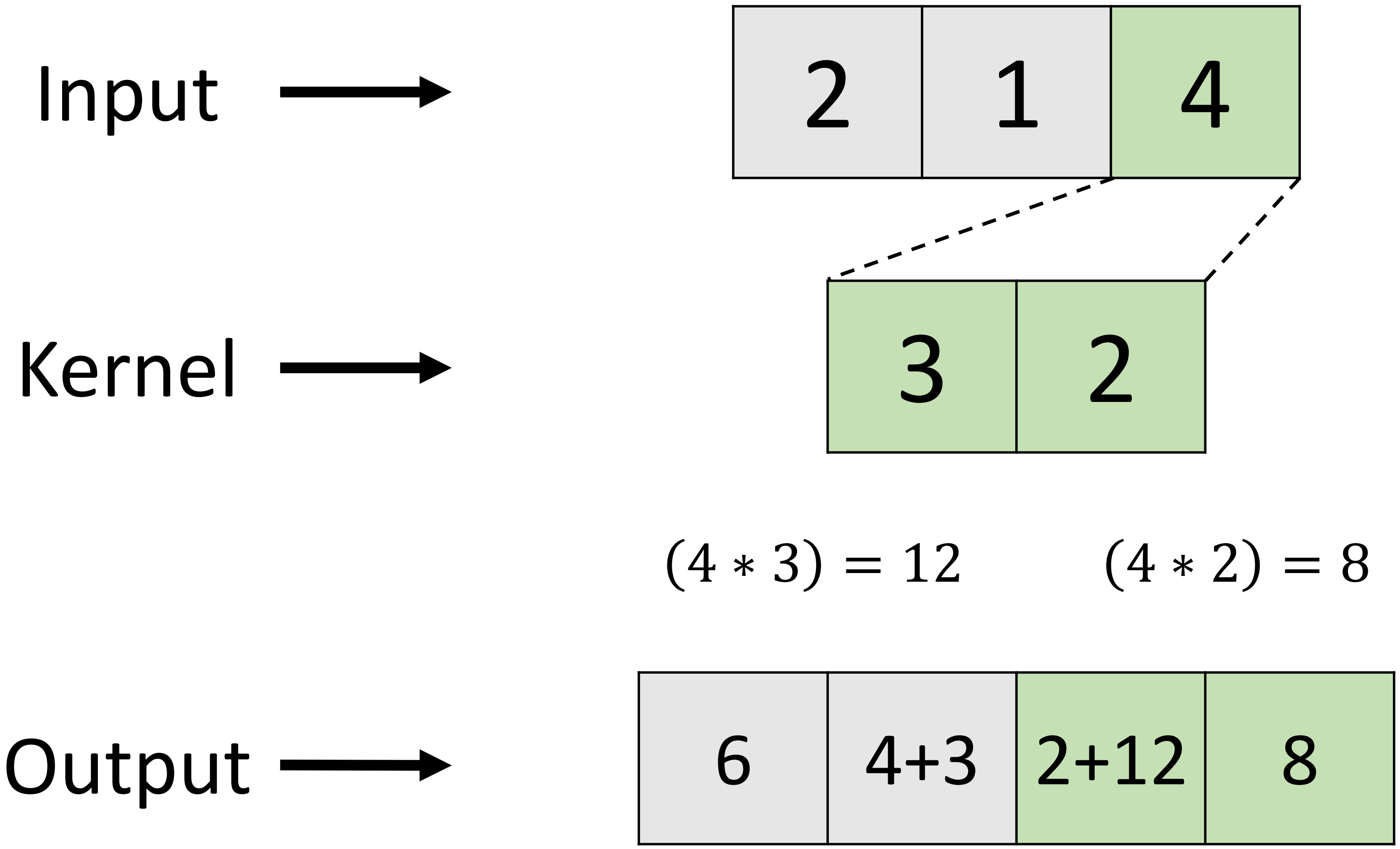
Transposed Convolutions (1D)



Transposed Convolutions (1D)



Transposed Convolutions (1D)



Transposed Convolutions (1D)

- This example used a stride of 1 and no padding

Input →	2	1	4	
Kernel →	3	2		
Output →	6	7	14	8

MARQUETTE UNIVERSITY | BE THE DIFFERENCE.

Transposed Convolutions (1D)

- We can pad with zeros NOT on the ends, but between entries

Input →

2	0	1	0	4

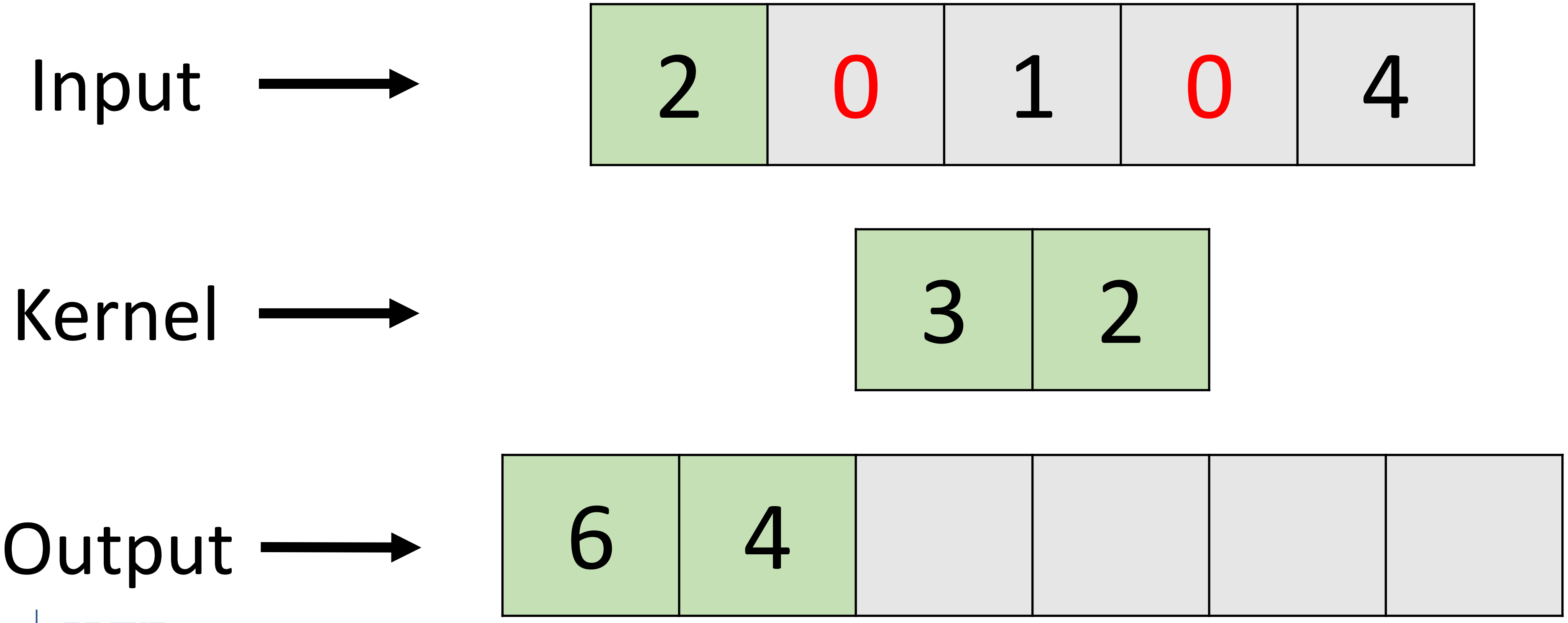
Kernel →

3	2

Output →

Transposed Convolutions (1D)

- We can pad with zeros NOT on the ends, but between entries



Transposed Convolutions (1D)

- We can pad with zeros NOT on the ends, but between entries

Input →

2	0	1	0	4

Kernel →

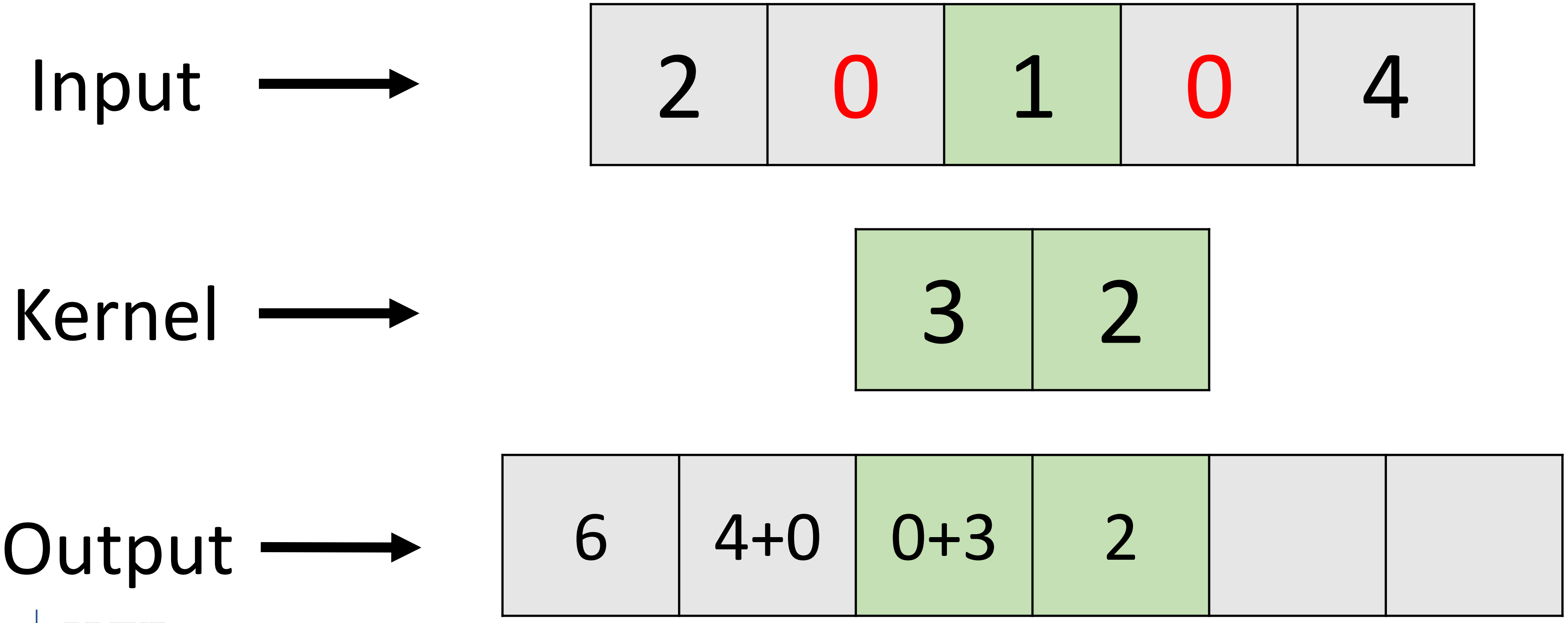
3	2

Output →

6	4+0	0			

Transposed Convolutions (1D)

- We can pad with zeros NOT on the ends, but between entries



Transposed Convolutions (1D)

- We can pad with zeros NOT on the ends, but between entries

Input →

2	0	1	0	4

Kernel →

3	2

Output →

6	4+0	0+3	2+0	0	

Transposed Convolutions (1D)

- We can pad with zeros NOT on the ends, but between entries

Input →

2	0	1	0	4

Kernel →

3	2

Output →

6	4+0	0+3	2+0	0+12	8

Transposed Convolutions (1D)

- Padding the input with 0's has the same effect as adding a stride to the output

Input →

2	0	1	0	4

Kernel →

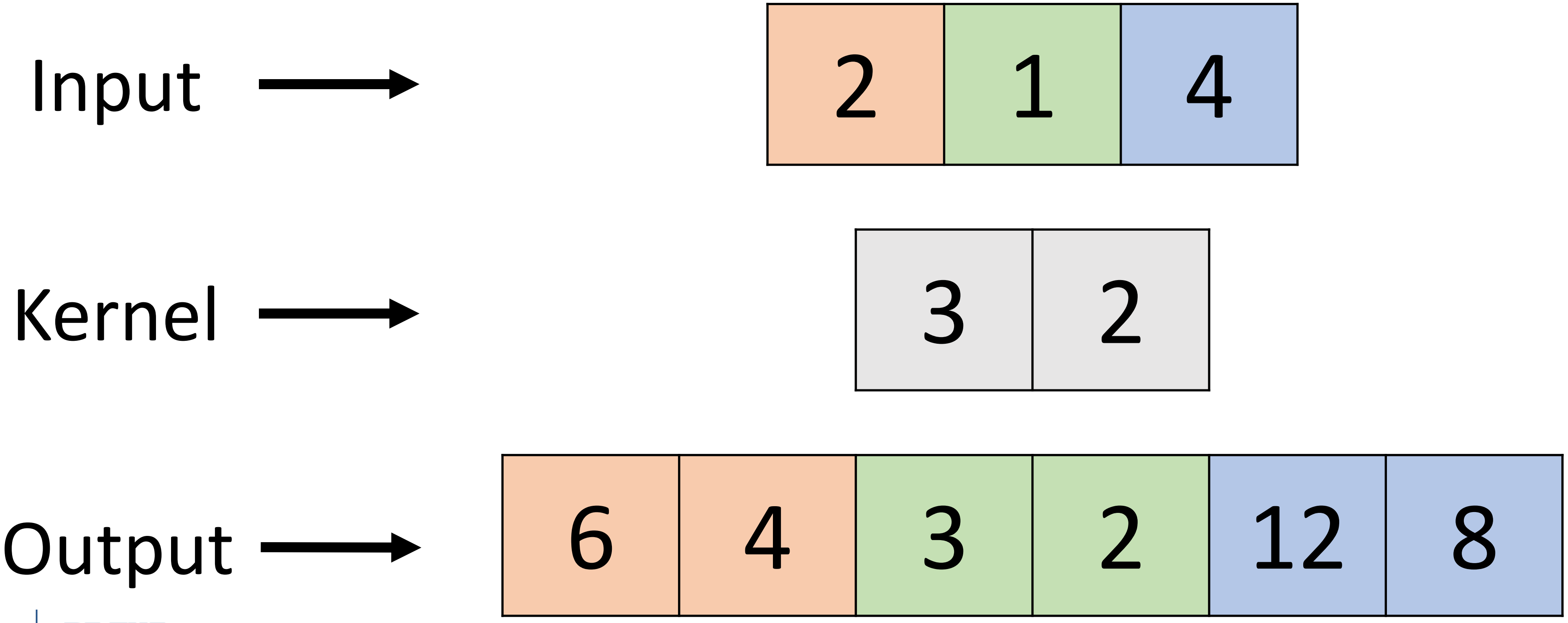
3	2

Output →

6	4	3	2	12	8

Transposed Convolutions (1D)

- Padding the input with 0’s has the same effect as adding a stride to the output (stride=2)



Questions?

- 1D Convolutions
- 2D Convolutions
- Multiple Channels
- Choosing weights to highlight features
- 1D Transposed Convolutions

- Up next: 2D Transposed Convolutions

Transposed Convolutions (2D)

- Again, 2D refers to the dimension of the stride parameter
- Padding between input entries with zeros or adjusting the stride of the output work with 2D transposed convolutions too

Transposed Convolutions (2D)

- Example with no padding (output stride = (1,1))

Input

1	2
3	4

Kernel

2	1
1	0

Output

2	1	
1	0	

Transposed Convolutions (2D)

- Example with no padding (output stride = (1,1))

Input

1	2
3	4

Kernel

2	1
1	0

Output

2	1+2	2
1	0+2	0

Transposed Convolutions (2D)

- Example with no padding (output stride = (1,1))

Input

1	2
3	4

Kernel

2	1
1	0

Output

2	1+2	2
1+6	0+2+3	0
3	0	

Transposed Convolutions (2D)

- Example with no padding (output stride = (1,1))

Input

1	2
3	4

Kernel

2	1
1	0

Output

2	1+2	2
1+6	0+2+ 3+8	0+4
3	0+4	0

Transposed Convolutions (2D)

- Example with no padding (output stride = (1,1))

Input

1	2
3	4

Kernel

2	1
1	0

Output

2	3	2
7	13	4
3	4	0

Transposed Convolutions (2D)

- The stride of (1,1) resulted in overlap in the orange output entries

Input

1	2
3	4

Kernel

2	1
1	0

Output

2	3	2
7	13	4
3	4	0

Transposed Convolutions (2D)

These mean the same thing. It's not two separate things being done; it's two ways of thinking of the same thing

- Example with padding (output stride = (2,2))

Input

1	0	2
0	0	0
3	0	4

Kernel

2	1
1	0

Output

2	1	4	2
1	0	2	0
6	3	8	4
3	0	4	0

MARQUETTE UNIVERSITY | BE THE DIFFERENCE.

Transposed Convolutions (2D)

Input

3	1
-1	2

1	1
2	1

Kernel

1	-1
2	1

1	0
-1	-1

Output

3	-3		
6	3		

Input

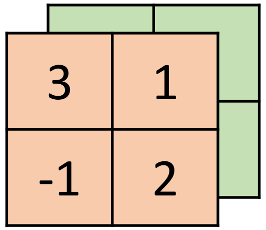
3	1
-1	2

Kernel

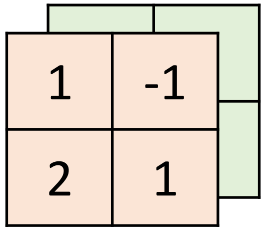
1	-1
2	1

Transposed Convolutions (2D)

Input



Kernel



Input $\begin{bmatrix} 3 & 1 \\ -1 & 2 \end{bmatrix}$ $\begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$

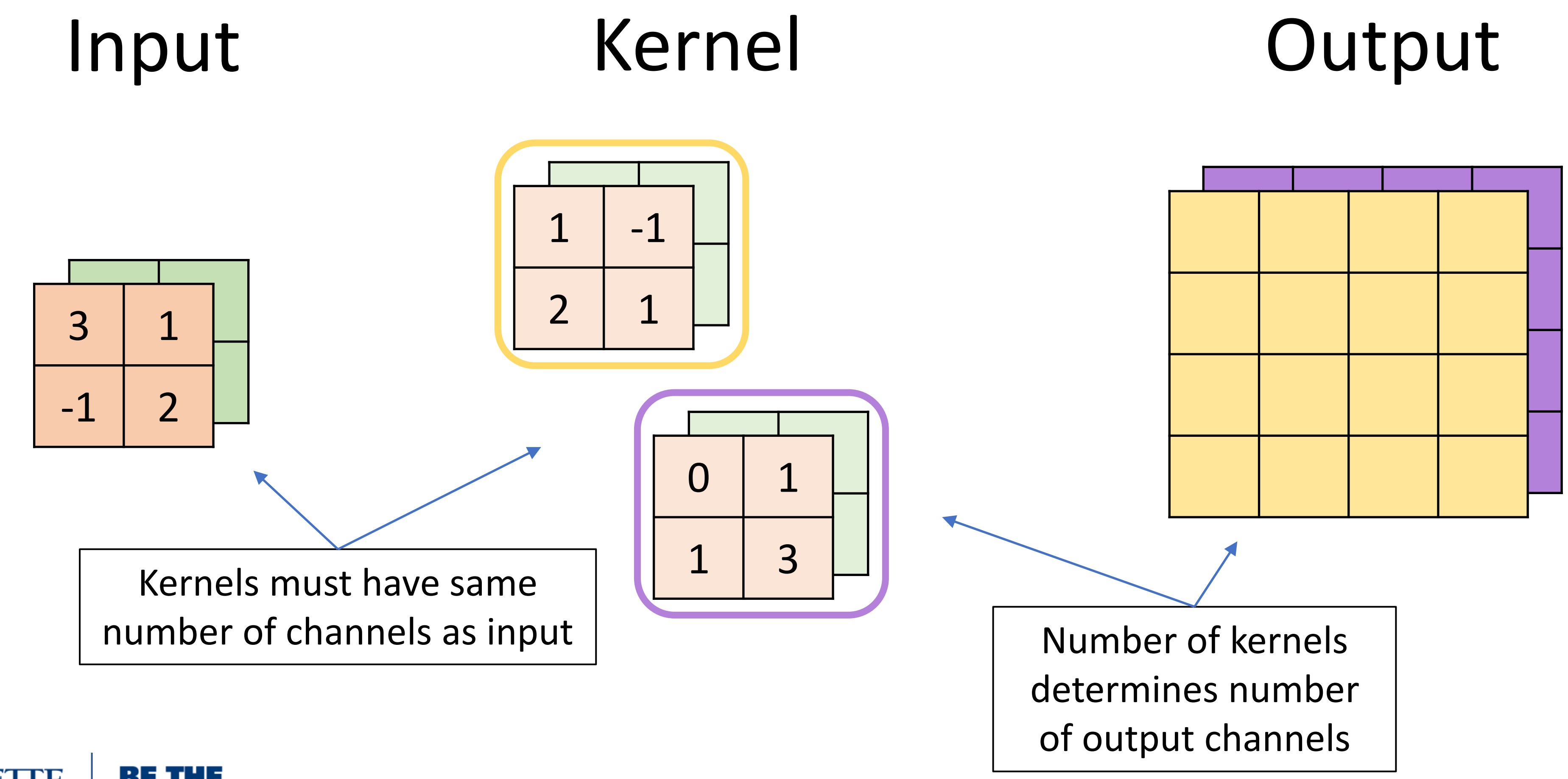
Kernel $\begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 & 0 \\ -1 & -1 \end{bmatrix}$

Output

$$\begin{bmatrix} 3 & -3 & 1 & -1 \\ 6 & 3 & 2 & 1 \\ -1 & 1 & 2 & -2 \\ -2 & -1 & 4 & 2 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 & 0 \\ -1 & -1 & -1 & -1 \\ 2 & 0 & 1 & 0 \\ -2 & -2 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 4 & -3 & 2 & -1 \\ 5 & 2 & 1 & 0 \\ 1 & 1 & 3 & -2 \\ -4 & -3 & 3 & 1 \end{bmatrix}$$

Output

Transposed Convolutions (2D)



Transposed Convolution Layer

- For each transposed convolutional layer, how many parameters need to be learned by the algorithm?

$$\underbrace{(num\ elements\ in\ kernel) * (num\ kernels)}_{\text{weights}} + \underbrace{(num\ kernels)}_{\text{biases}}$$

Questions?

- 1D Convolutions
- 2D Convolutions
- Multiple Channels
- Choosing weights to highlight features
- 1D Transposed Convolutions
- 2D Transposed Convolutions

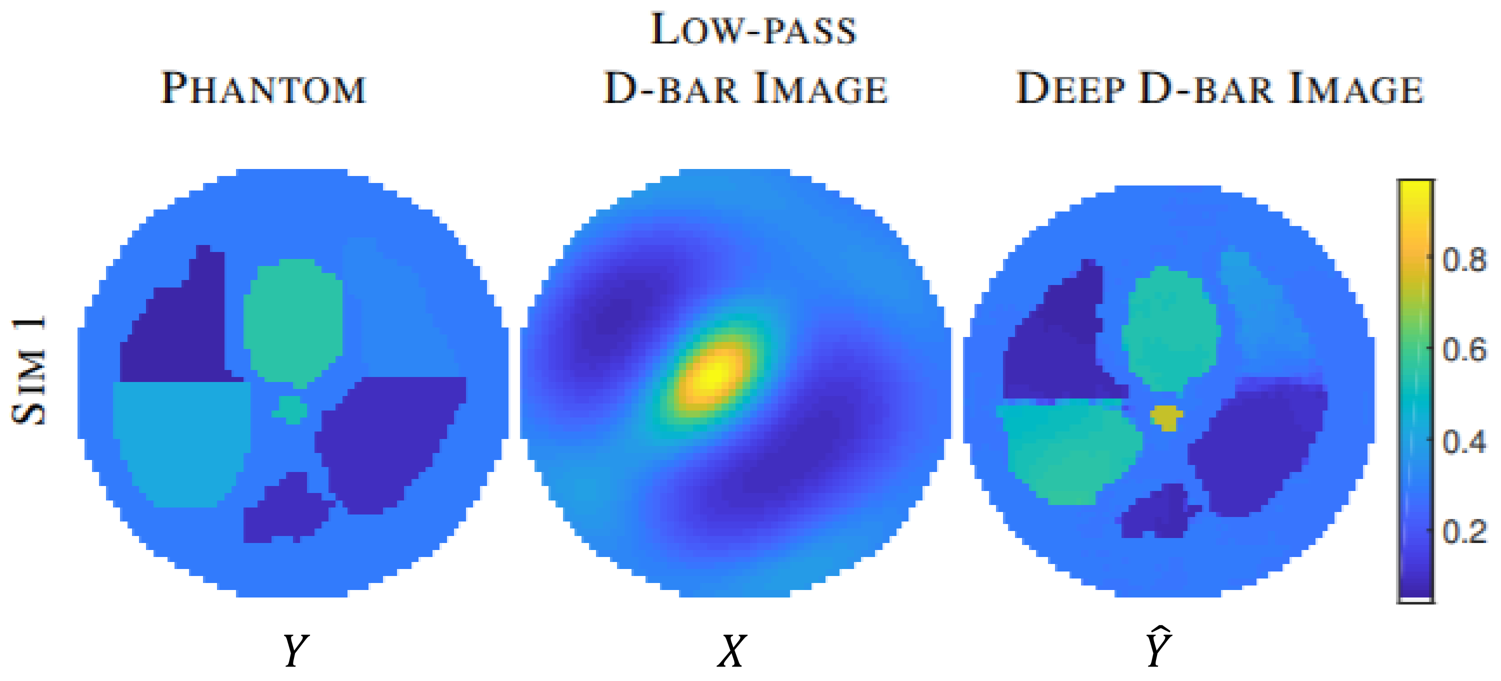
- Up next: U-Net Architecture for post processing images

Outline

- 1D Convolution
  - Stride, Padding
- 2D Convolution
  - Single channel input
  - Multiple channel input (i.e. RGB)
  - Multiple channels on output
- Choosing Weights
  - Finding changes, edges, peaks, features, etc.
- Pooling Layers
- Transposed Convolutions
  - 1D
  - 2D
  - Stride, padding
  - Multiple channel input
  - Multiple channel output
- **U-Net Architecture**
  - Post-processing images

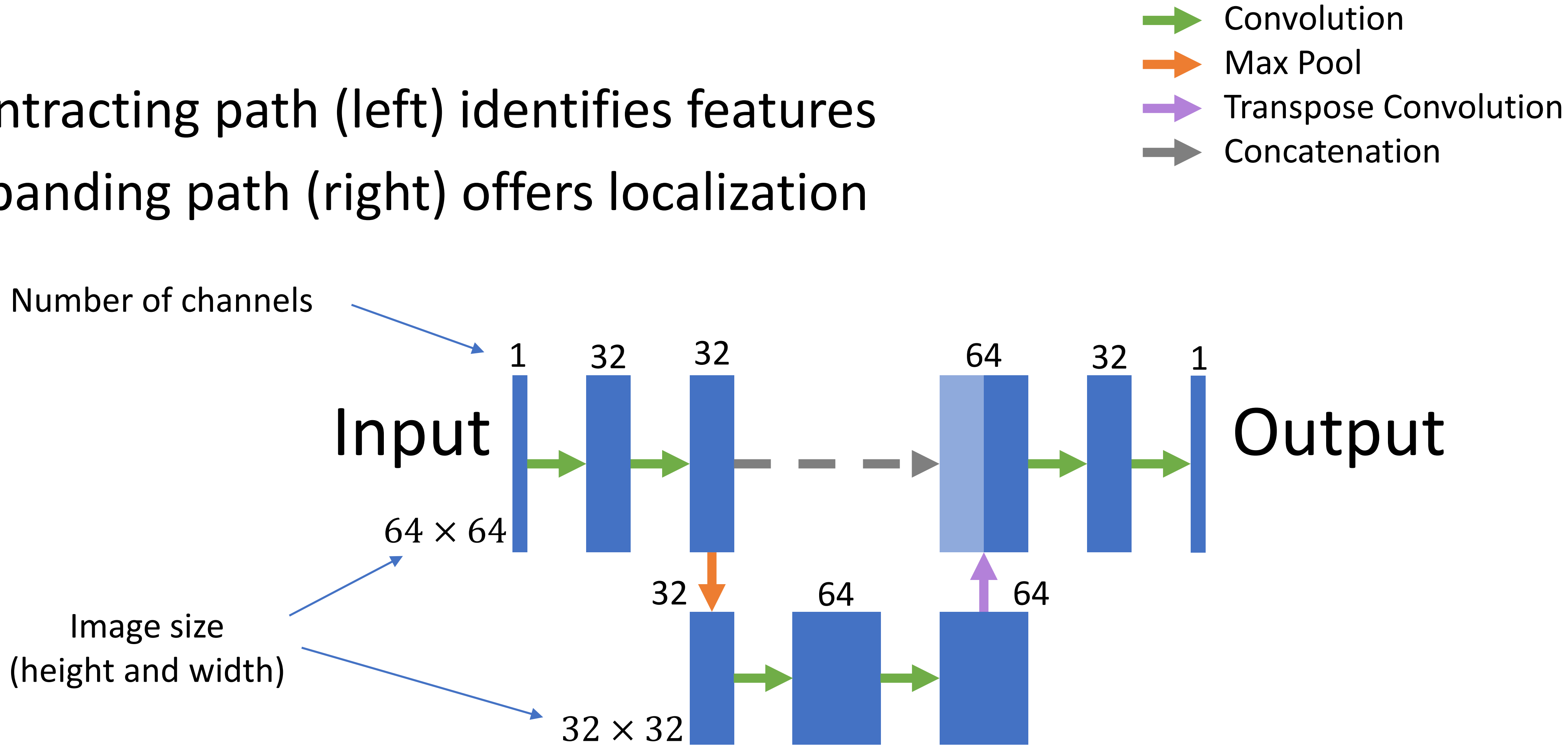
U-Net Architecture

- Presented by Ronneberger et al. in 2015 for image segmentation
- Shown to sharpen EIT D-bar reconstructions by Hamilton and Hauptmann in 2018



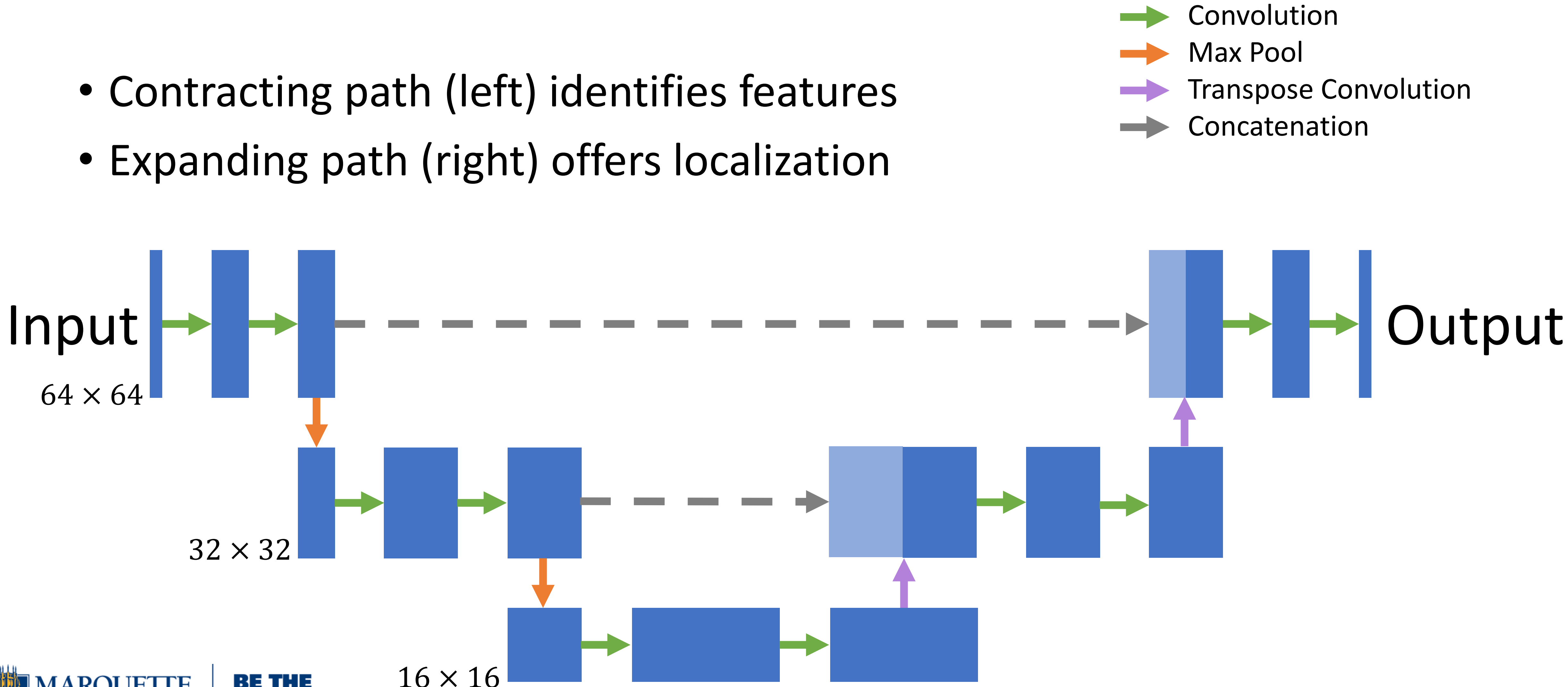
U-Net Architecture

- Contracting path (left) identifies features
- Expanding path (right) offers localization



U-Net Architecture

- Contracting path (left) identifies features
- Expanding path (right) offers localization



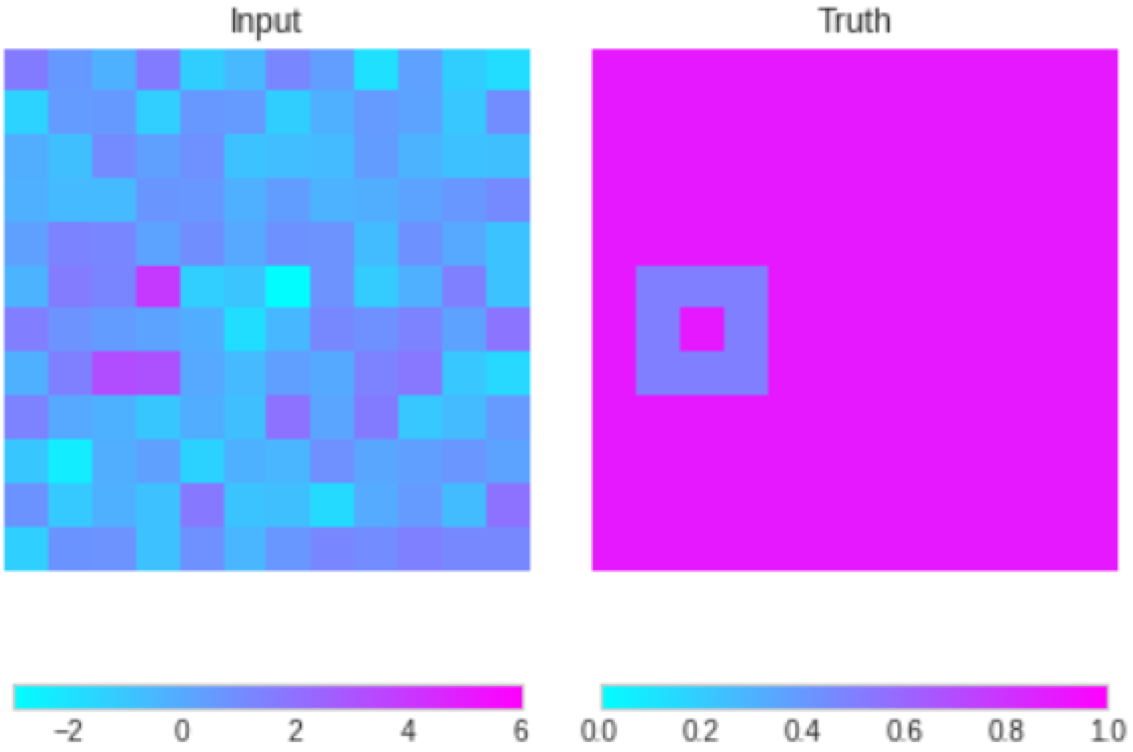
Questions?

- 1D Convolutions
- 2D Convolutions
- Multiple Channels
- Choosing weights to highlight features
- 1D Transposed Convolutions
- 2D Transposed Convolutions
- U-Net Architecture
- Up next: U-Net Example in Google Colab

U-Net Example

- Input is a blurry image with (or without) a smiley face
- Truth is an image with a ring (or without) and no noise

- Network
  - U-Net
  - 1 Max Pool Layer
  - MSE loss function
  - ReLU activation



Why use CNNs? (as opposed to dense networks)

- Number of weights is not related to the image sizes
  - Could have used 500x500 smiley face images
- Feature learning
  - By the end of training, the classifier knew what features to look for
- Spatially invariant
  - Can detect a smiley face in any location

Thank You!

Questions?

MARQUETTE UNIVERSITY | BE THE DIFFERENCE.