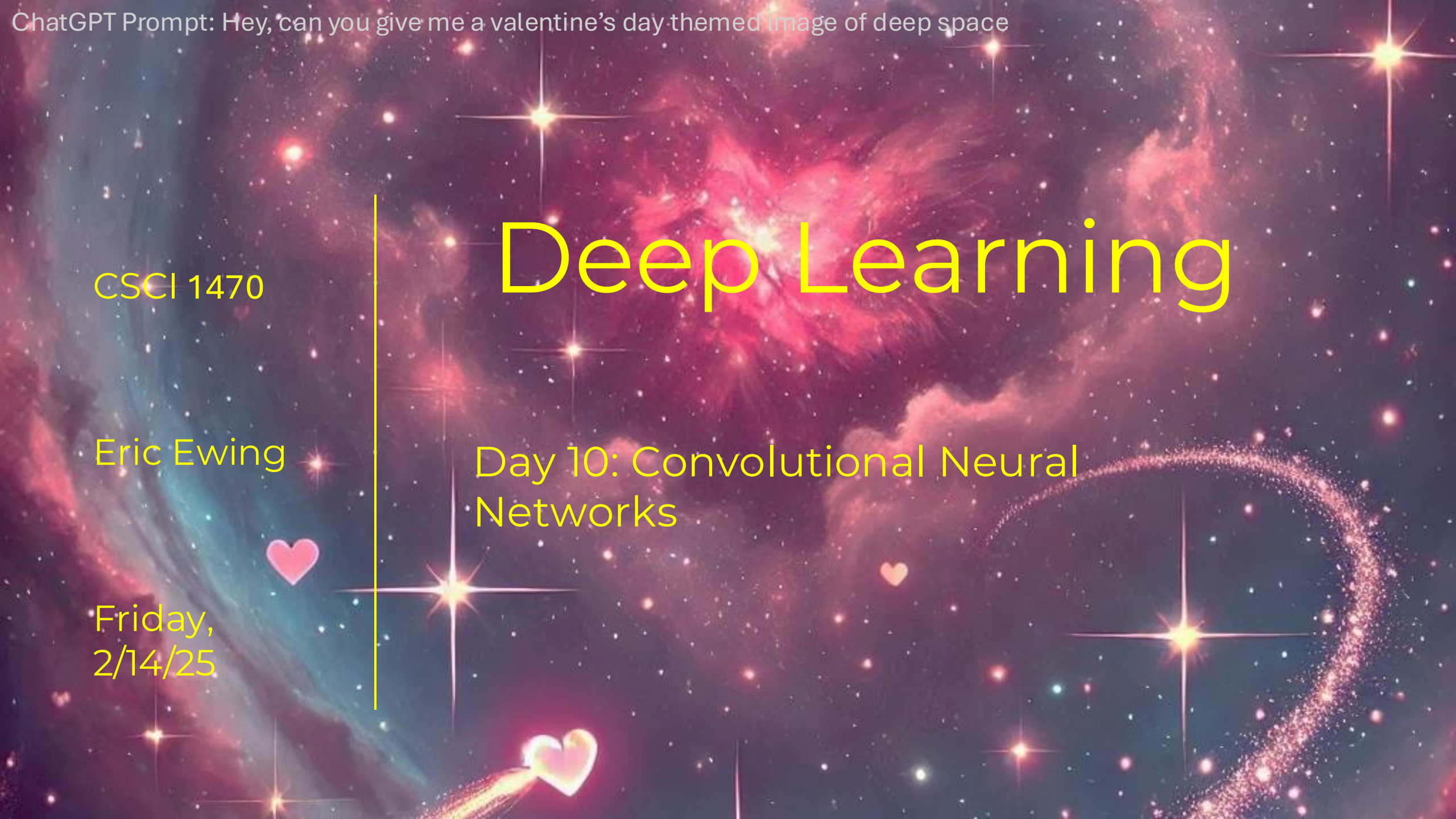


Deep Learning

CSCI 1470

Eric Ewing

Friday, 2/14/25

Day 10: Convolutional Neural Networks

Today’s Goals: Continue learning about CNNs

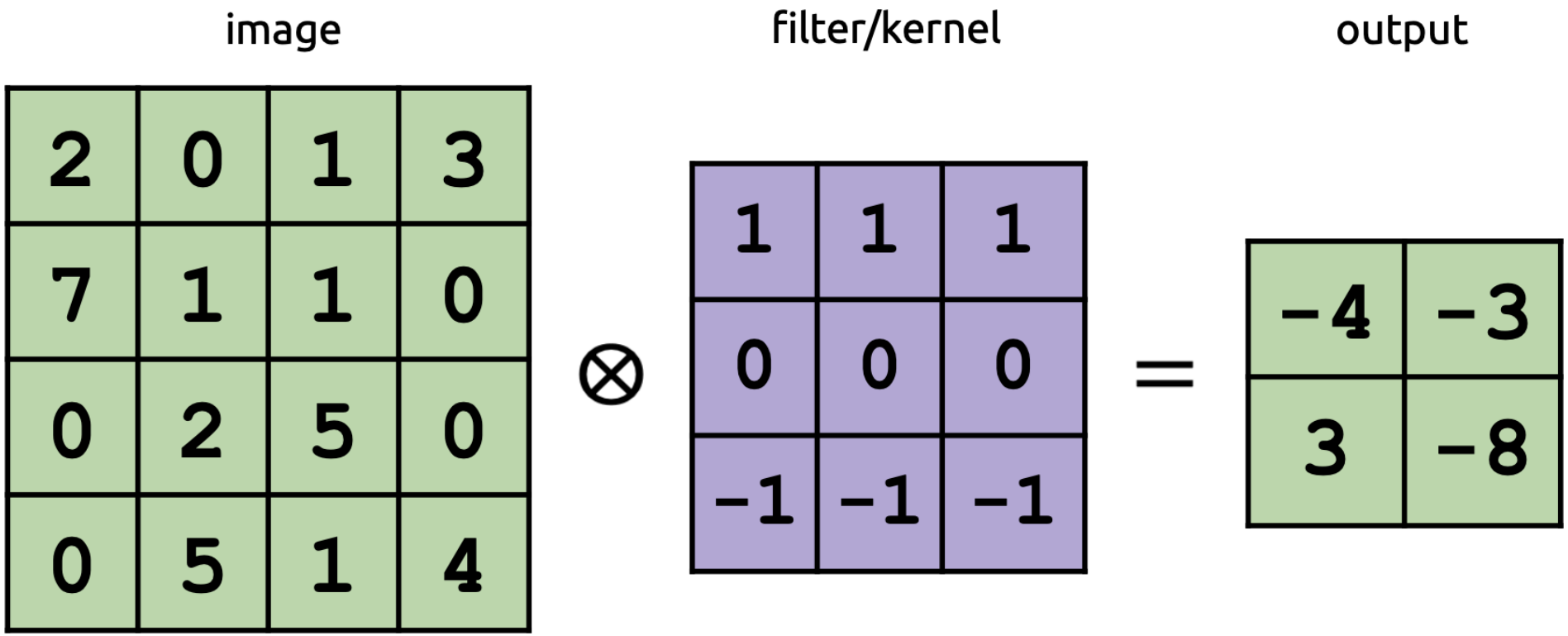
(1) Review of Convolutions

(2) Learning Filters

(3) What new hyperparameters do we have and what effects do they have? (stride, padding, size, etc.)

What Convolution Does (Visually)

In summary:



What Convolution Does (Mathematically)

$$V(x,y) = (\mathrm{I} \otimes K)(x,y) = \sum_m \sum_n I(x+m, y+n) K(m,n)$$

The output at pixel (x, y)

"Image I convolved with kernel K"

Sum over kernel columns

Sum over kernel rows

Multiply kernel value with corresponding image pixel value

What Convolution Does (Mathematically)

$$V(x,y) = (\mathrm{I} \otimes K)(x,y) = \sum_m \sum_n I(x+m, y+n) K(m,n)$$

The output at pixel (x, y)

"Image I convolved with kernel K"

Sum over kernel columns

Sum over kernel rows

Multiply kernel value with corresponding image pixel value

Is this a matrix multiplication?

What Convolution Does (Mathematically)

$$V(x,y) = (\mathrm{I} \otimes K)(x,y) = \sum_m \sum_n I(x+m, y+n) K(m,n)$$

The output at pixel (x, y)

"Image I convolved with kernel K"

Sum over kernel columns

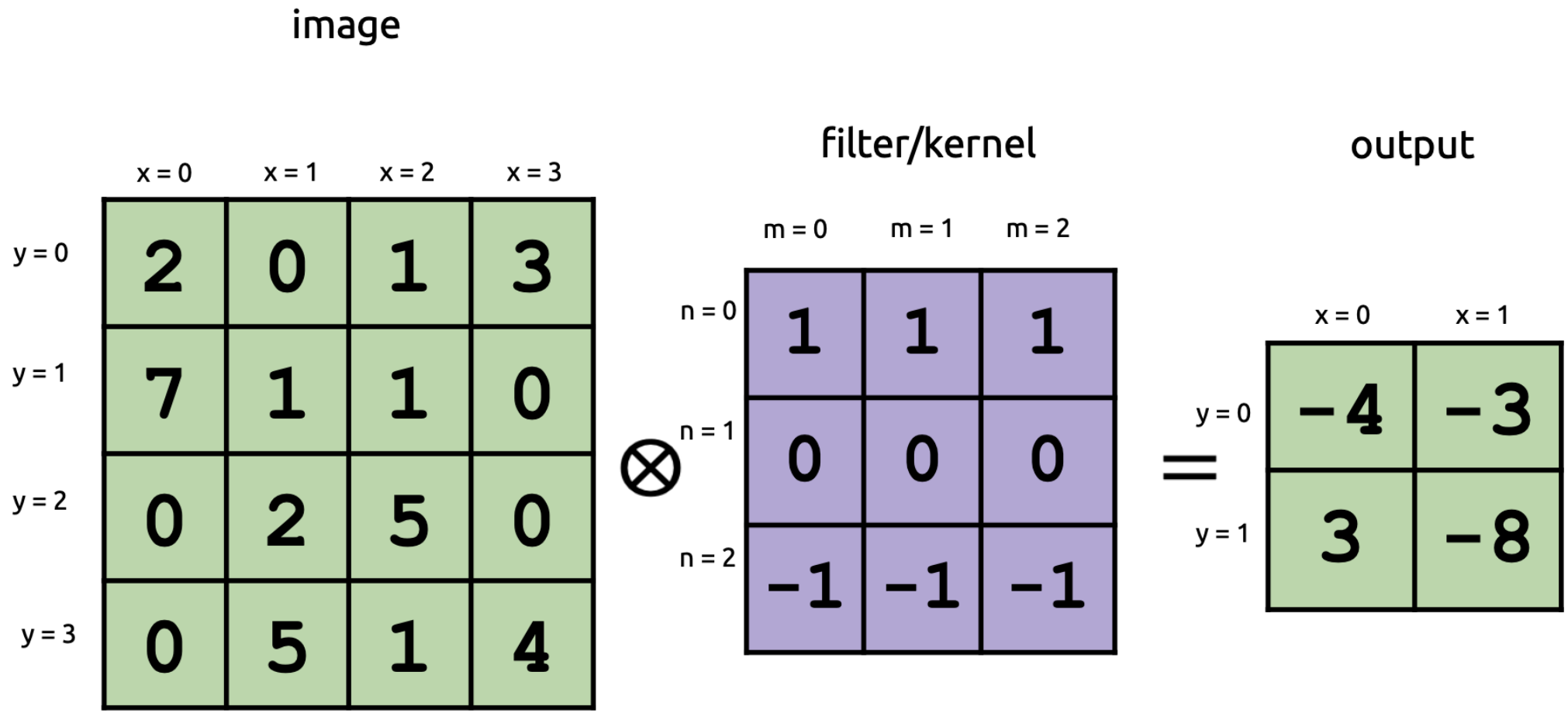
Sum over kernel rows

Multiply kernel value with corresponding image pixel value

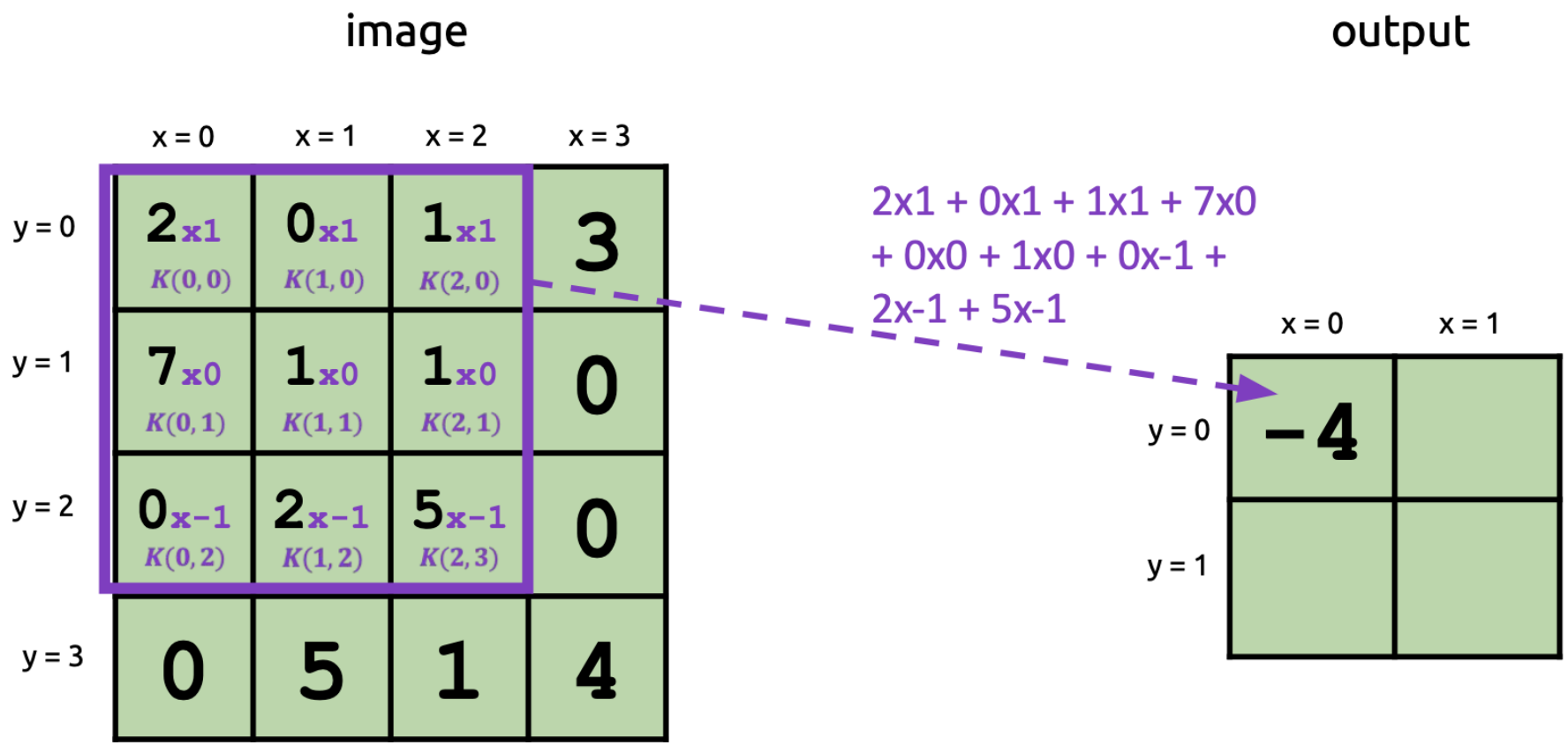
Is this a matrix multiplication?

No, output for each pixel is a single value, not a matrix.

What Convolution Does (Mathematically)

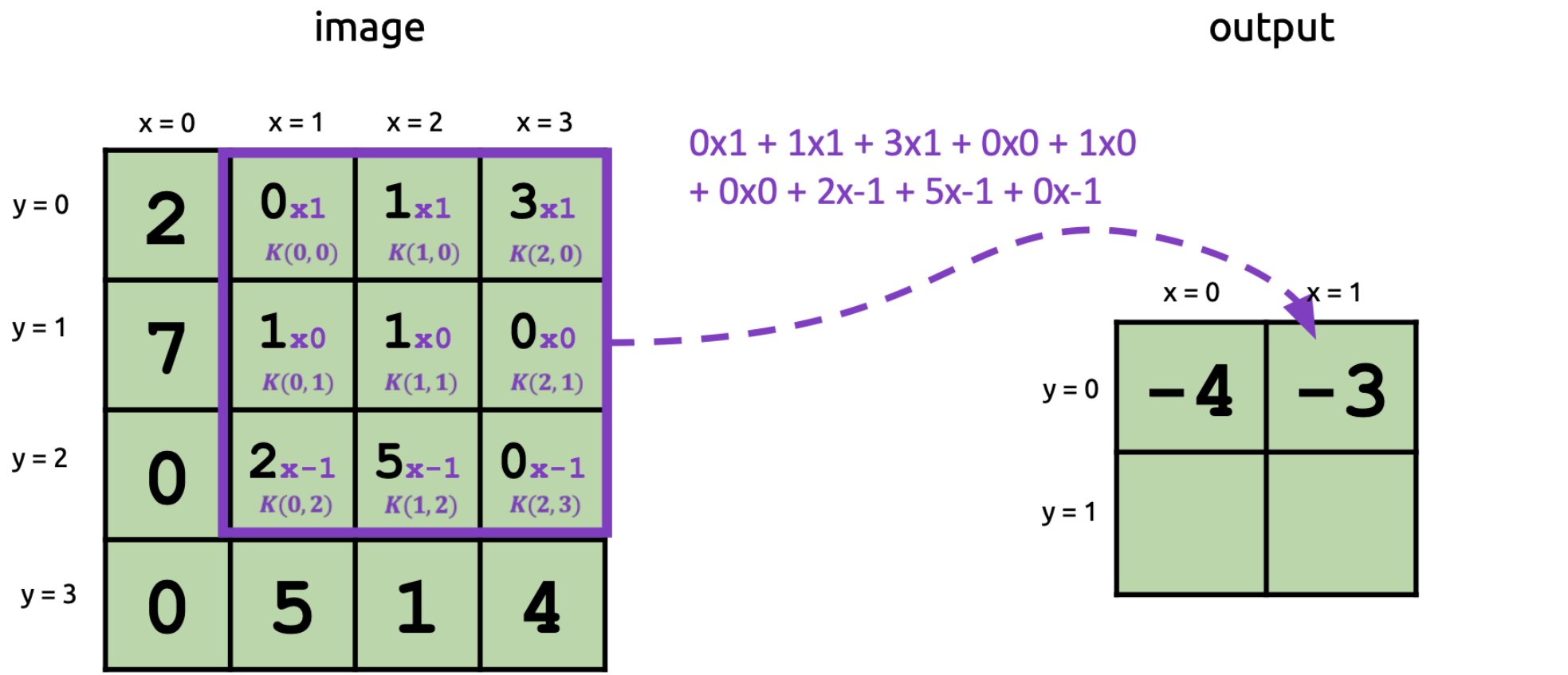


What Convolution Does (Mathematically)



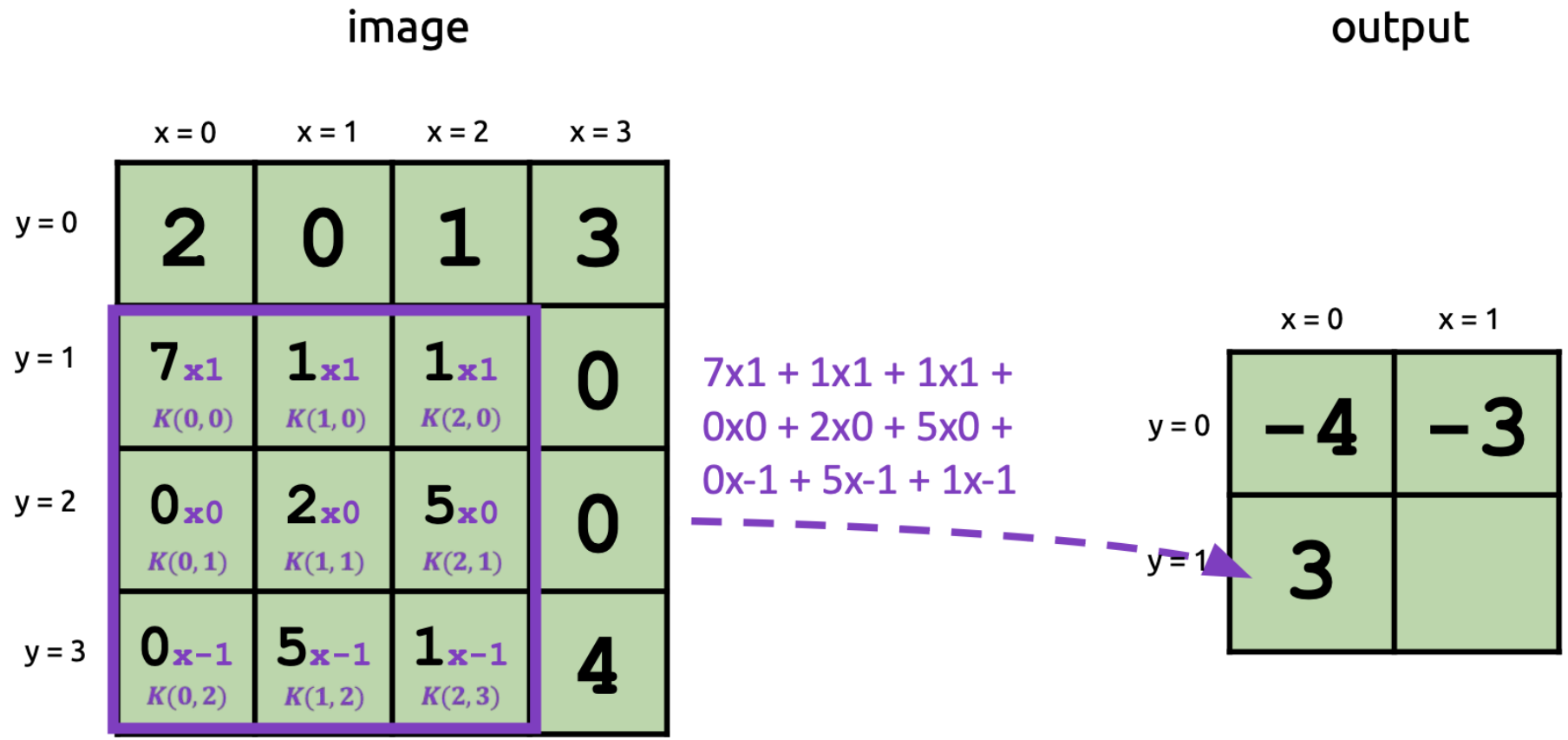
$$V(0,0) = (\mathrm{I} \otimes K)(0,0) = \sum_{m=0}^{2} \sum_{n=0}^{2} I(0+m, 0+n) K(m,n)$$

What Convolution Does (Mathematically)



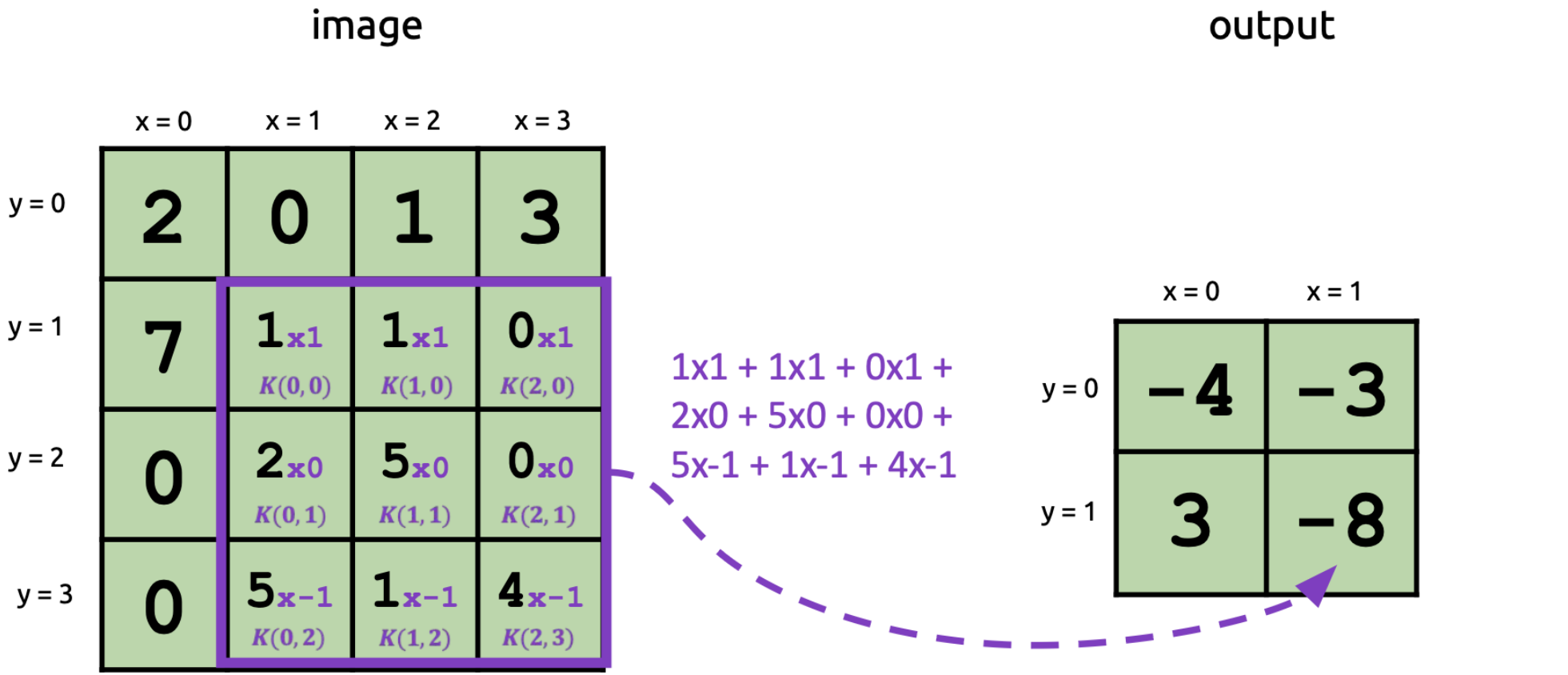
$$V(1,0) = (\mathrm{I} \otimes K)(1,0) = \sum_{m=0}^{2} \sum_{n=0}^{2} I(1+m, 0+n) K(m,n)$$

What Convolution Does (Mathematically)



$$V(0,1) = (\mathrm{I} \otimes K)(0,1) = \sum_{m=0}^{2} \sum_{n=0}^{2} I(0+m, 1+n) K(m,n)$$

What Convolution Does (Mathematically)



$$V(1,1) = (\mathrm{I} \otimes K)(1,1) = \sum_{m=0}^{2} \sum_{n=0}^{2} I(1+m, 1+n) K(m,n)$$

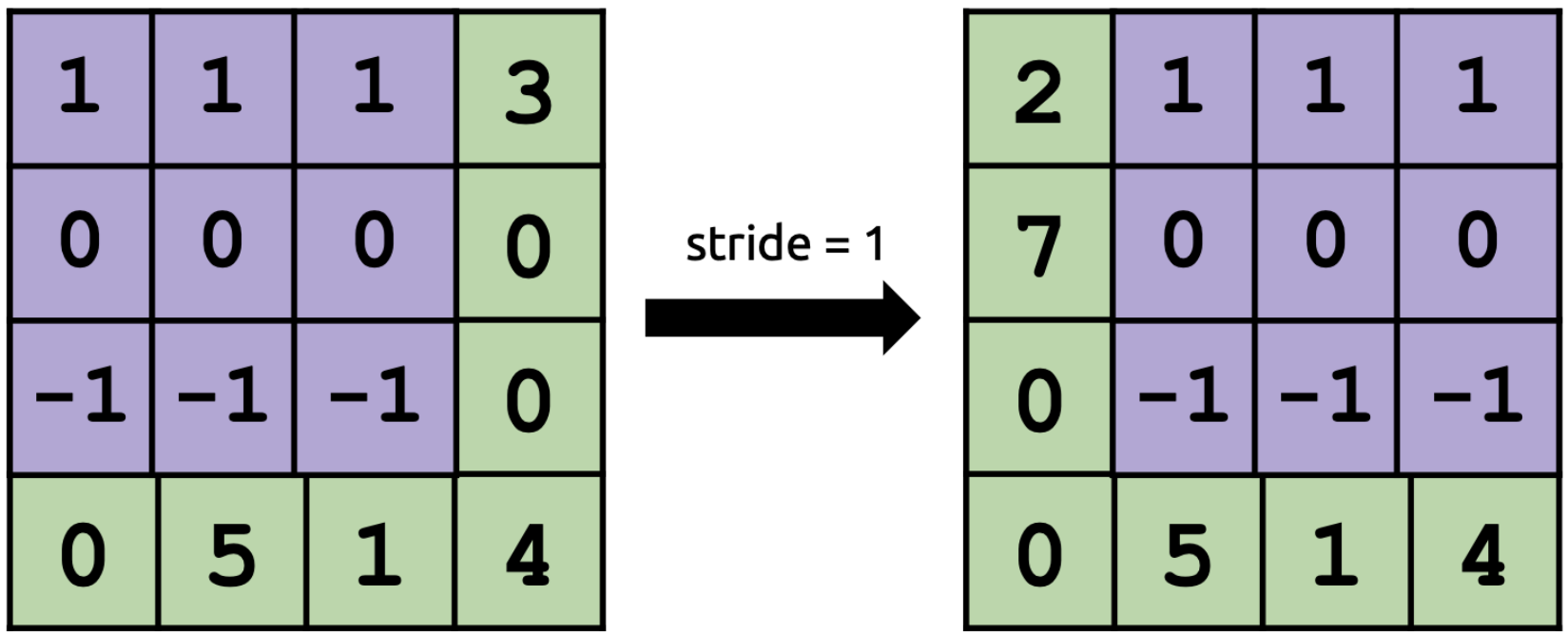
What Convolution Does (In Code)

```
// Input: Image I, Kernel K, Output V, pixel index x,y
// Assumes K is 3x3
function apply_kernel(I, K, V, x, y)
    for m = 0 to 2:
        for n = 0 to 2:
            V(x,y) += K(m,n) * I(m+x, n+y)
```

$$Equation\text{: } V(x,y) = (\mathrm{I} \otimes K)(x,y) = \sum_m \sum_n I(x+m, y+n) K(m,n)$$

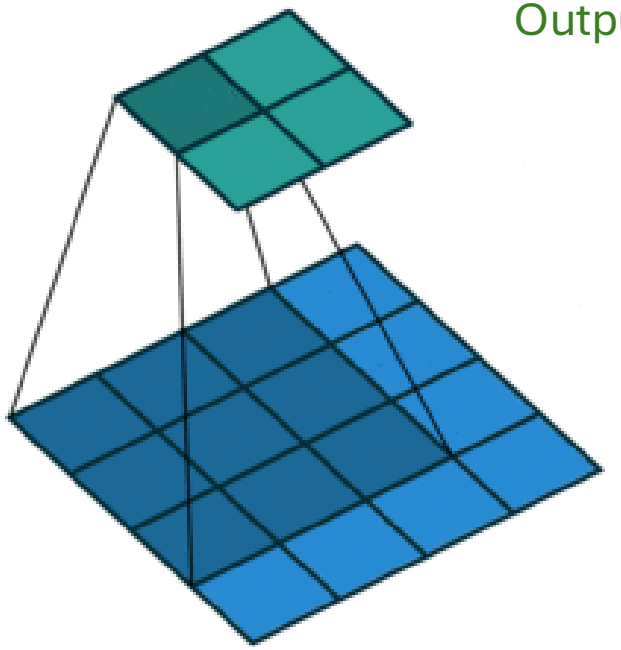
Stride

- We don’t just have to slide the filter by one pixel every time
- The distance we slide a filter by is called ***stride***
  - All the examples we’ve seen thus far have been stride = 1



Stride

Stride=1



Output

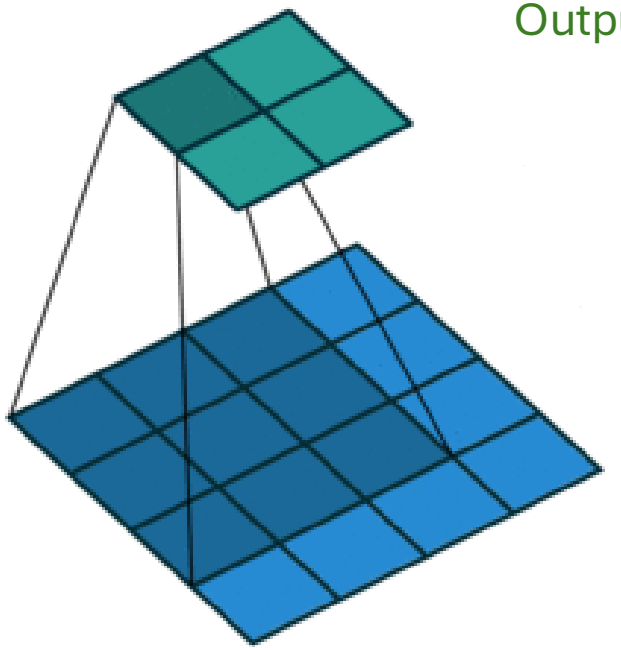
Input

Accessed: https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html

Stride

Stride=1

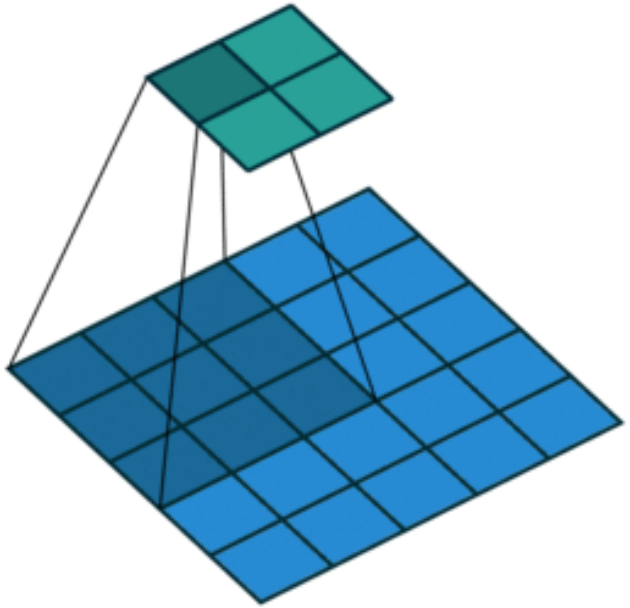
Output



Input

Stride=2

Output

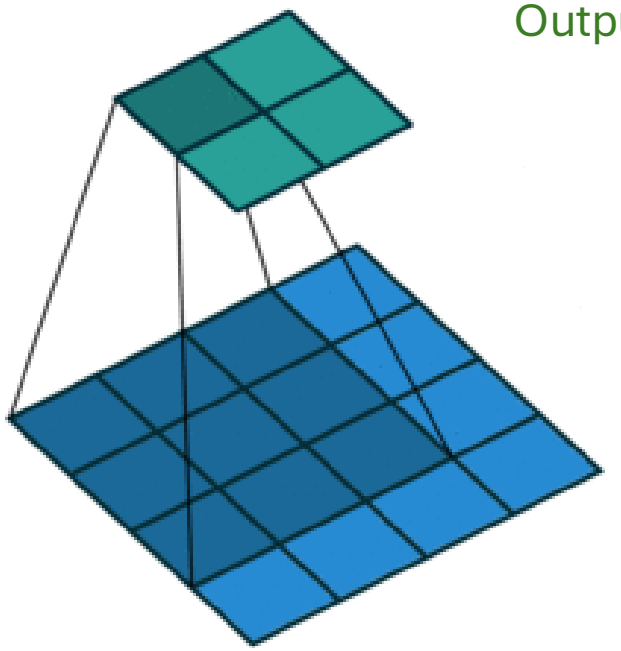


Input

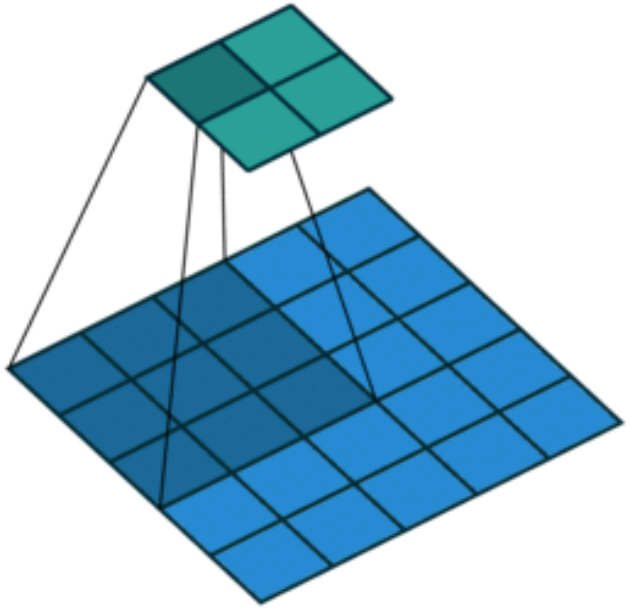
Accessed: https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html

Stride

Stride=1



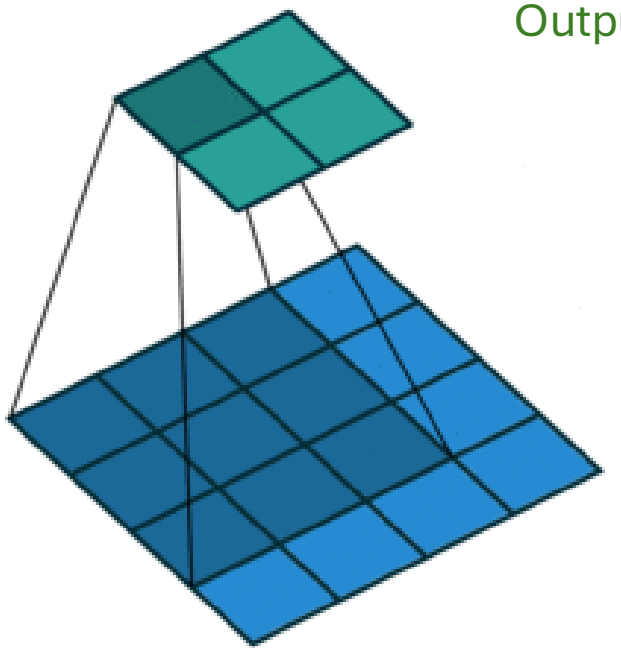
Stride=2



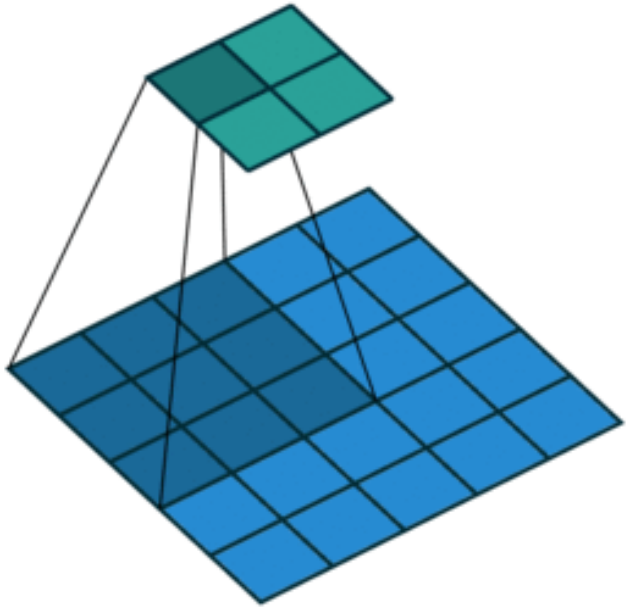
Any connection between input and output size?

Stride

Stride=1



Stride=2

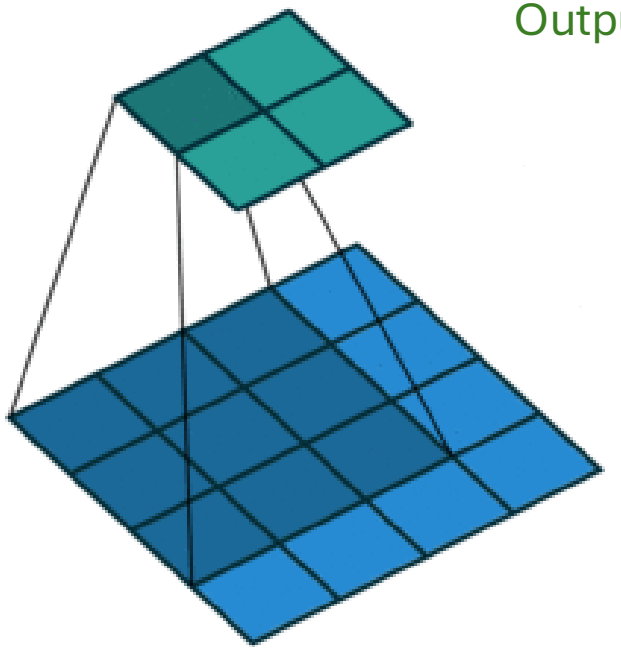


Larger stride turns **larger** input into **same** sized output

Stride

Stride=1

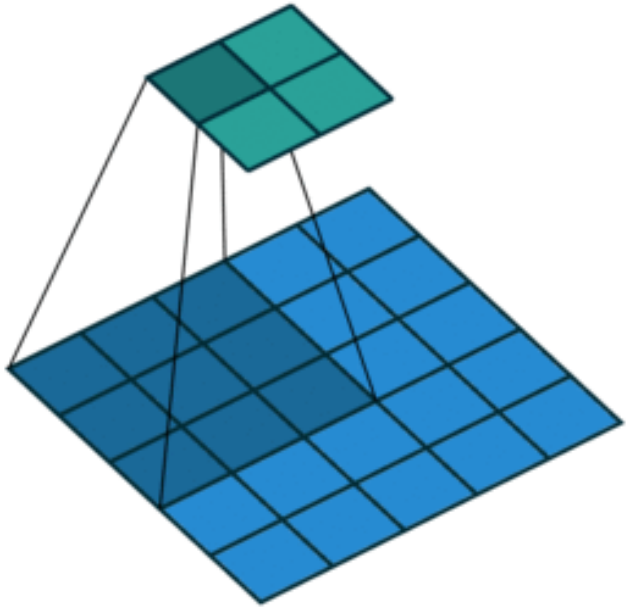
Output



Input

Stride=2

Output

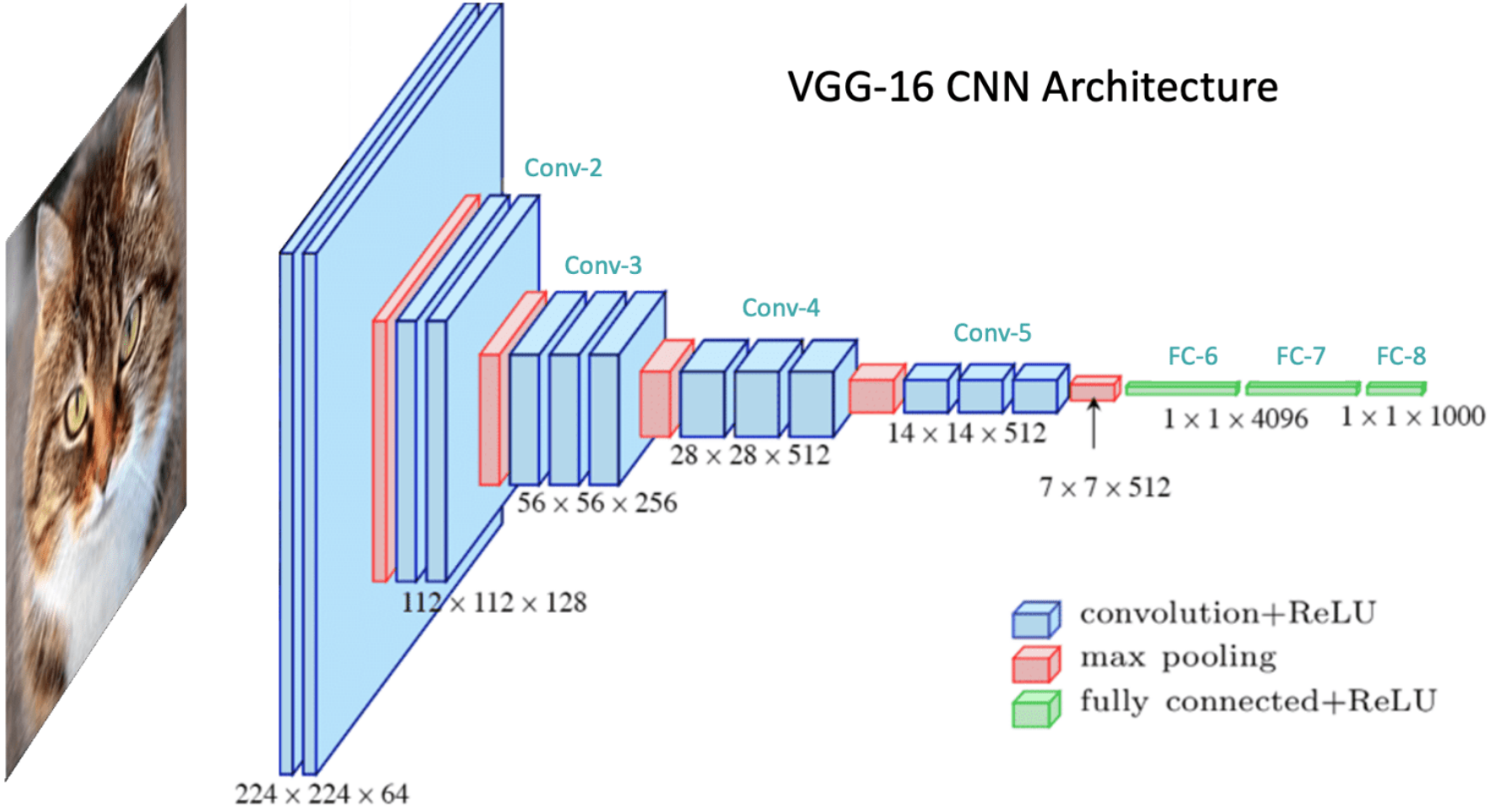


Input

Larger stride turns **same** sized input into **smaller** sized output

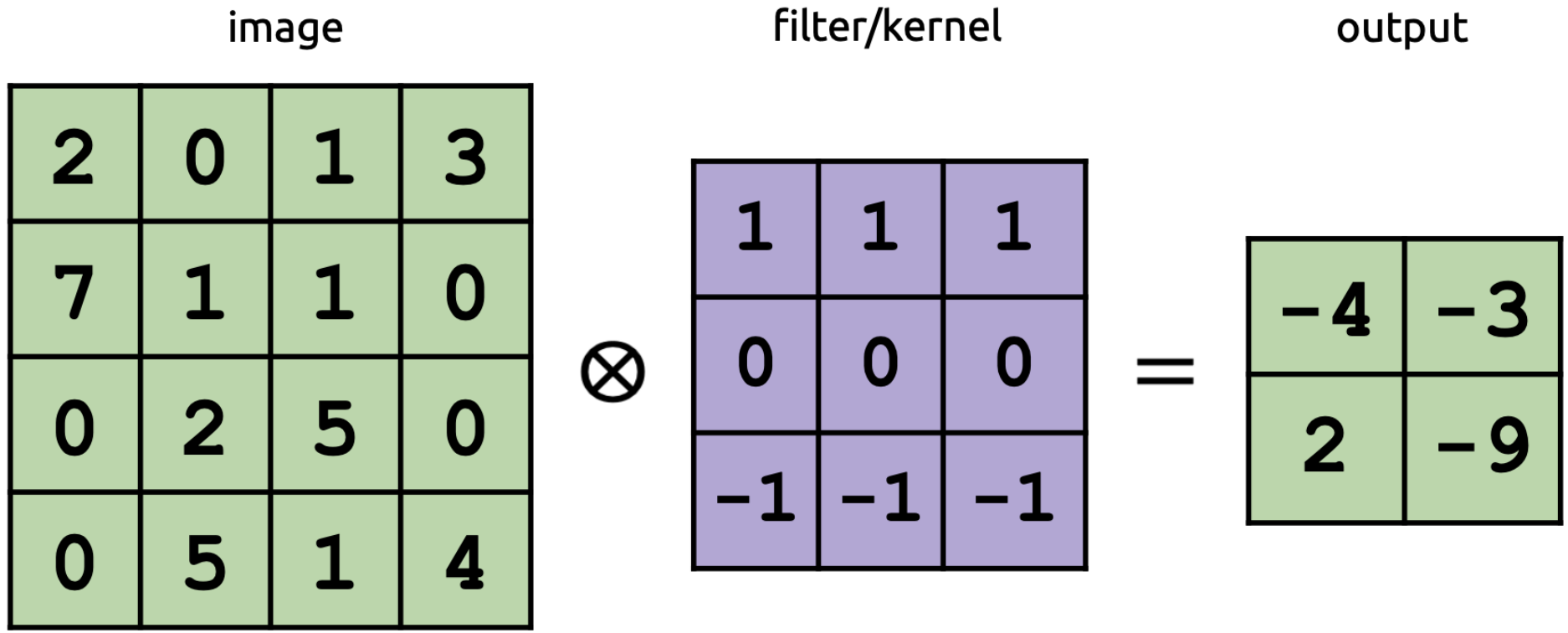
Accessed: https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html

Convolutional Neural Networks

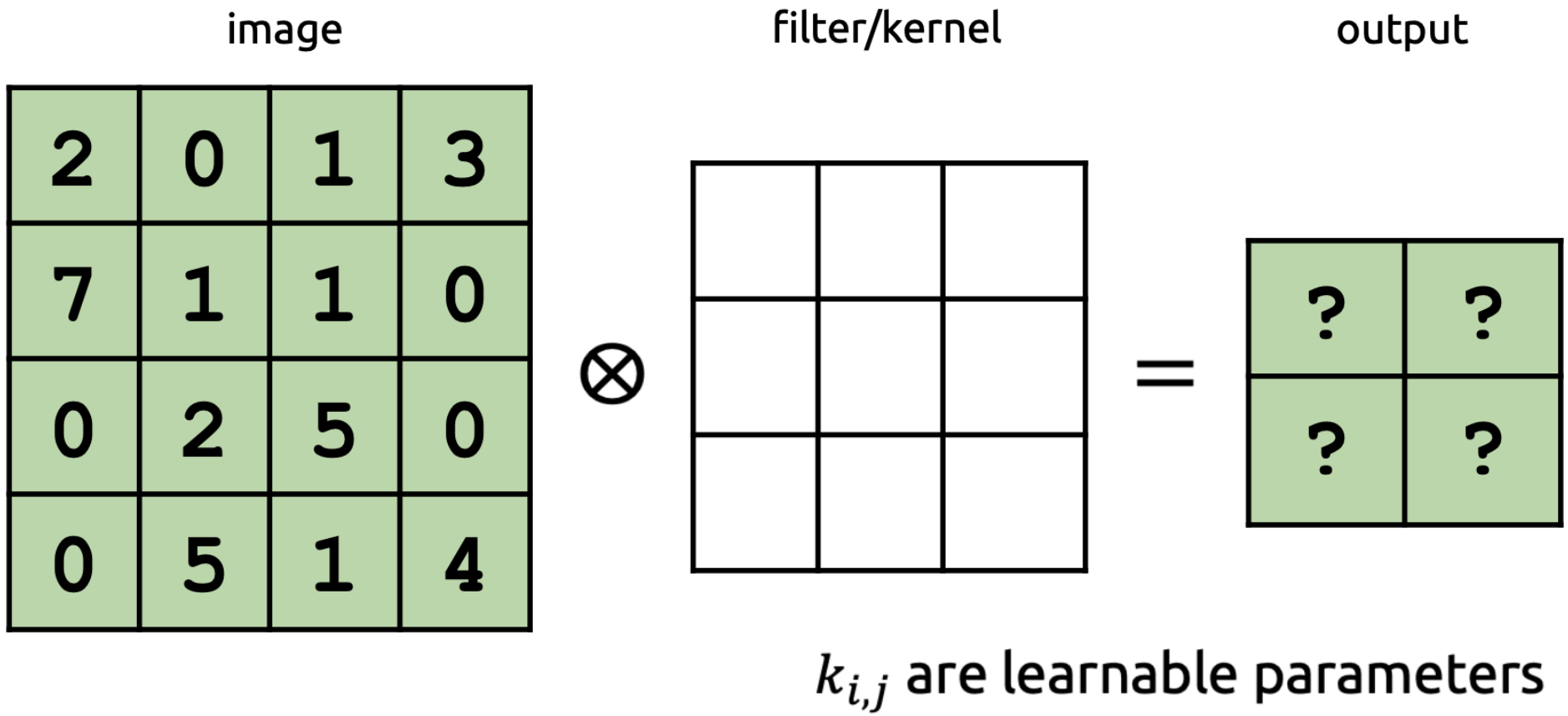


https://learnopencv.com/understanding-convolutional-neural-networks-cnn/

Key Idea 1: Filters are ***Learnable***

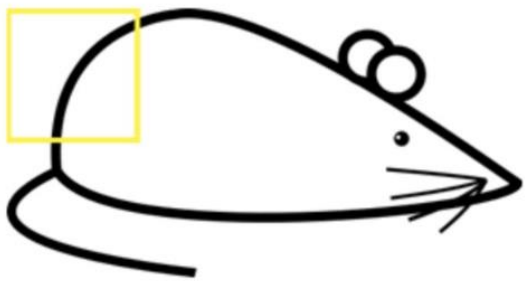


Key Idea 1: Filters are ***Learnable***



Key Idea 1: Filters are ***Learnable***

Original image

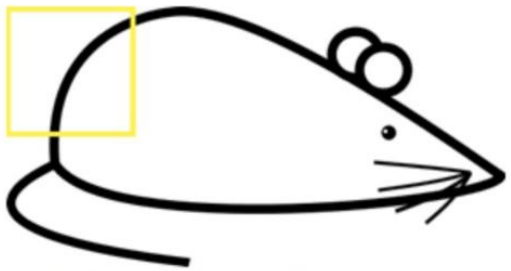


Visualization of the filter on the image

Label="Mouse"

Detecting patterns using learned filters

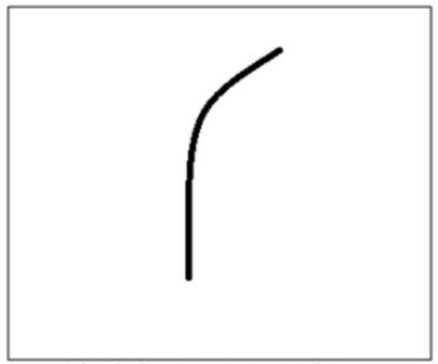
Original image



Visualization of the filter on the image

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Image courtesy: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

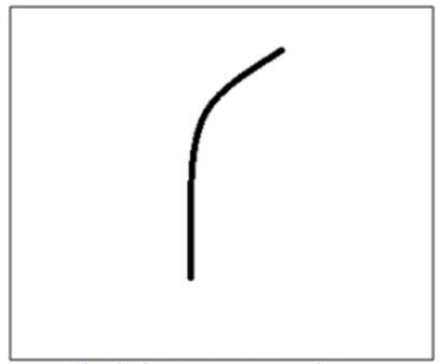
Detecting patterns using learned filters

Original image

How to detect other patterns?

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

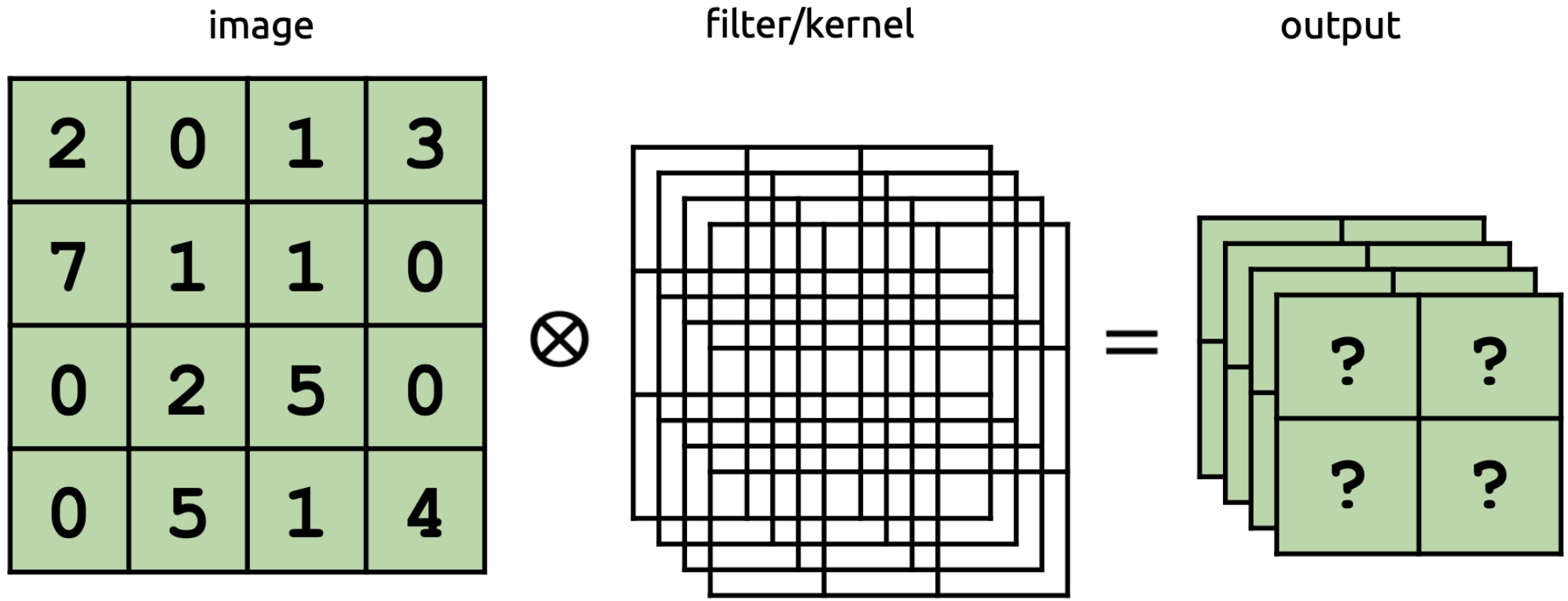
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

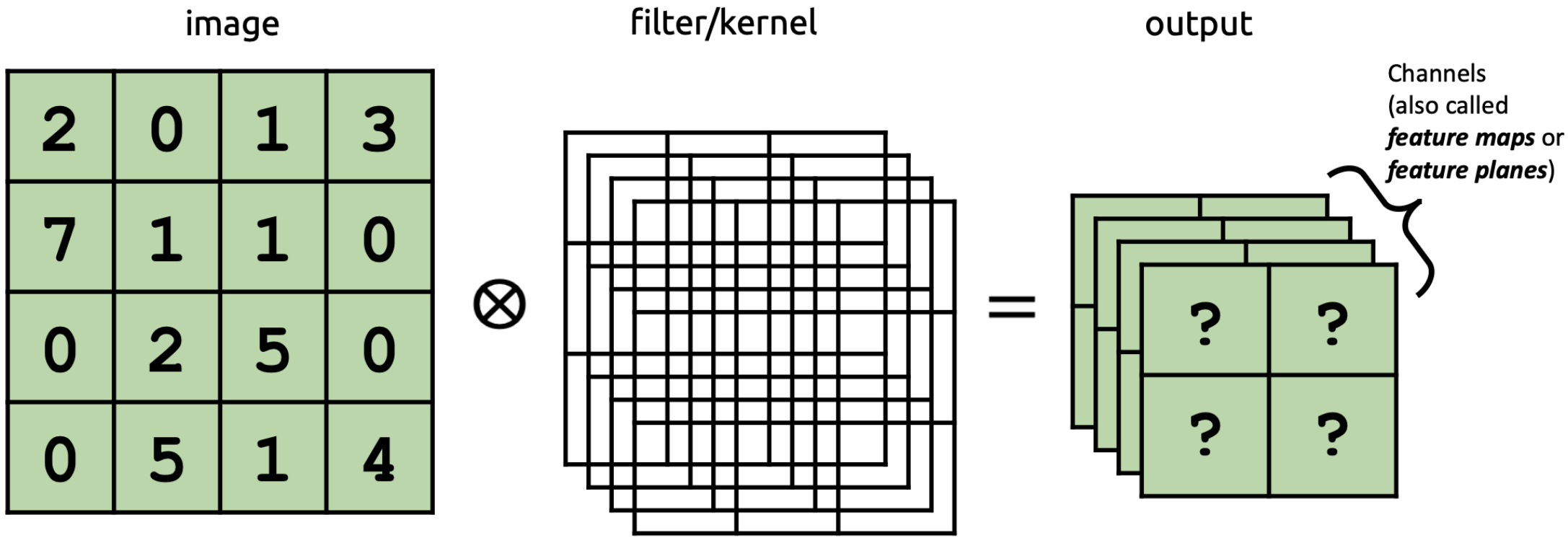
Image courtesy: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Key Idea 2: Learn ***many*** filters



This block of filters is called a ***filter bank***

Key Idea 2: Learn ***many*** filters



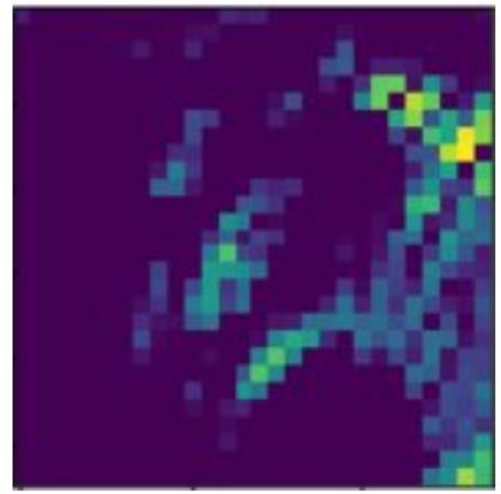
The output is now a multi-channel image

Any questions?

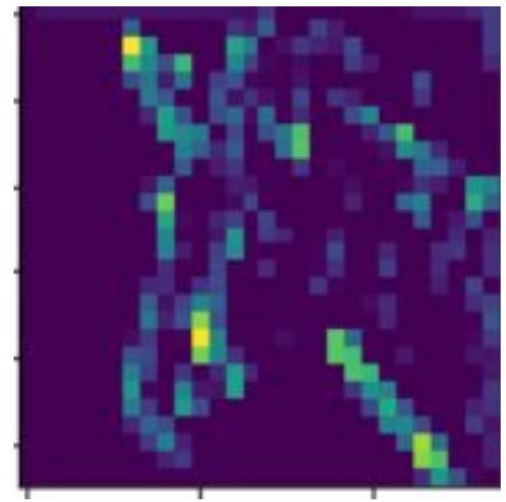
Key Idea 2: Learn ***many*** filters

- Why are multiple filters a good idea?
  - Can learn to extract different ***features*** of the image

Input image



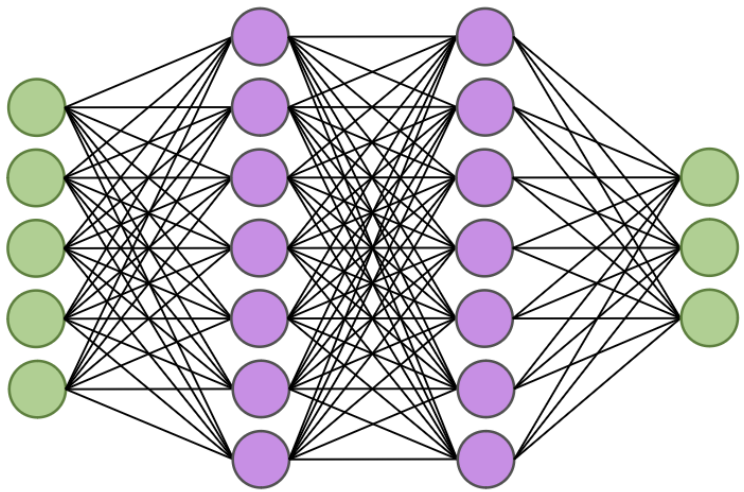
Output of filter 1



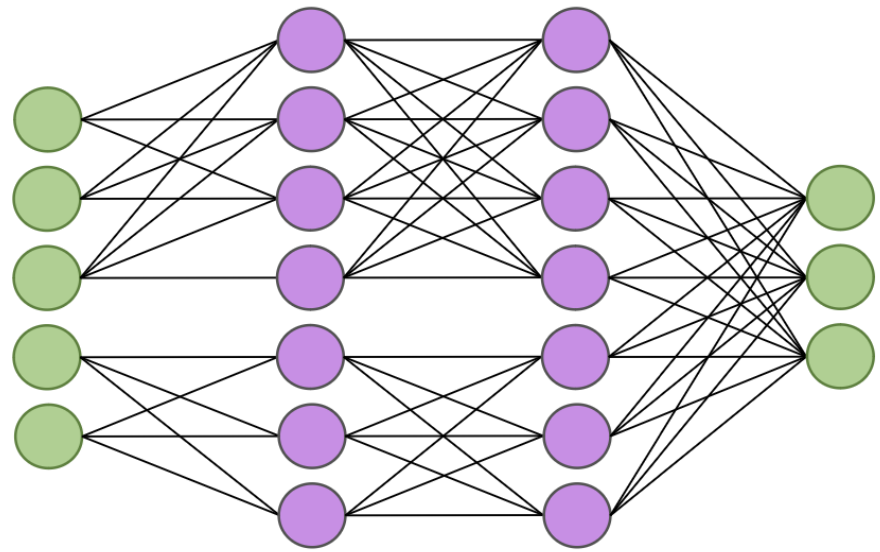
Output of filter 2

How is convolution “partially connected?”

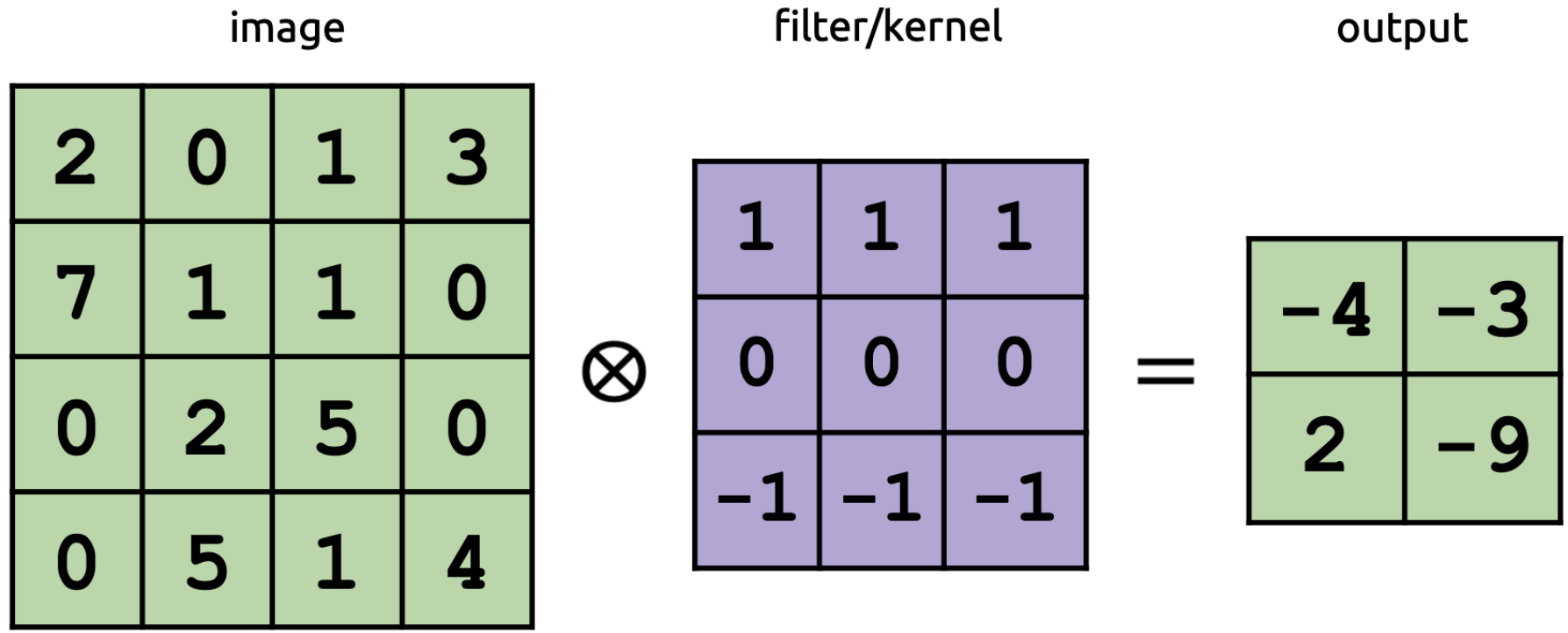
Fully Connected



Partially Connected

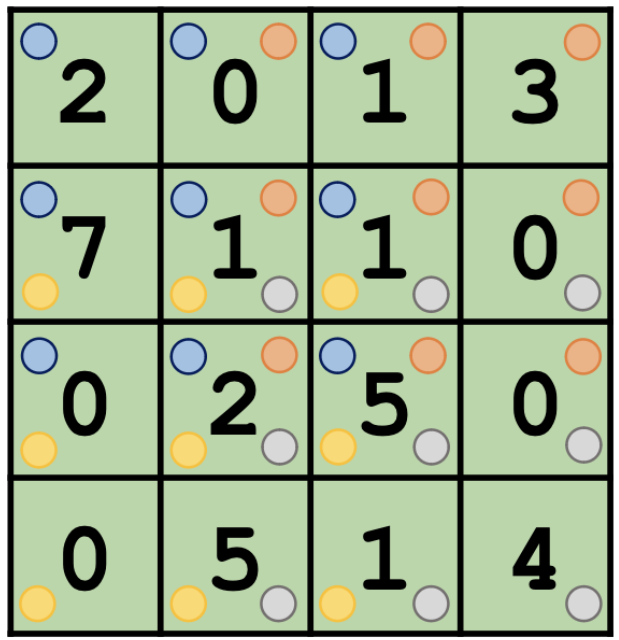


Only certain input pixels are “connected” to certain output pixels



Only certain input pixels are “connected” to certain output pixels

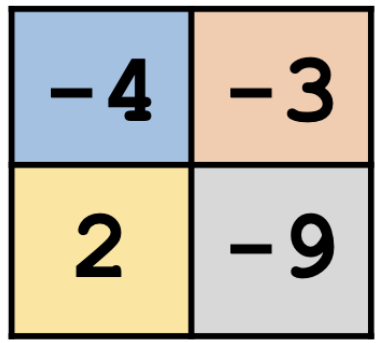
image



Colored dots in the input pixels represent which output pixels that input pixel contributes to

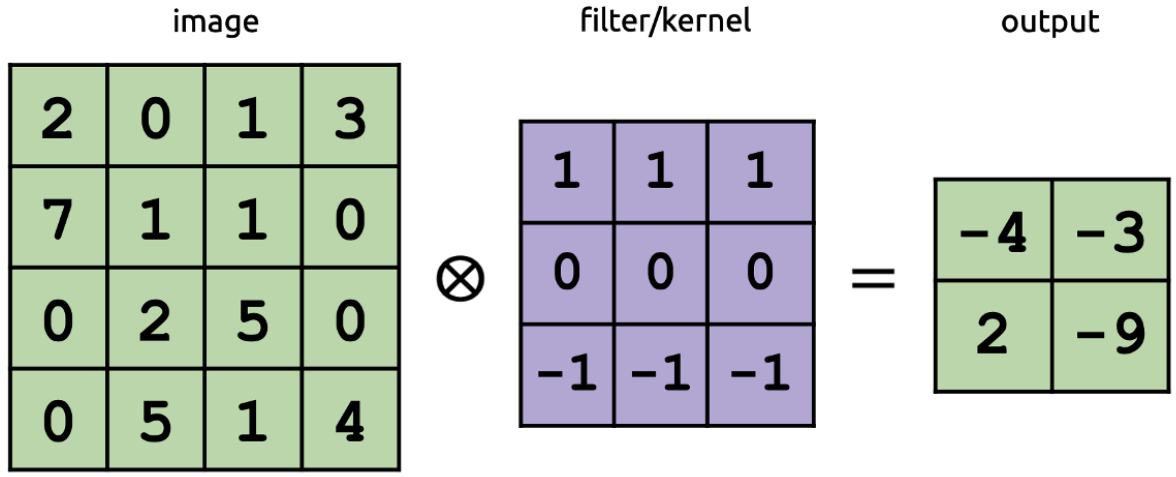
If this were fully connected, every input pixel would have all four output colors

output



Convolutions in Tensorflow

tf.nn.conv2d(input, filter, stride, padding)



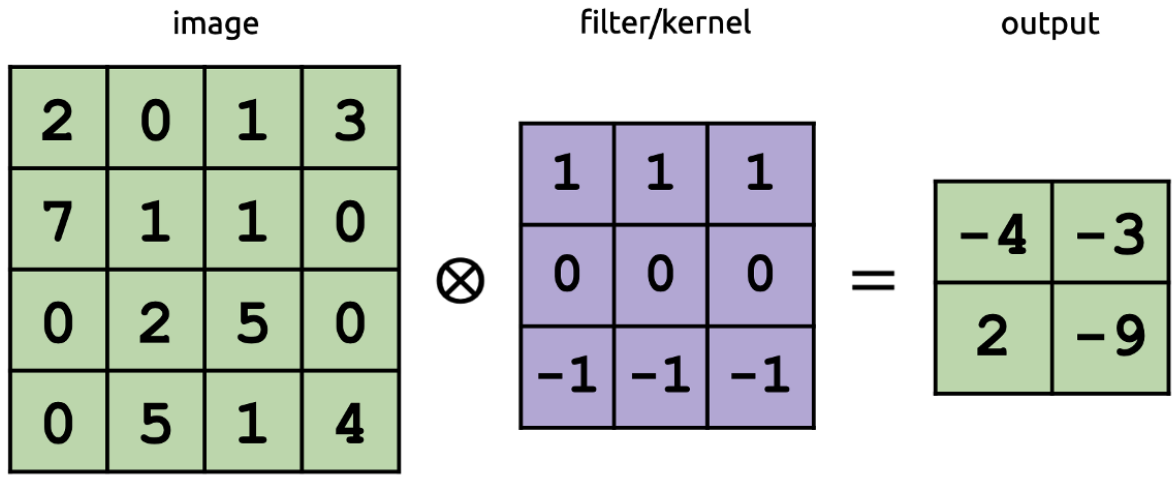
Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d

Convolutions in Tensorflow

tf.nn.conv2d(input, filter, stride, padding)

Input “Image”

Kernel/filter



Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d

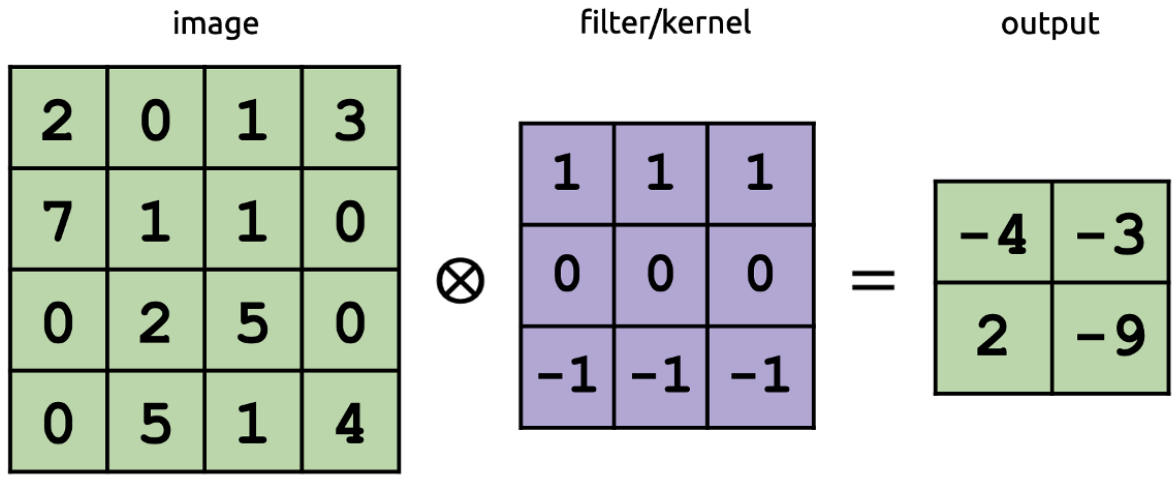
Convolutions in Tensorflow

tf.nn.conv2d(input, filter, stride, padding)

Input “Image”

Kernel/filter

What is the shape of the input?
Tensor of:[# items in batch, width, height, # channels]



Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d

Convolutions in Tensorflow

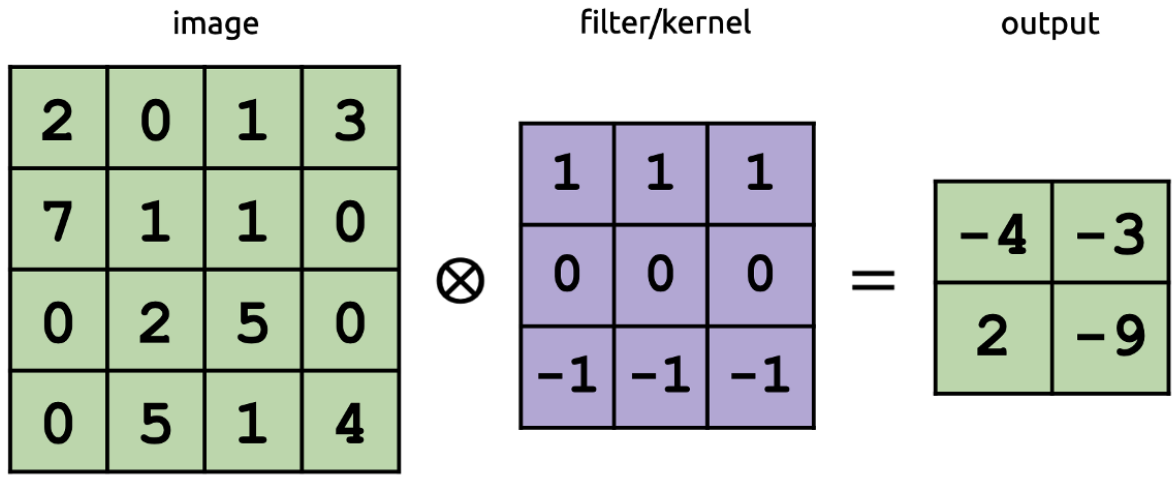
tf.nn.conv2d(input, filter, stride, padding)

Input “Image”

Kernel/filter

What is the shape of the input?
Tensor of:[# items in batch, width, height, # channels]

Channels: Number of input “colors”



Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d

Convolutions in Tensorflow

tf.nn.conv2d(input, filter, stride, padding)

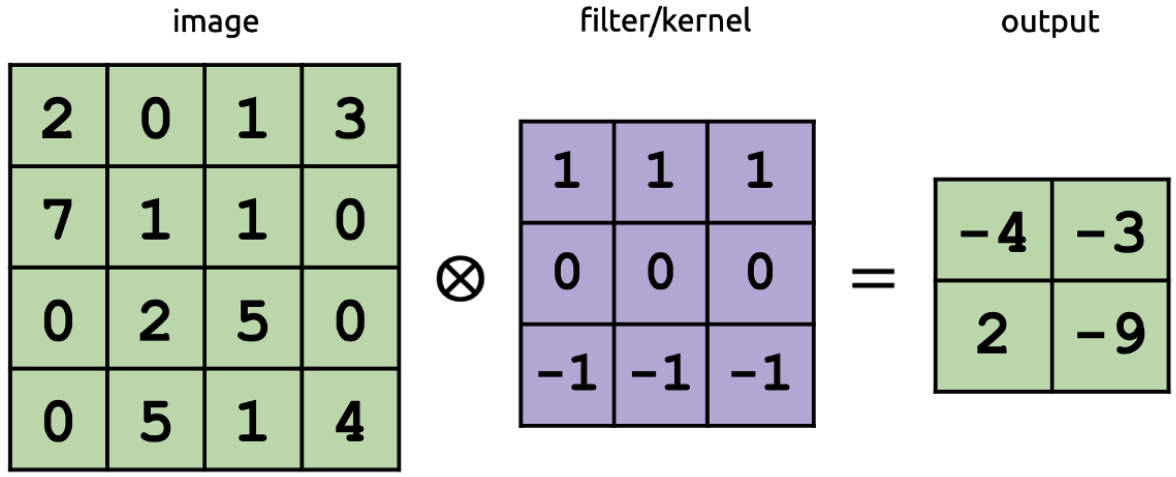
Input “Image”

Kernel/filter

What is the shape of the input?
Tensor of:[# items in batch, width, height, # channels]

Channels: Number of input “colors”

How can we determine the output size of a convolution?

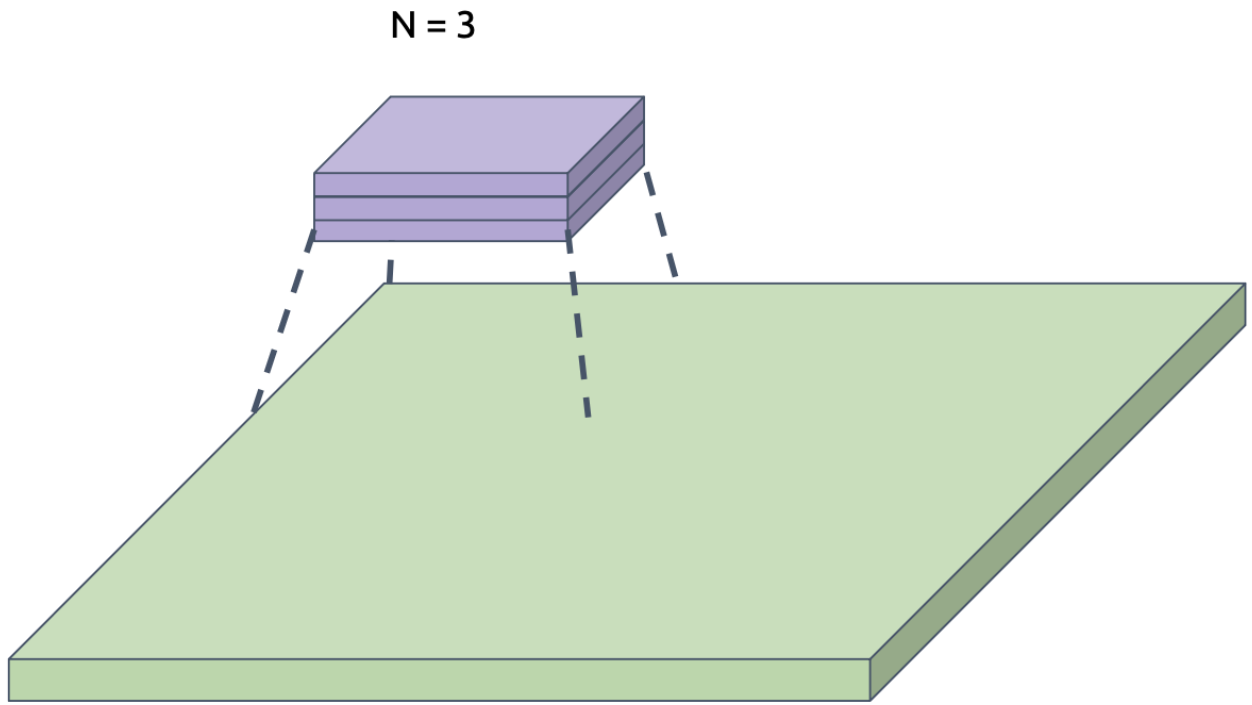


Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d

Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

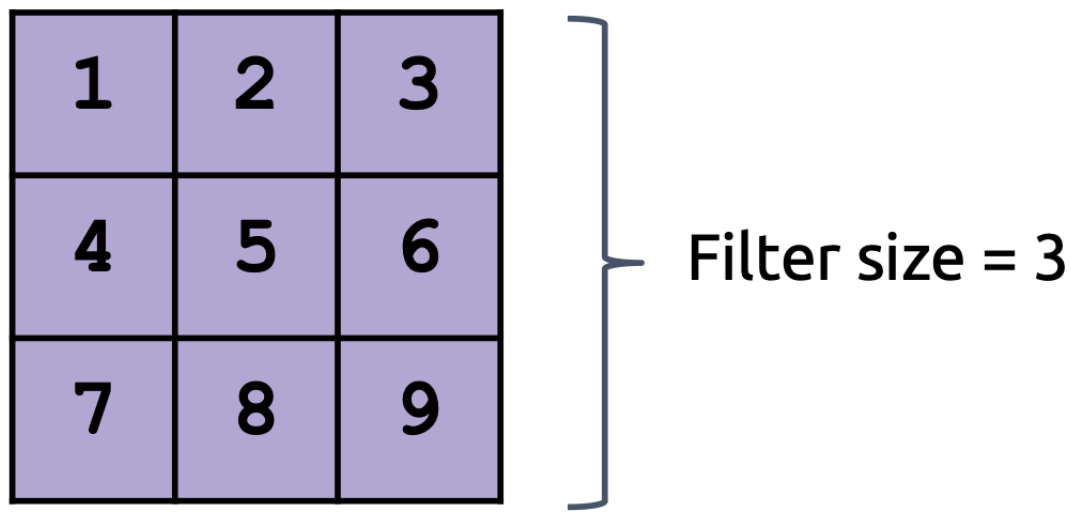
- Number of filters, **N**



Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

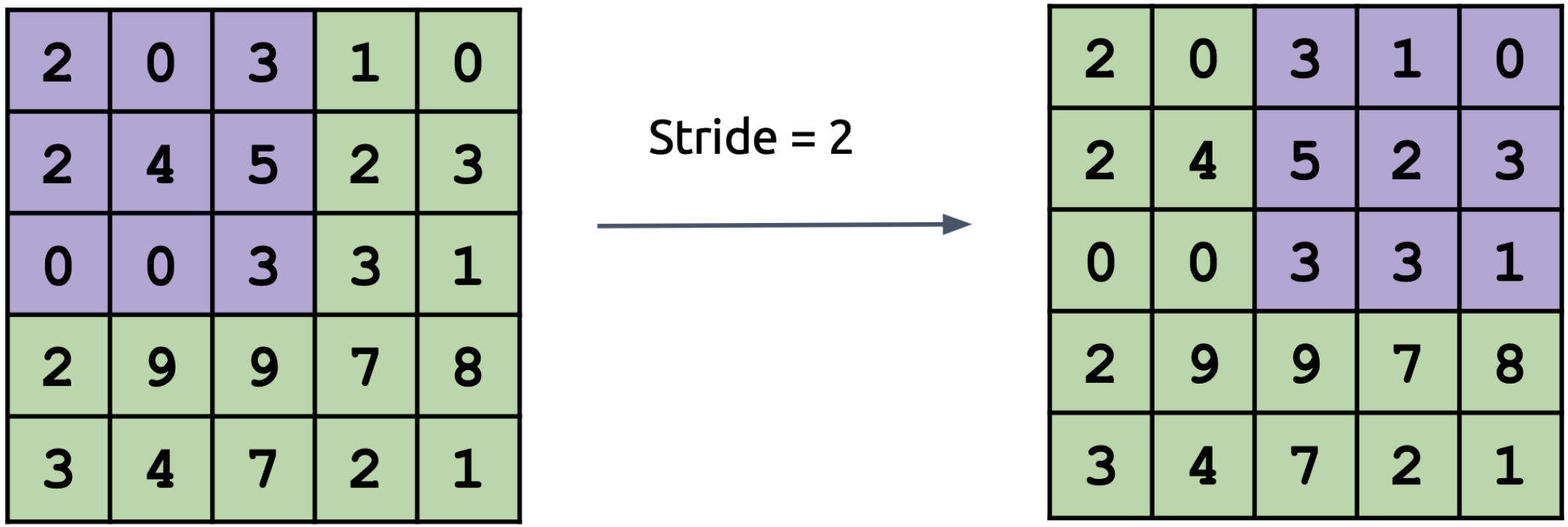
- Number of filters, **N**
- The size of these filters, **F**



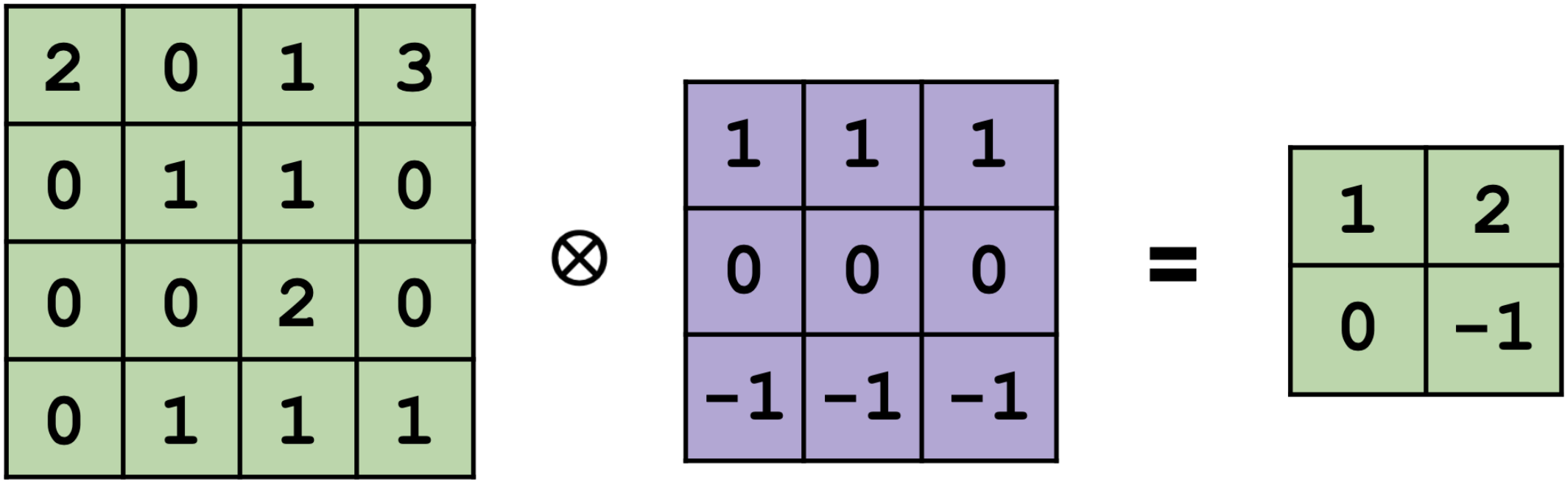
Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

- Number of filters, **N**
- The size of these filters, **F**
- The stride, **S**



“Problem” With Convolution



- Output of convolution is always smaller than the input
- Why might we want the output size to be the same?
  - To avoid the filter “eating at the border” of the image when applying multiple conv layers

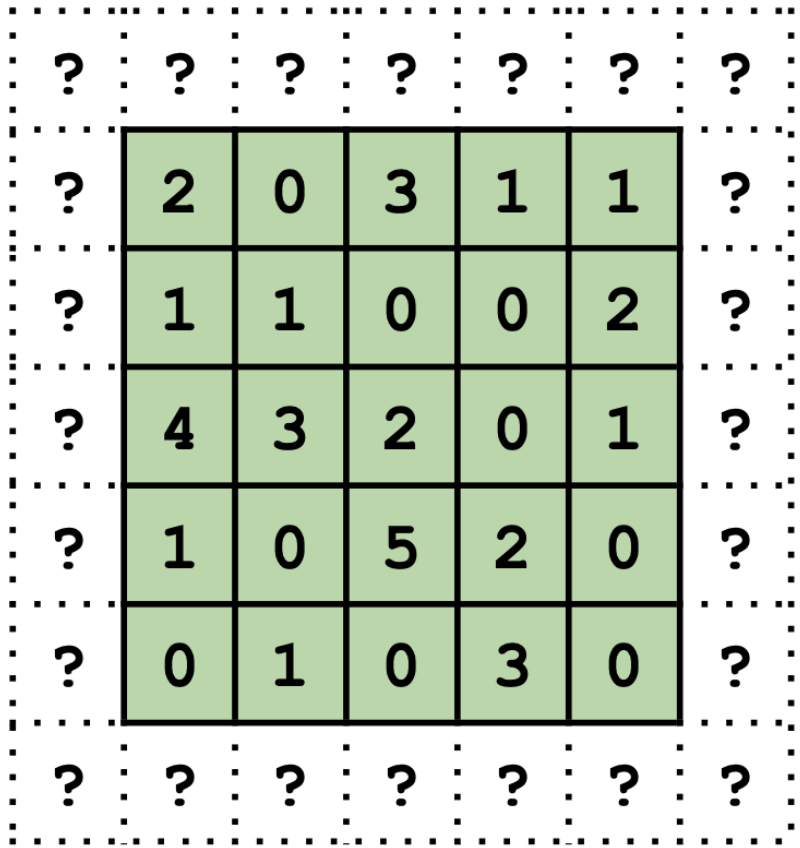
Solution: Padding

Apply the kernel to ‘imaginary’ pixels surrounding the image

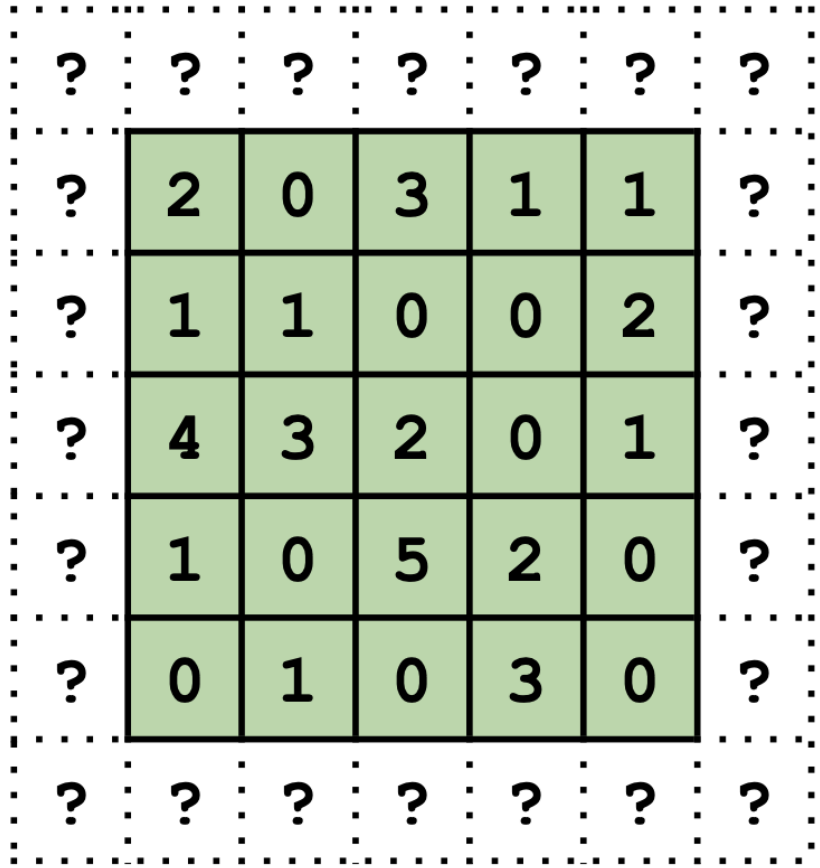
2	0	3	1	1
1	1	0	0	2
4	3	2	0	1
1	0	5	2	0
0	1	0	3	0

Solution: Padding

Apply the kernel to ‘imaginary’ pixels surrounding the image

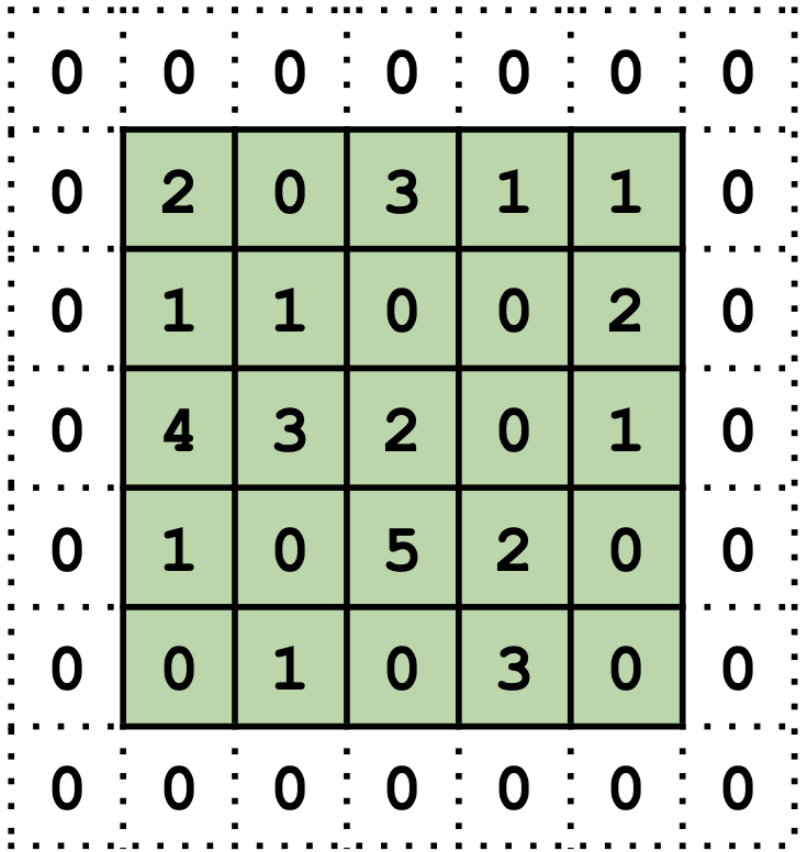


What Values to Use For These Pixels?



What Values to Use For These Pixels?

Standard practice: fill with zeroes



What Values to Use For These Pixels?

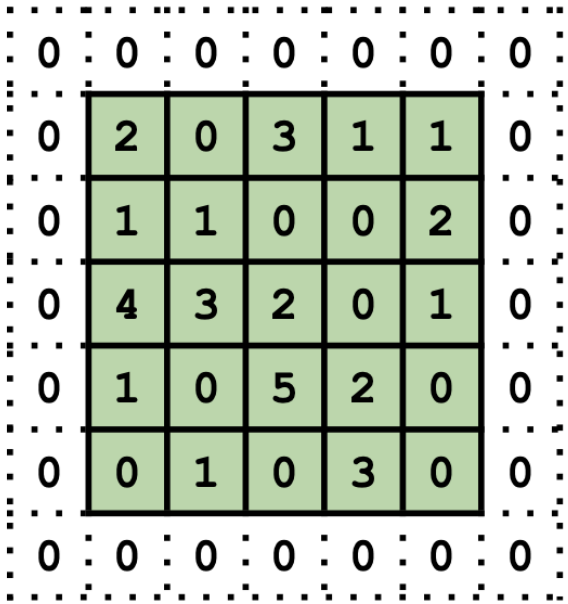
Standard practice: fill with zeroes

- Zero-valued padding pixels just result in some terms in the convolution sum being zero

$$V(x,y) = (\mathrm{I} \otimes K)(x,y) = \sum_m \sum_n I(x+m, y+n) K(m,n)$$

This is zero for a padding pixel

- End result: equivalent to a applying a ‘masked’ version of the filter that only covers the valid pixels



Padding Modes in Tensorflow

2 available options: ‘VALID’ and ‘SAME’:

Valid

Filter only slides over “Valid” regions of the data

2	0	1	3
0	1	1	0
0	0	2	0
0	1	1	1

Same

Filter slides over the bounds of the data, ensuring output size is the “Same” as input size (when stride = 1)

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

VALID Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
             padding='VALID')
```

2	0	1	3
0	1	1	0
0	0	2	0
0	1	1	1

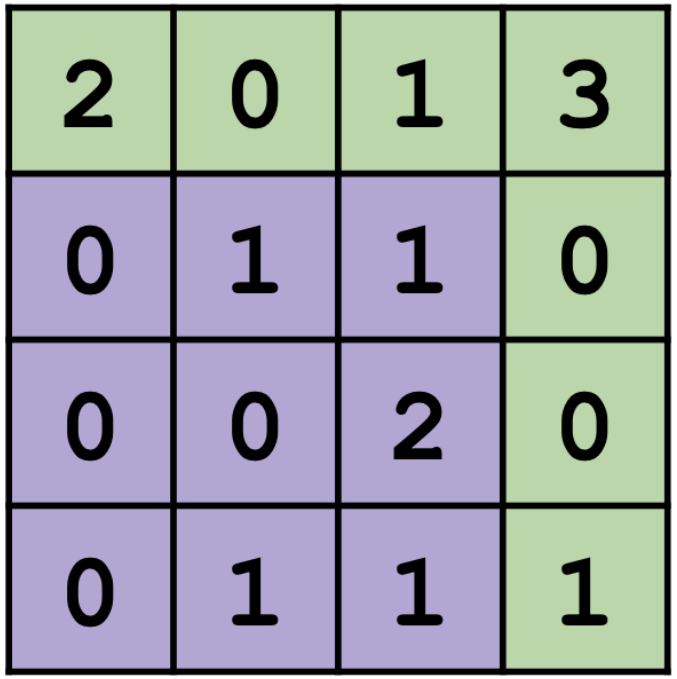
VALID Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
             padding='VALID')
```

2	0	1	3
0	1	1	0
0	0	2	0
0	1	1	1

VALID Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
             padding='VALID')
```

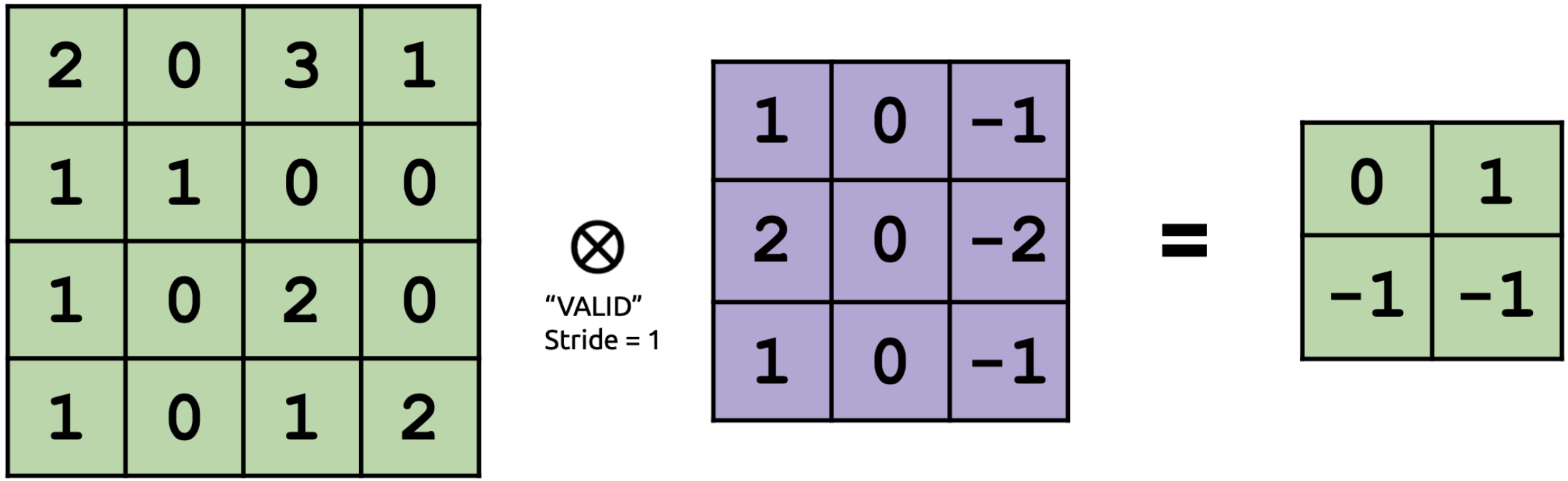


VALID Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
             padding='VALID')
```

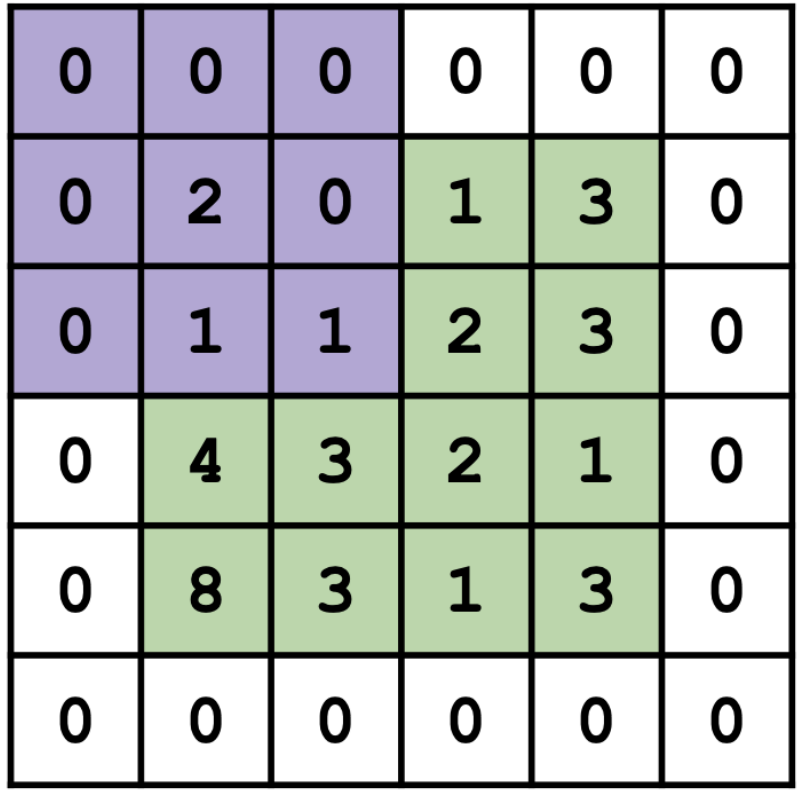
2	0	1	3
0	1	1	0
0	0	2	0
0	1	1	1

We already tried this! (reduced output size)



SAME Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
             padding='SAME')
```



SAME Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
             padding='SAME')
```

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

SAME Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
             padding='SAME')
```

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

SAME Padding in Tensorflow

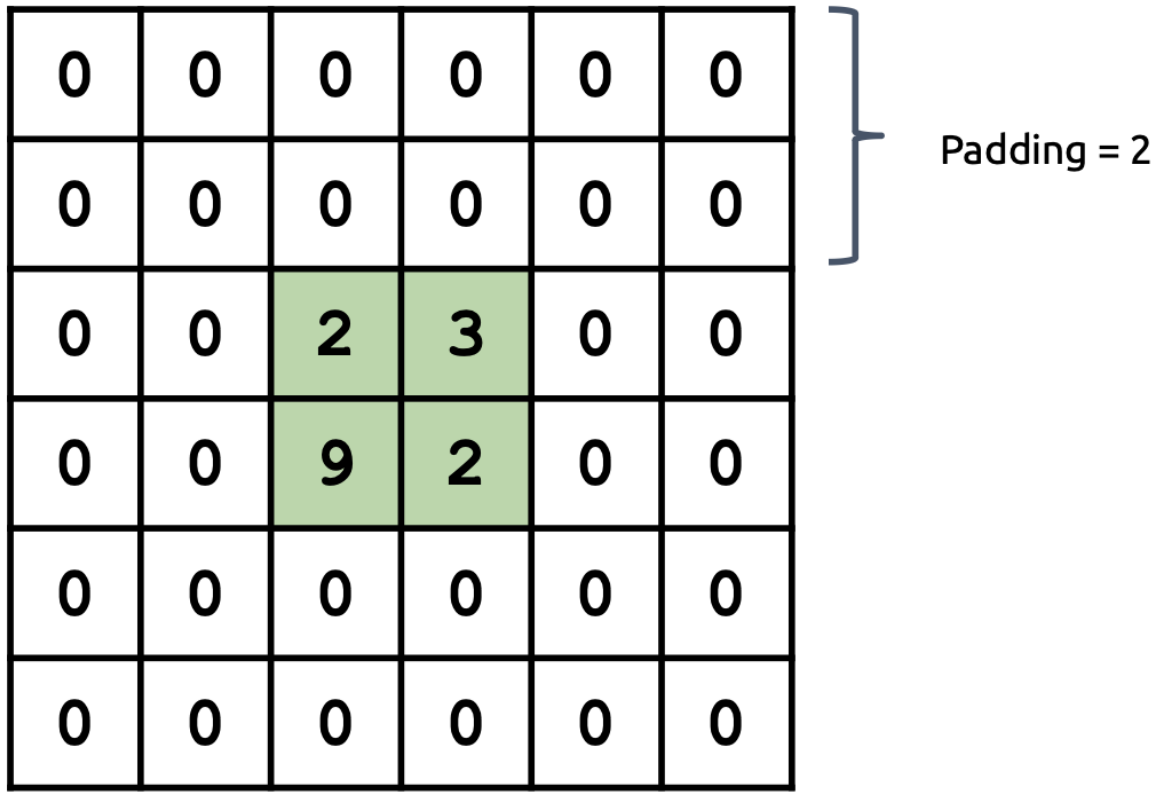
```
tf.nn.conv2d(input, filter, strides,
             padding='SAME')
```

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

Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

- Number of filters, **N**
- The size of these filters, **F**
- The stride, **S**
- The amount of padding, **P**



Output Size of a Convolution Layer

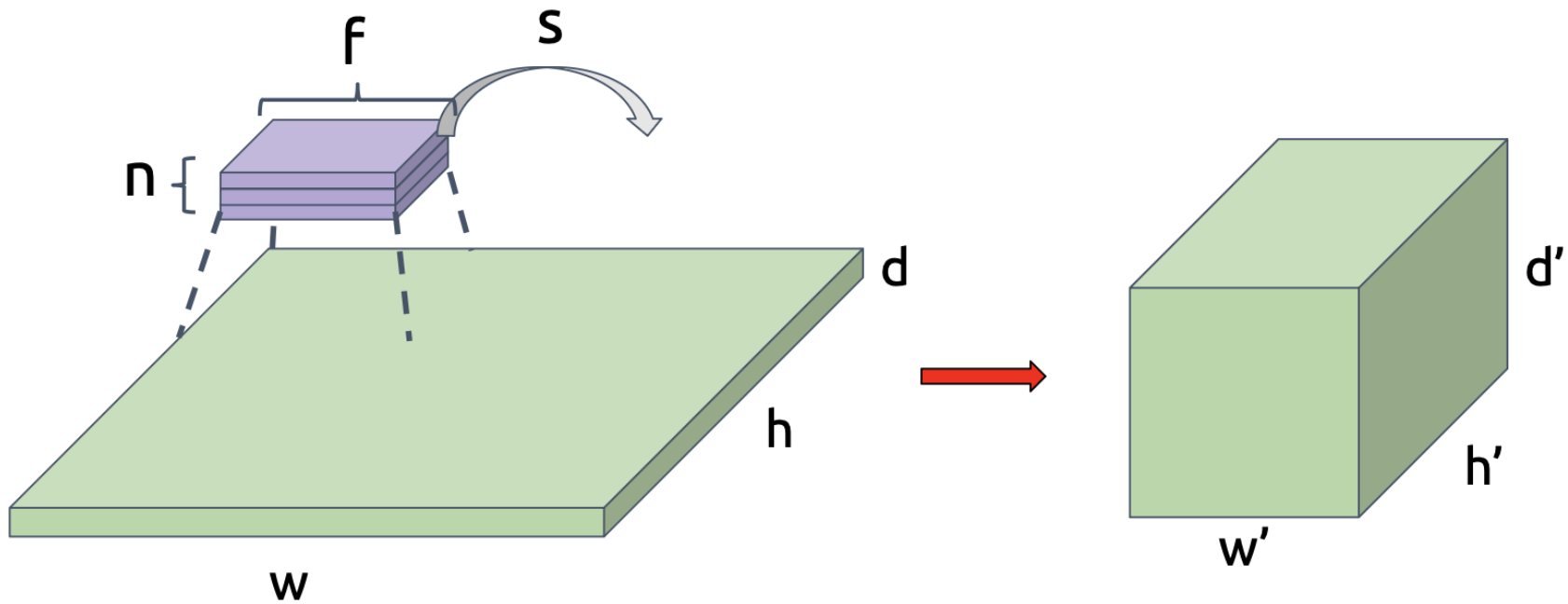
Suppose we know the number of filters, their size, the stride, and padding (**n,f,s,p**).

Then for a convolution layer with input dimension **w x h x d**, the output dimensions **w' x h' x d'** are:

$$w' = \frac{w - f + 2p}{s} + 1$$

$$h' = \frac{h - f + 2p}{s} + 1$$

$$d' = n$$



Output Size for “VALID” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

Let $w = 4$

num filters $n = 1$
filter size $f = 3$
stride $s = 1$
padding $p = 0$

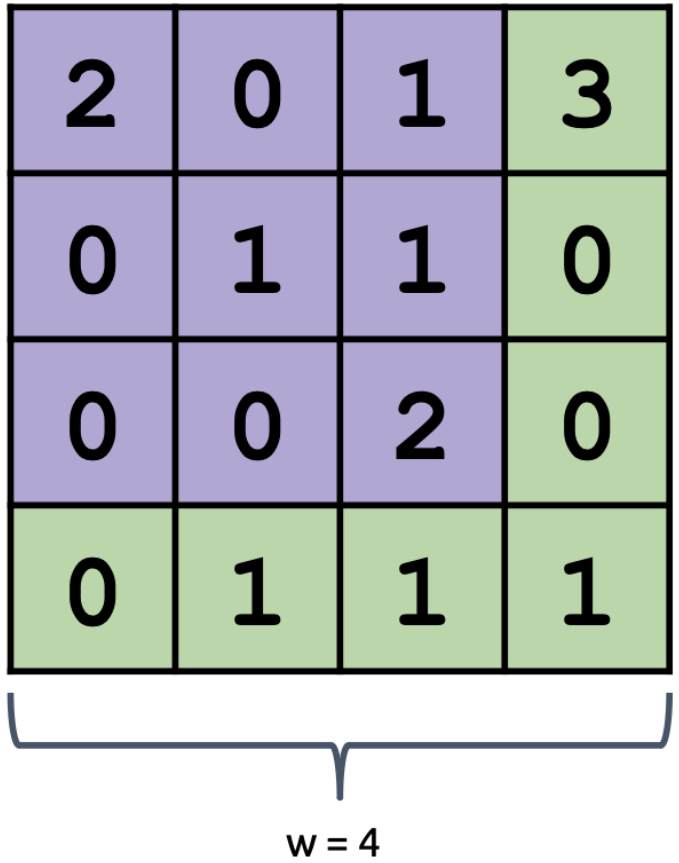
$$w' = \frac{4 - 3 + 2 \cdot 0}{1} + 1$$

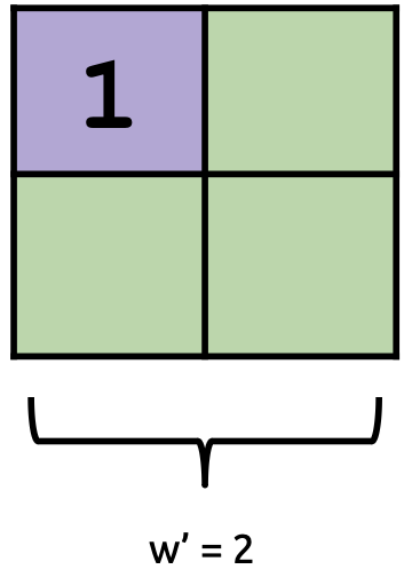
$$= 1 + 1 = 2$$

Output Size for “VALID” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 0

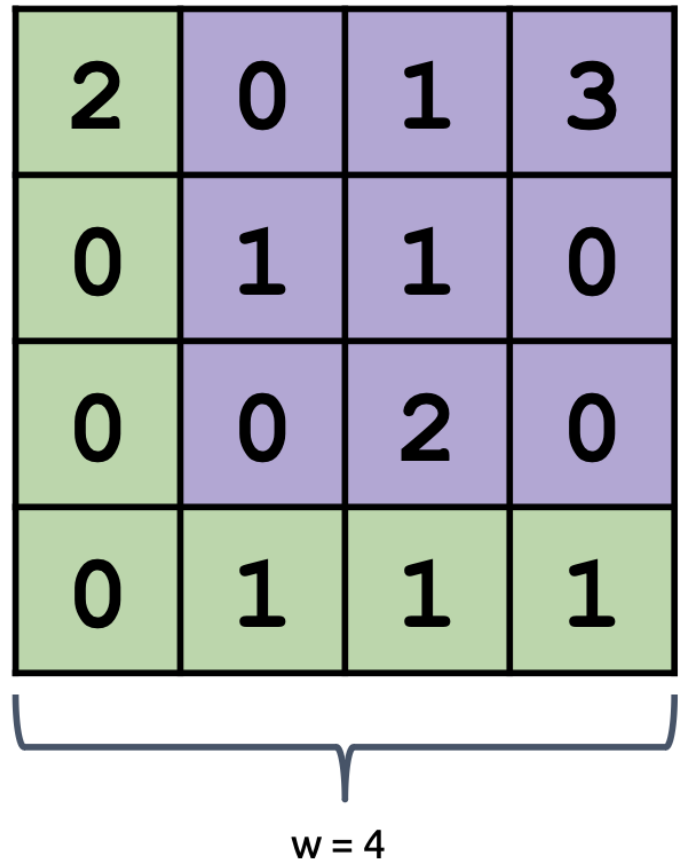


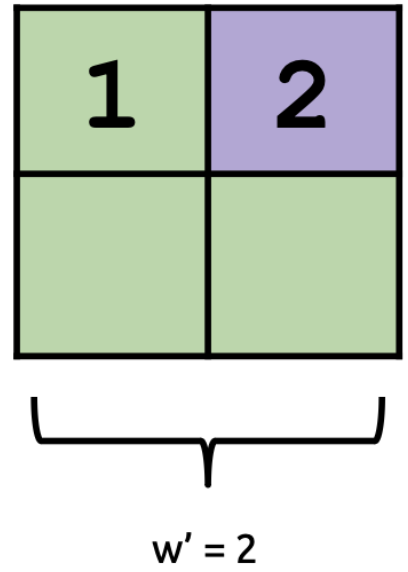


Output Size for “VALID” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 0

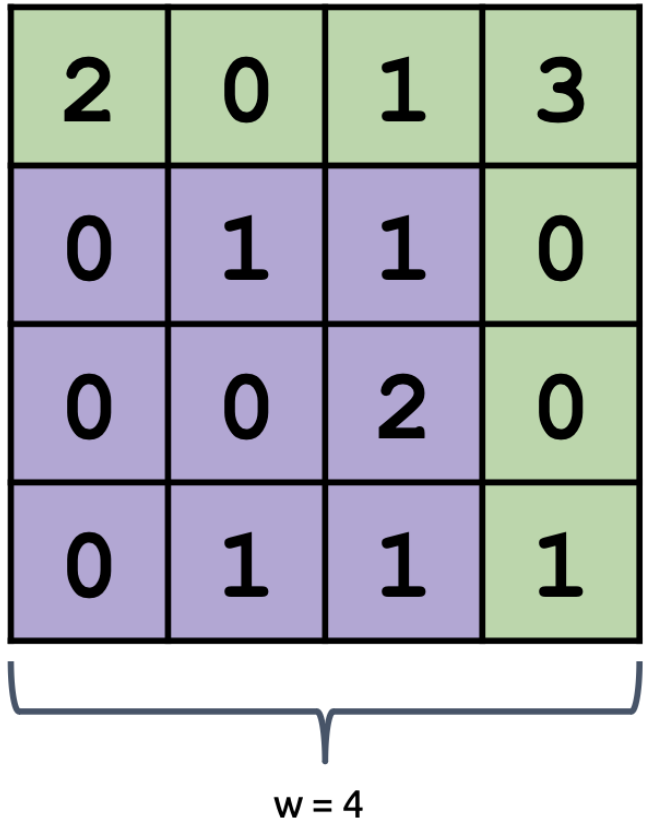


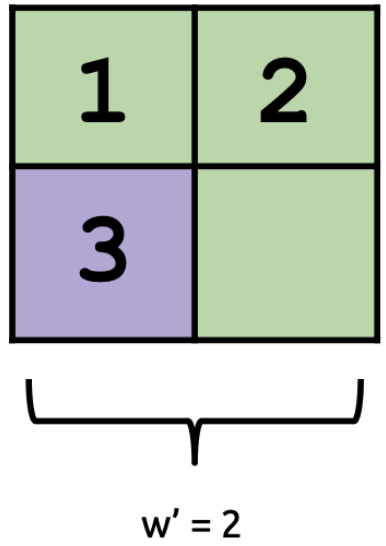


Output Size for “VALID” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 0

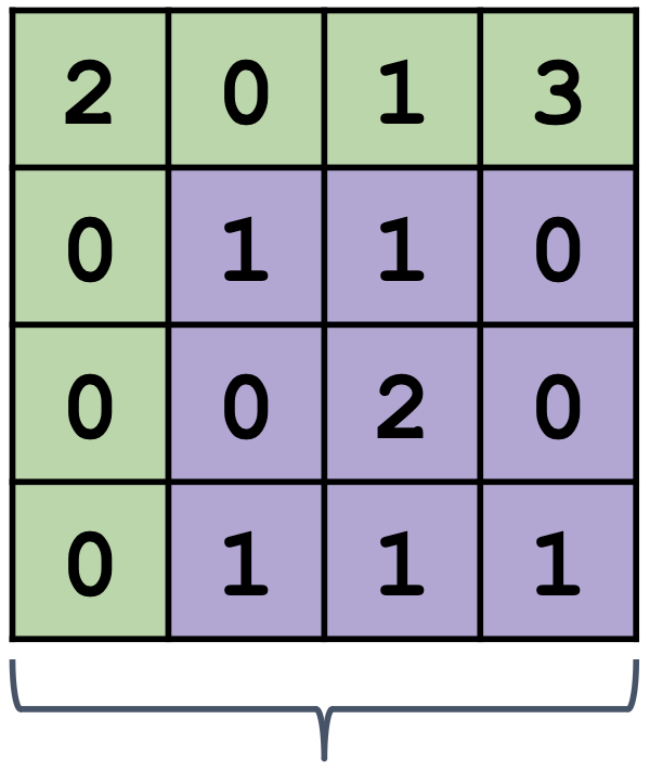


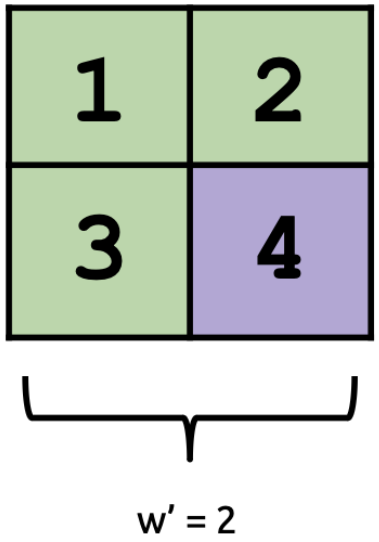


Output Size for “VALID” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 0





Output Size for “SAME” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

Let $w = 4$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

$$w' = \frac{4 - 3 + 2 \cdot 1}{1} + 1$$

$$= 3 + 1 = 4$$

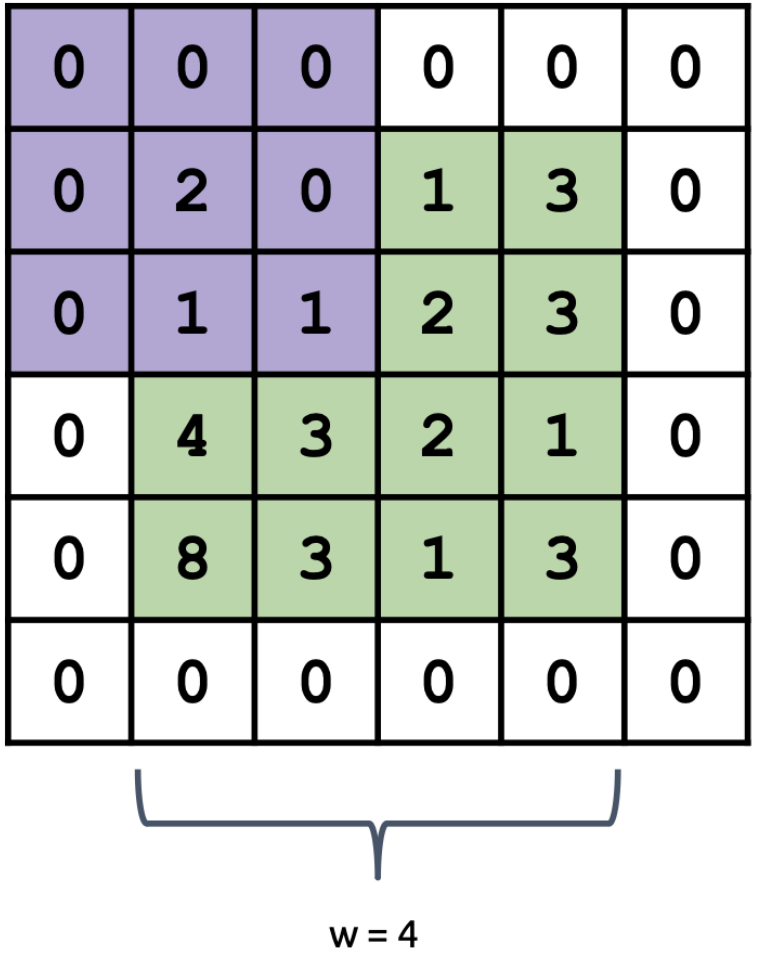
Padding size needs to be determined

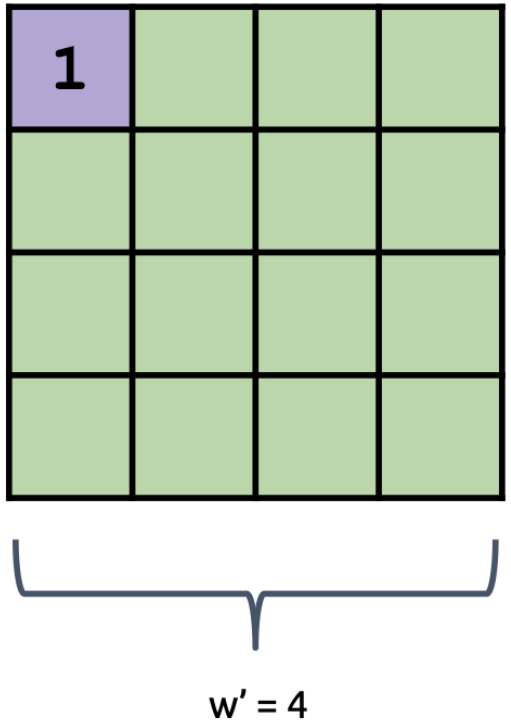
Output Size for “SAME” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined



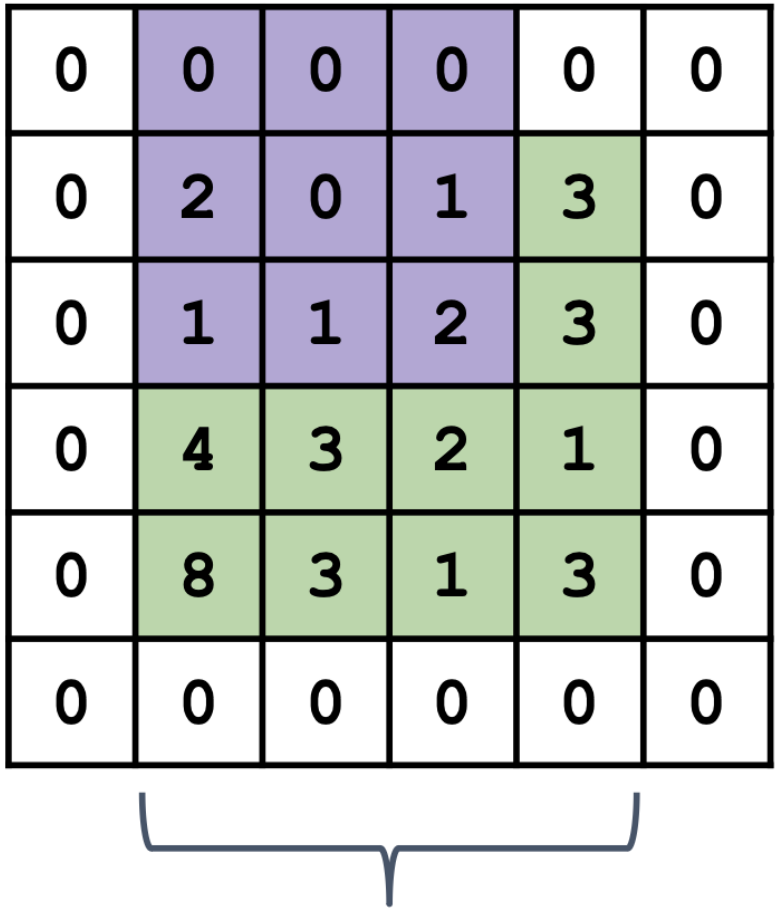


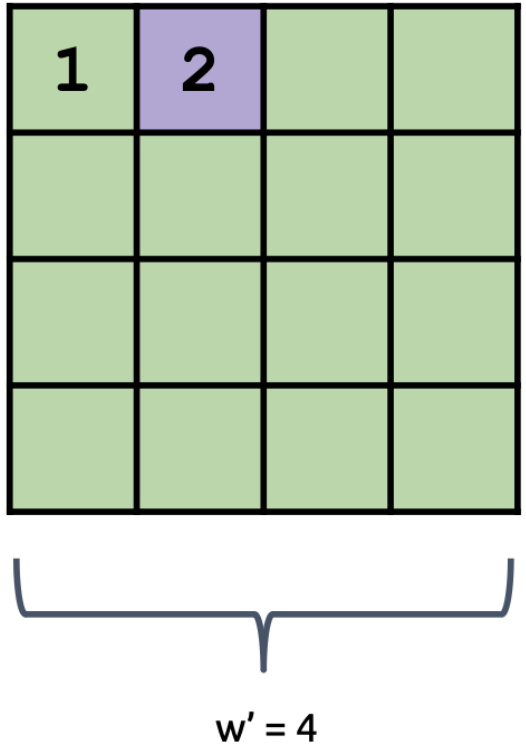
Output Size for “SAME” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined



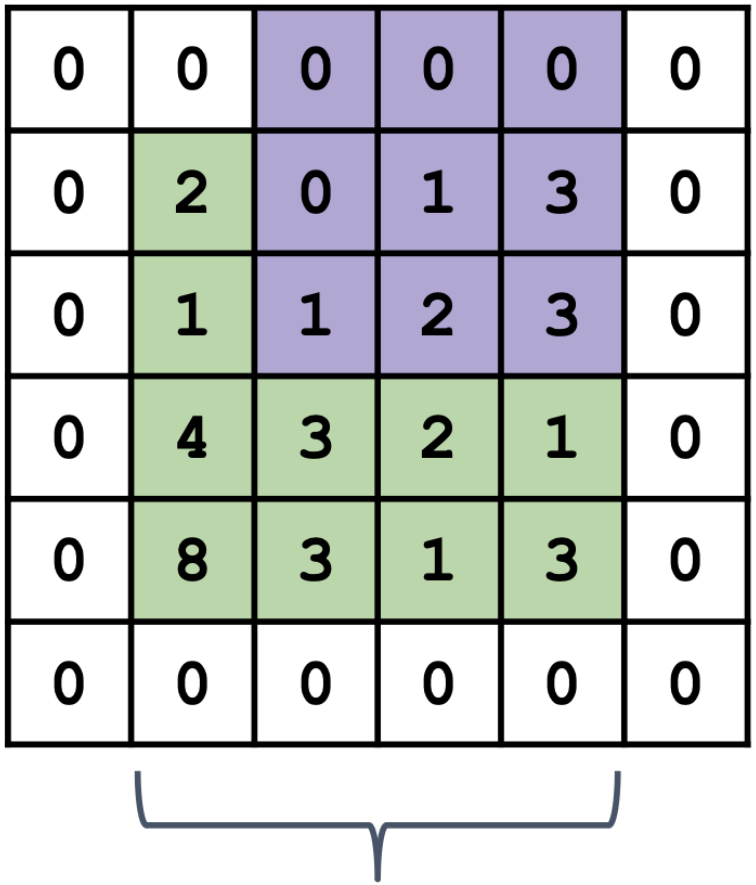


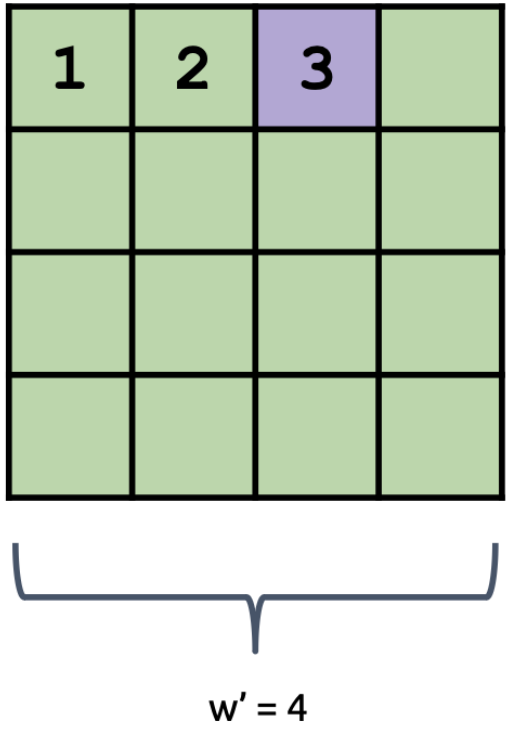
Output Size for “SAME” Padding

$$w' = \frac{w - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined





Output Size for “SAME” Padding

Any questions?

$$w' = \frac{w - f + 2p}{s} + 1$$

num filters n = 1
filter size f = 3
stride s = 1
padding p = 1*

Padding size needs to be determined

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

w = 4

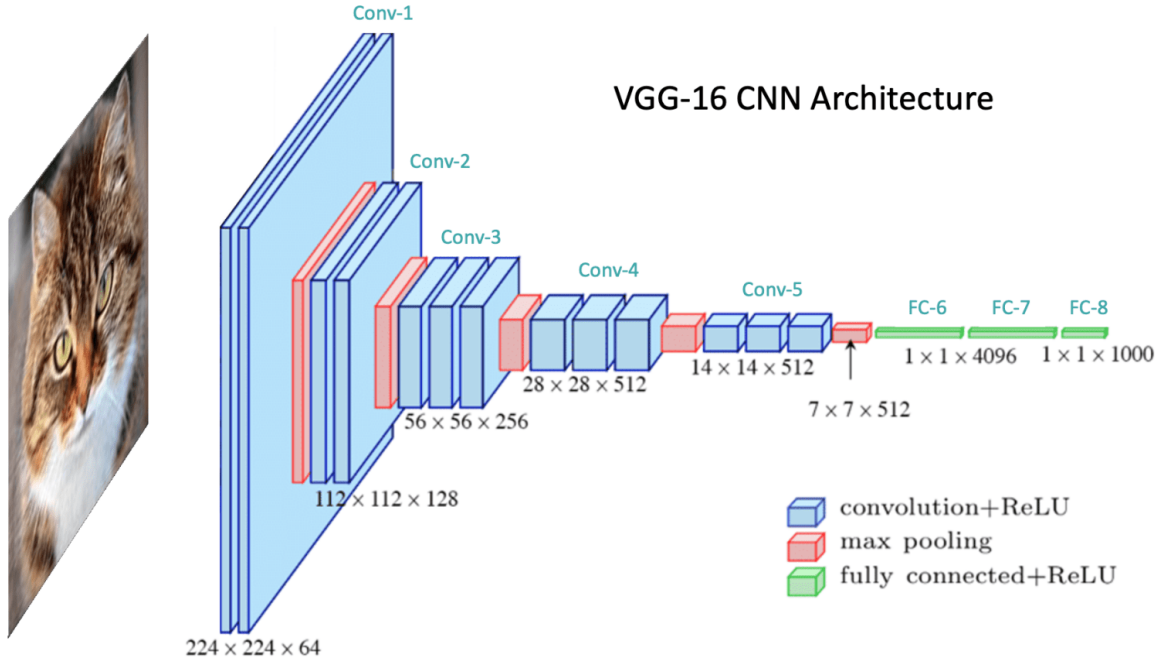
1	2	3	4

w’ = 4

Getting network output

Remaining Question: If the convolution creates another [h x w x d] tensor, how do we actually get an output?

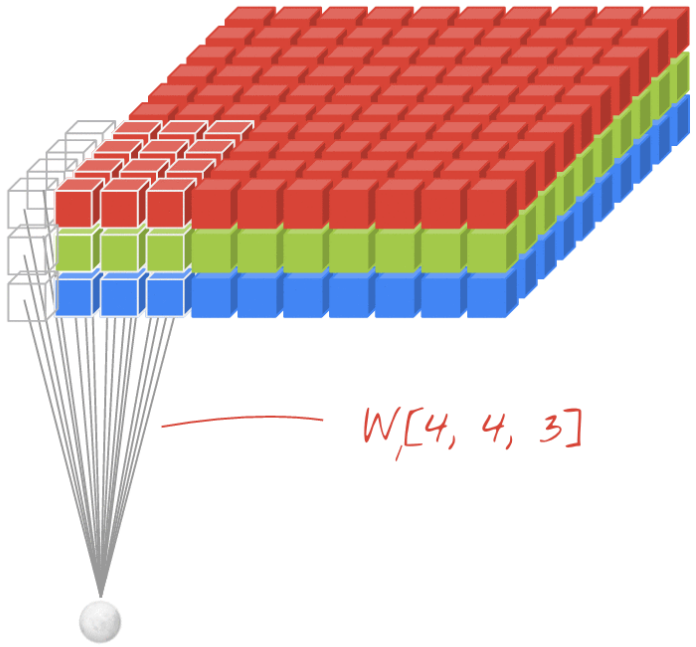
How can we turn use convolutions for classification?



https://learnopencv.com/understanding-convolutional-neural-networks-cnn/

Color images…

Remaining Question: What if our input has multiple channels (colors)? Do we apply filters to each individual color matrix? Or in some other way?



Source: Martin Görner

Recap

2	0	3	1
1	1	0	0
1	0	2	0
1	0	1	2

⊗ "VALID" Stride = 1

1	0	−1
2	0	−2
1	0	−1

=

0	1
−1	−1

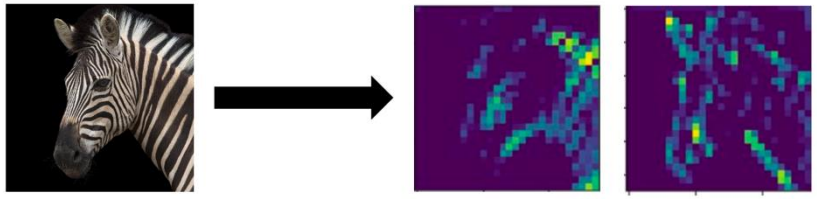
Filters and Stride

Learning Filters

Convolutions are partially connected

Padding

Tensorflow Conv2d Function




```
tf.nn.conv2d(input, filter, strides, padding)
```

Input Image (4-D Tensor)

Filter/Kernel (4-D Tensor)

Strides along each dimension

Type of Padding (String "Valid" or "Same")