AI Deep Learning: Convolutional Neural Networks (II)

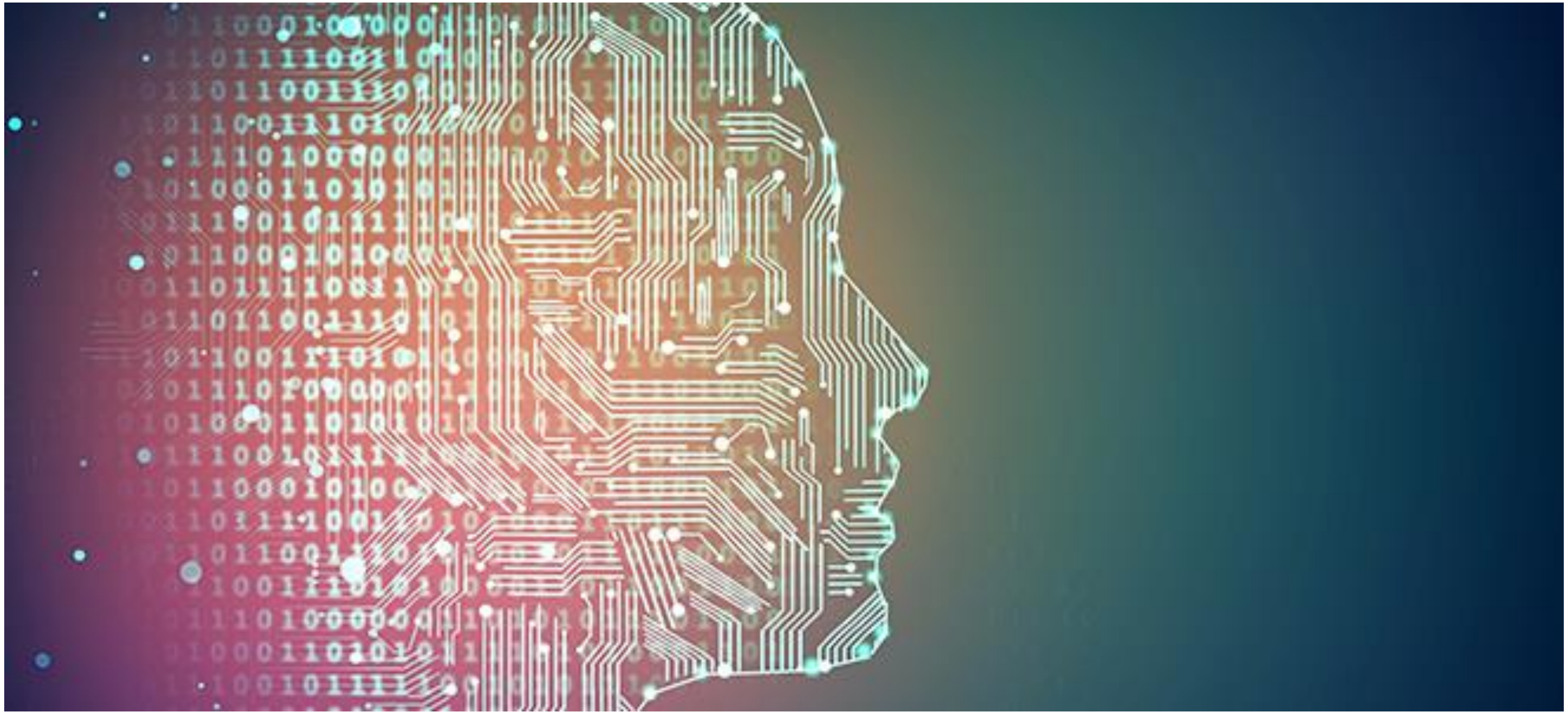
Deep Learning with Big Data

Thuan L Nguyen, PhD

AI Deep Learning: Convolutional Neural Networks (CNN)

1. AI Deep Learning: Convolutional Neural Networks Overview
2. AI Deep Learning: Properties of Images
3. AI Deep Learning: Convolution Layers in Convolutional Neural Networks
4. AI Deep Learning: Pooling Layers in Convolutional Neural Networks
5. AI Deep Learning: Convolutional Neural Networks: Architectures: Overview
6. AI Deep Learning: Convolutional Neural Networks: Architectures: Layers
7. AI Deep Learning: Build and Train Convolutional Neural Networks

AI Deep Learning: Convolutional Neural Networks (CNN)



AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Convolution(al) Operation, Feature Maps, and Receptive Fields

- The **convolution operation** is performed by sliding the filter over the input.
- At every location, **element-wise matrix multiplication** is done and the **results** are **summed**.
- The sum goes into the **feature map**.
- The green area where the convolution operation takes place is called the **receptive field**.
  - Due to the size of the filter, the receptive field is also 3x3.

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

Input x Filter

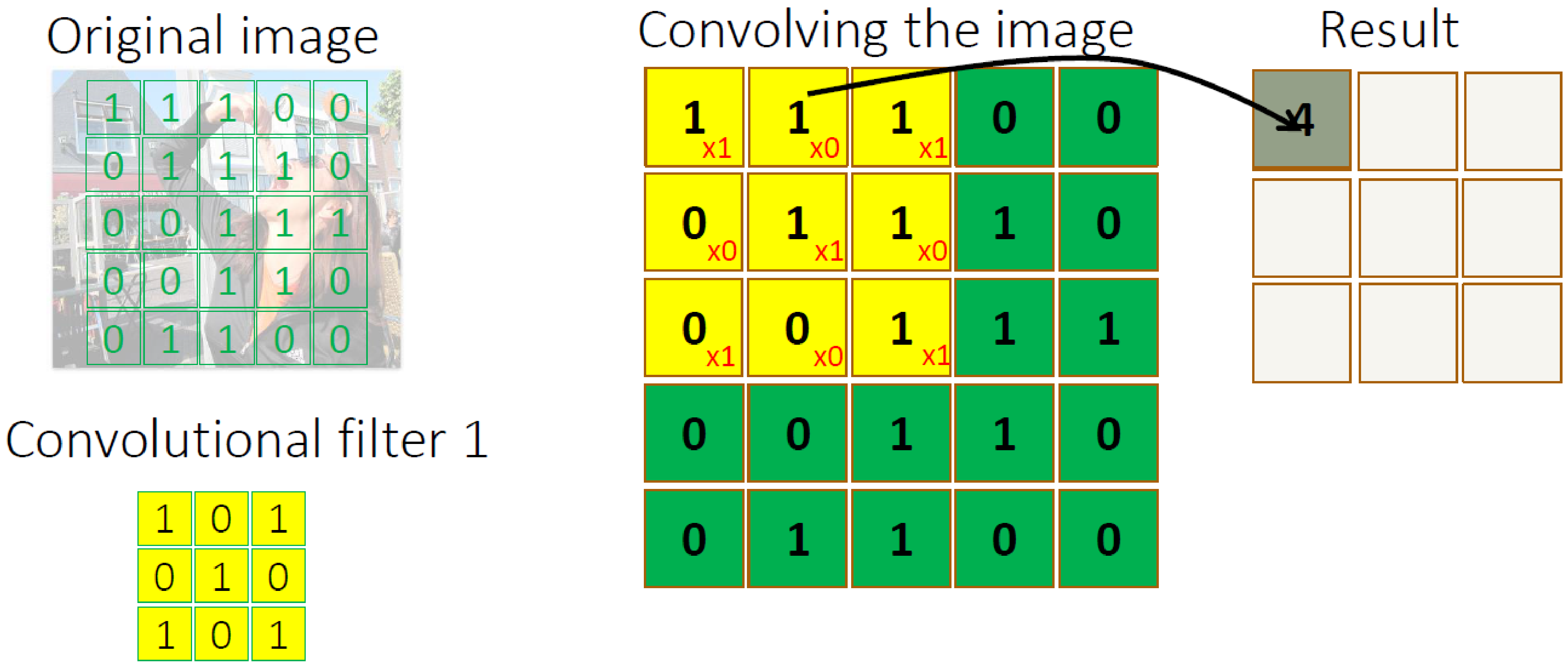
4		

Feature Map

AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Examples of Convolution(al) Operation

- This was an example convolution operation shown in 2D using a 3x3 filter:



AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Examples of Convolution(al) Operation

- This was an example convolution operation shown in 2D using a 3x3 filter:

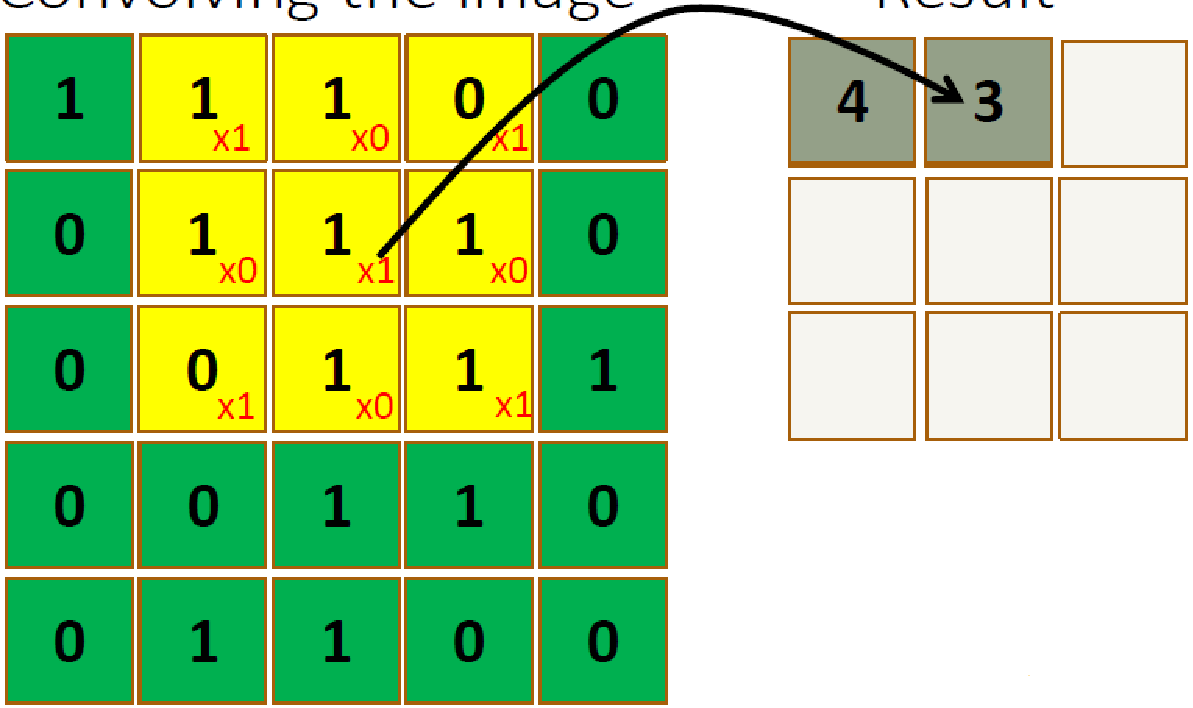
Original image

Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image

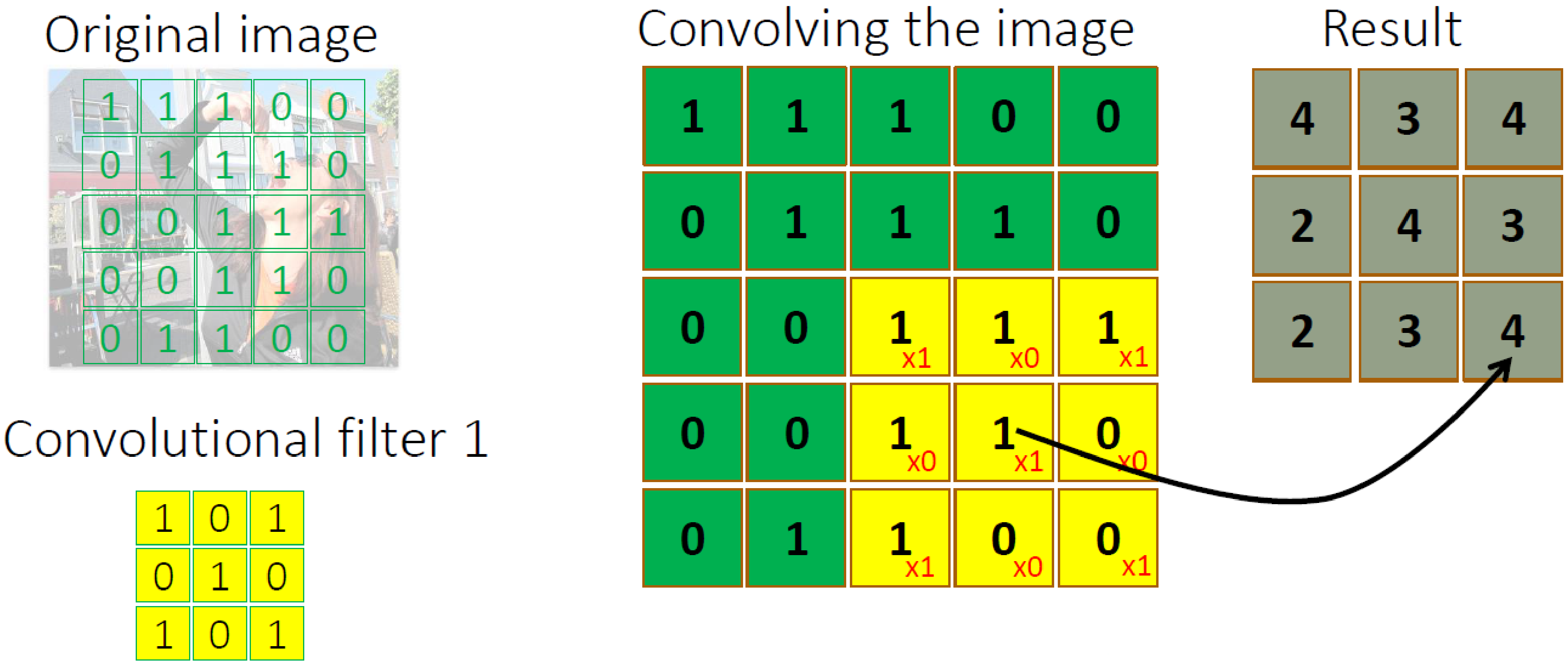
Result



AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Examples of Convolution(al) Operation

- This was an example convolution operation shown in 2D using a 3x3 filter:



AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Convolution(al) Operations & Final Outputs

- In previous slides is an example convolution operation shown in 2D using a 3x3 filter.

- **In reality**:
  - An image is represented as a **3D matrix** with dimensions of **height, width** and **depth**,
    - Where depth corresponds to color channels (RGB).

- A 2D **convolutional layer → 2D filter** has a specific height and width, like **3x3** or 5x5,

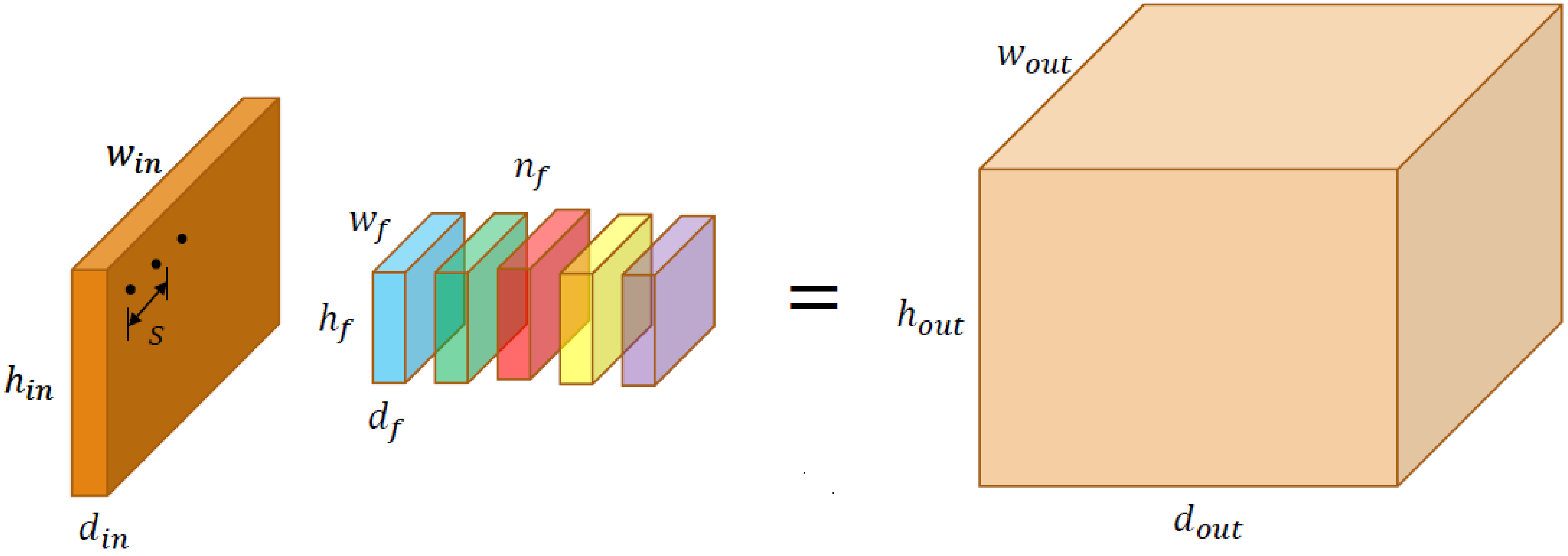
- With actual convolution operations:
  - **Multiple convolutions** are performed on an input:
    - **Each** using a different filter and resulting in a distinct feature map.

- We then **stack** all these feature maps together.
  - To create the **final output** of the **convolution layer**.

AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Convolution(al) Operations & Final Outputs

- Convolution operations result in a set of feature maps.
- All these feature maps are stacked together to create the **final output** of the **convolution layer**.

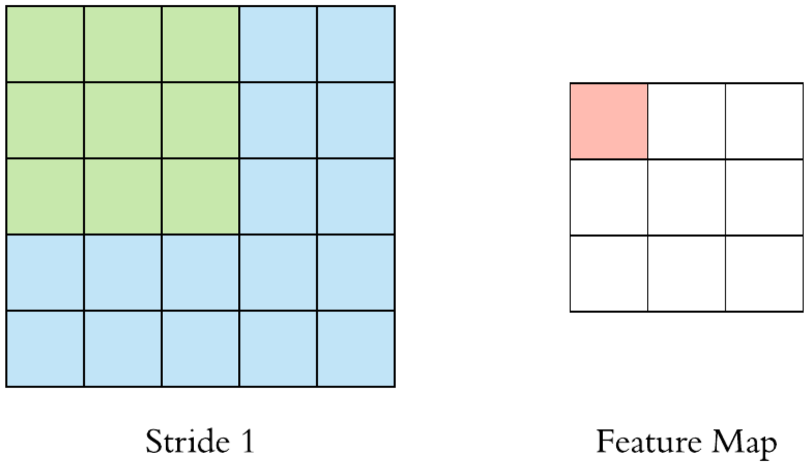


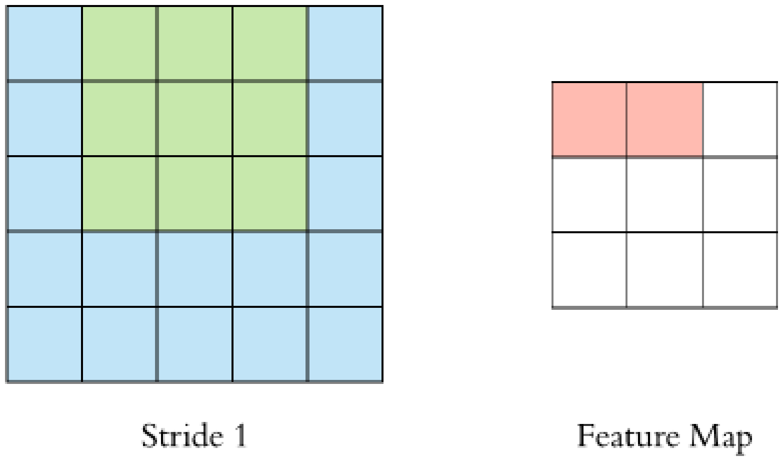
AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Convolution(al) Operations: Stride

- In convolution operations, **stride** specifies **how much** we move the convolution filter **at each step**.

- By **default**, the value is **1**, as you can see in the figure below.



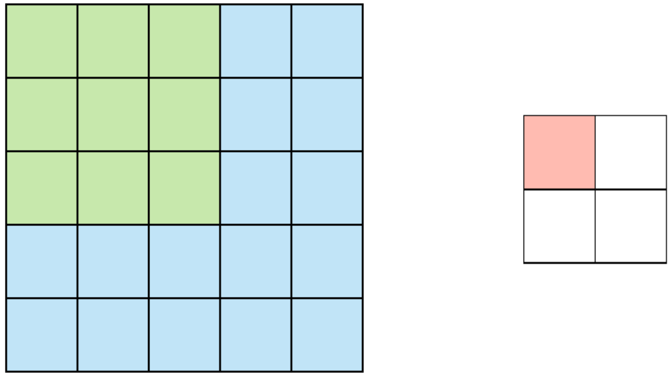


AI Deep Learning: Convolutional Neural Networks (CNN)

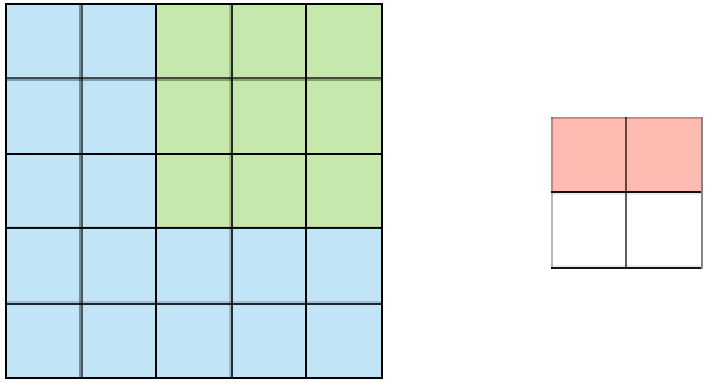
CNN: Convolution(al) Operations: Stride

- It is possible to use **bigger strides**
  - That makes the resulting feature map **smaller**
    - It’s likely to skip over potential locations.
  - That results in less overlap between the receptive fields.

- The following figure demonstrates a stride of 2.



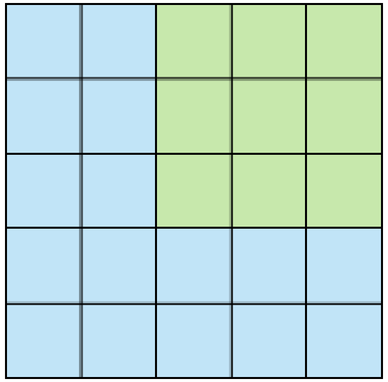
Stride 2 Feature Map



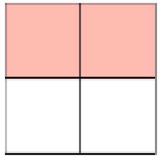
Stride 2 Feature Map

AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Convolution(al) Operations: Padding



Stride 2



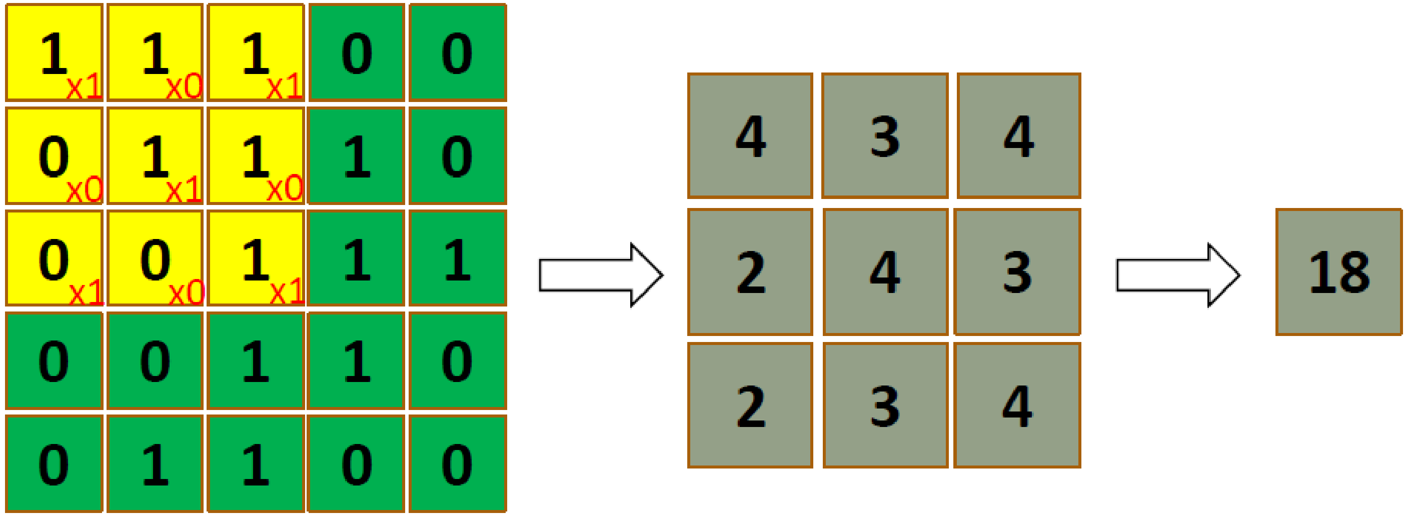
Feature Map

- So, the **size** of the feature map is **smaller** than the input:
  - Because the convolution filter needs to be contained in the input.
- It’s GOOD!
  - Convolution operations reduce dimensionality.
  - Fewer dimensions, simpler & faster computation.

AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Convolution(al) Operations: Padding

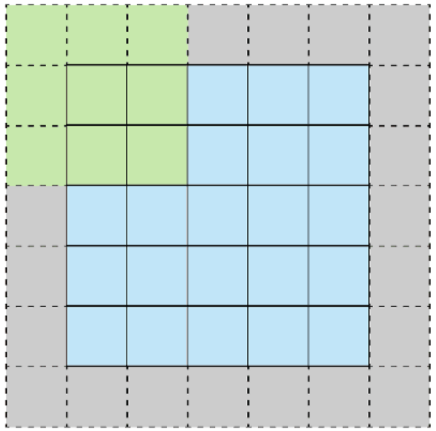
- BUT!
  - It’s also a big problem. What?
  - The image inputs get smaller and smaller
    - → Details are lost → recognition accuracy drops
    - → Run out of “latent pixels”
    - → Convolutional neural networks: not too deep architectures



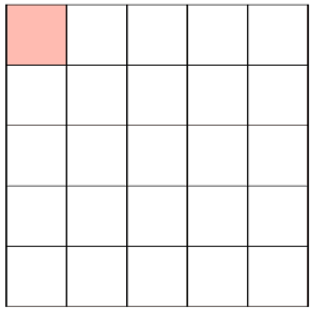
- If we want to **maintain** the **same dimensionality:**
  - Use **padding** to surround the input with zeros.

AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Convolution(al) Operations: Padding



Stride 1 with Padding



Feature Map

- The gray area around the input is the **padding**.
- We either **pad** with zeros or the values on the edge.
  - Padding makes the dimensionality of the feature map **match** the input.
- Padding is commonly used in convolutional neural networks to preserve the size of the feature maps.
  - Otherwise, the size of feature maps would shrink at each layer, which is not desirable.

AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Convolution(al) Operations: Zero Padding

- HOWTO calculate how many layers of values for padding?
  - Given **h_filter**: height of the filter; **w_filter**: width of the filter
  - Number of layers of values for padding = (h_filter – 1) / 2 OR (w_filter – 1) / 2
  - For example, with h_filter = 3 and w_filter = 3, layer of padding = 1

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

AI Deep Learning: Convolutional Neural Networks (CNN)

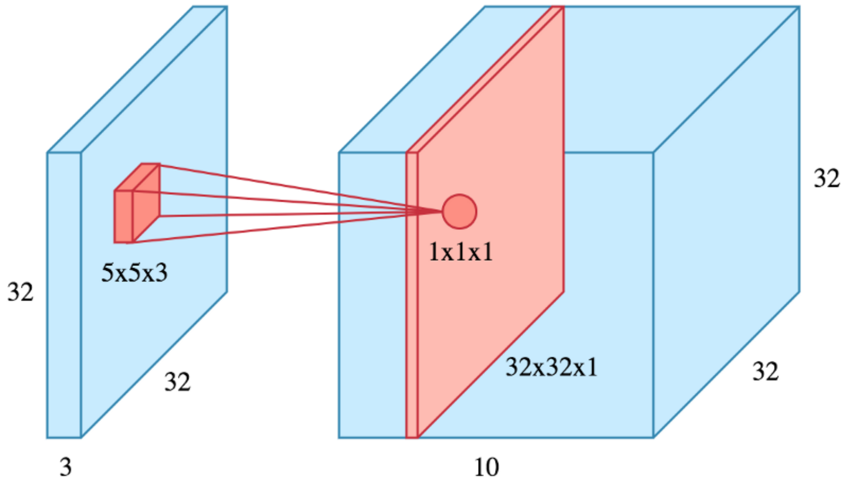
CNN: Convolution(al) Operations: Suggested Best Practices

- Image size:
  - Resize the image to have a size in the **power of 2**, e.g., 32x32, 256x256
- Convolution filter:
  - A filter of (h, w) = **3 x 3** should work well with deep neural networks
- Stride:
  - Use the default stride: **s = 1**
- Padding:
  - Add **1 layer** of zero padding

AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Convolution(al) Operations: An Example of 3D Filters

- First, to be simple, a convolution using a single filter.

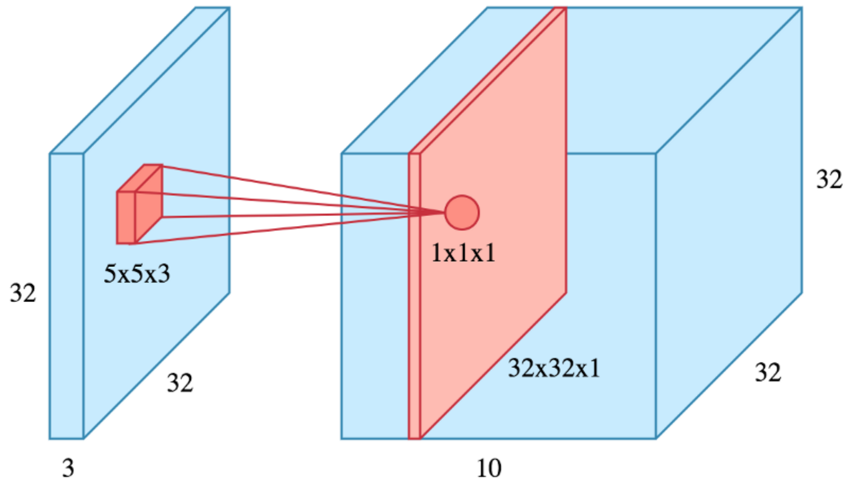


- Given a 32x32x3 image and a filter of size 5x5x3
  - NOTES: *the depth of the convolution filter matches the depth of the image, both being 3*.
- When the filter is at a particular location, it covers a small volume of the input, and the convolution operation is performed as above.
- In this convolution operation, the sum of matrix multiply in 3D – instead of 2D – is done; the result is still a scalar.
- Slide the filter over the input as above and perform the convolution at every location aggregating the result in a feature map.
- This feature map is of size 32x32x1, shown as the red slice on the right.

AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Convolution(al) Operations: An Example of 3D Filters

- If we used 10 different filters we would have 10 feature maps of size 32x32x1:
  - Stacking them along the depth dimension produces the final output of the convolution layer:
    - a volume of size 32x32x10, shown as the large blue box on the right.

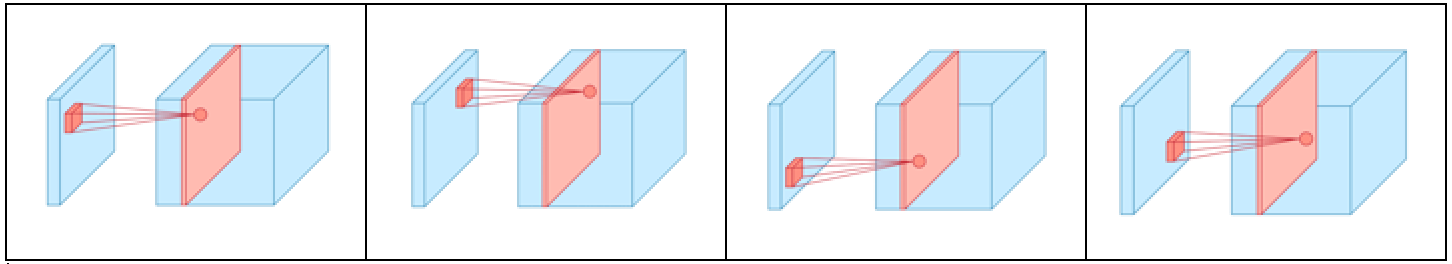


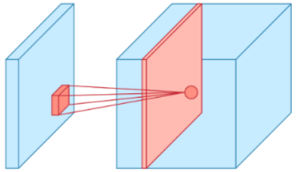
CNN: Convolution(al) Operations: An Example of 3D Filters

- NOTES:
  - *The height and width of the feature map are unchanged and still 32 due to padding.*

- To help with visualization:
  - Slide the filter over the input as shown in the simple scenario of the 2D filter.
  - At each location, a scalar is calculated.
  - The set of all these scalars make the feature map.

- The figures show the sliding operation at 4 locations.
  - However, in reality, the convolution operation is performed over the entire input, i.e., all over the blue rectangle to the left.

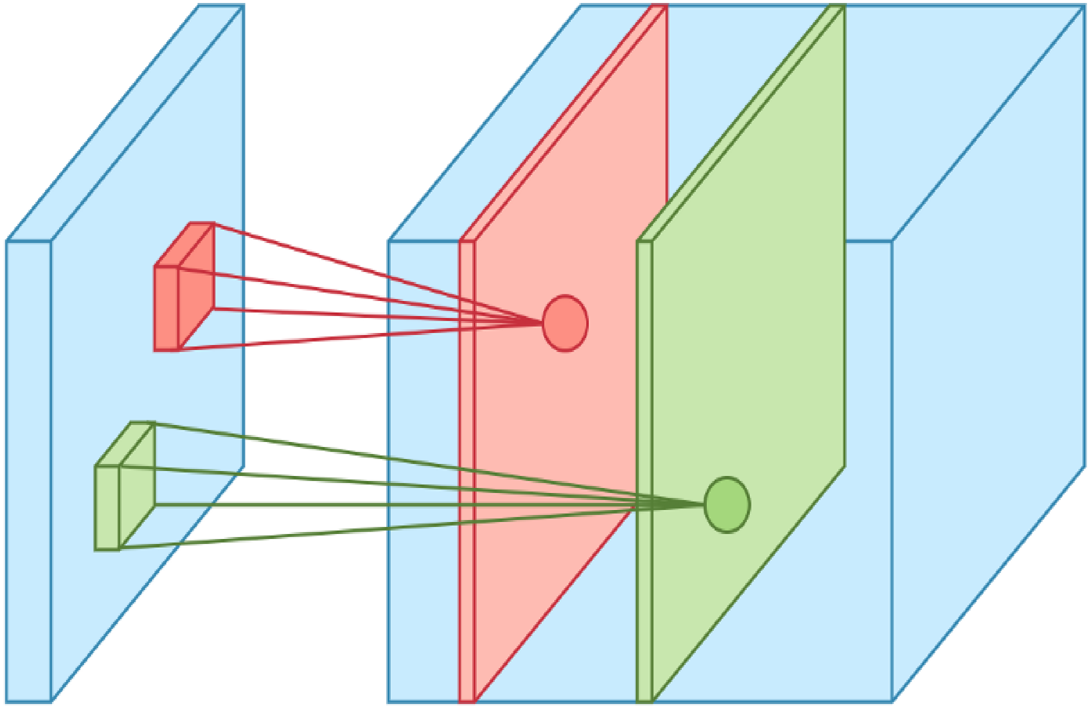




AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Convolution(al) Operations: An Example of 3D Filters

- Let' see how two feature maps are stacked along the depth dimension:
    - The convolution operation for each filter is performed independently.
    - The resulting feature maps are separated.



AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Pooling

- After a convolution operation we usually perform **pooling** to reduce the dimensionality.
  - This enables us to reduce the number of parameters
  - That helps both shorten the **training time** and combat **overfitting**.

- **Pooling layers** down-sample each feature map independently.
  - That **reduces** the height and width, **keeping** the depth intact.

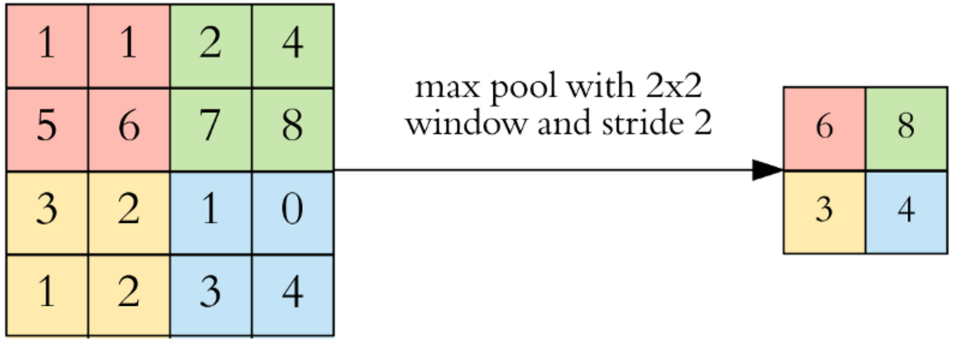
- The most **common** type of pooling is **max pooling**.
  - That just takes the **max value** in the pooling window.

- Different from the convolution operation, pooling has **no parameters**.
  - It slides a window over its input, and simply takes the max or the average value in the window.
  - Similar to a convolution, we specify the **window size** and **stride**.

AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Pooling: Why?

- Pooling aggregates multiple values into a single value.
- Pooling: Invariance to small transformations
- Pooling reduces the size of the layer output/input to next layer.
  - So, faster computations
- Pooling keeps most important information for the next layer



AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Pooling: Max Pooling vs. Average Pooling

- Max pooling:
  - Keep the max value in the window

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

4	9
5	7

- Average pooling:
  - Calculate the average of all the values in the window and keep it.

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

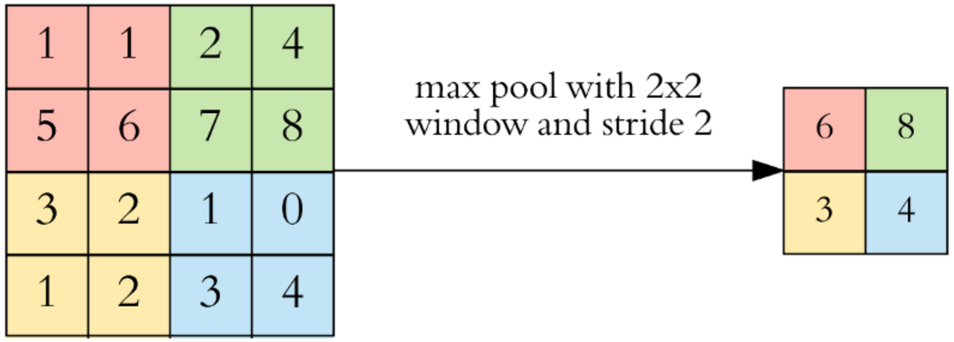
2.5	4
2	2.5

AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Pooling: An Example with Max Pooling

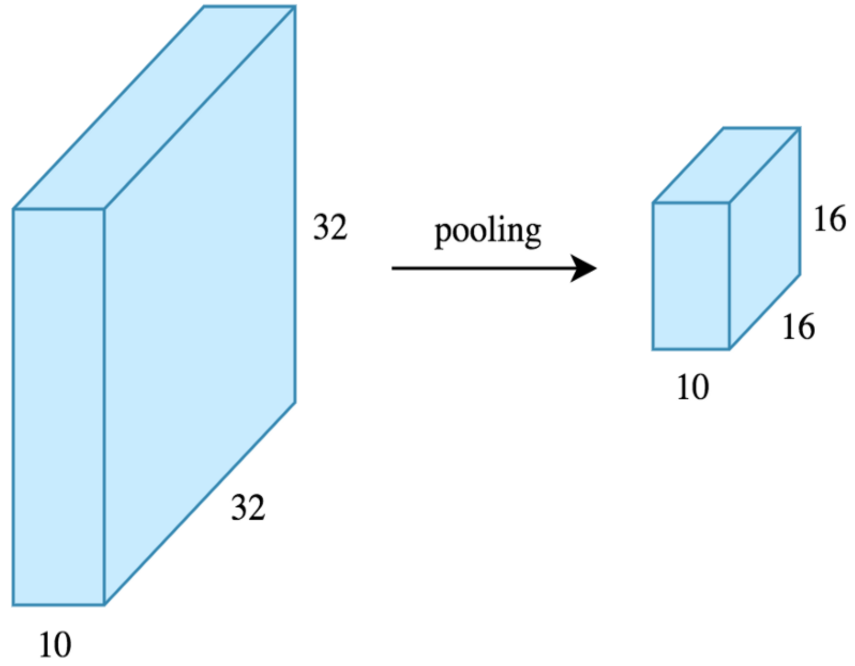
- Here is the result of max pooling using a 2x2 window and stride 2
    - Each color denotes a different window.
    - Since both the window size and stride are 2, the windows are not overlapping.

- This window and stride configuration **halves the size of the feature map**.
    - This is the **main use case of pooling**, down-sampling the feature map while keeping the important information.



AI Deep Learning: Convolutional Neural Networks (CNN)

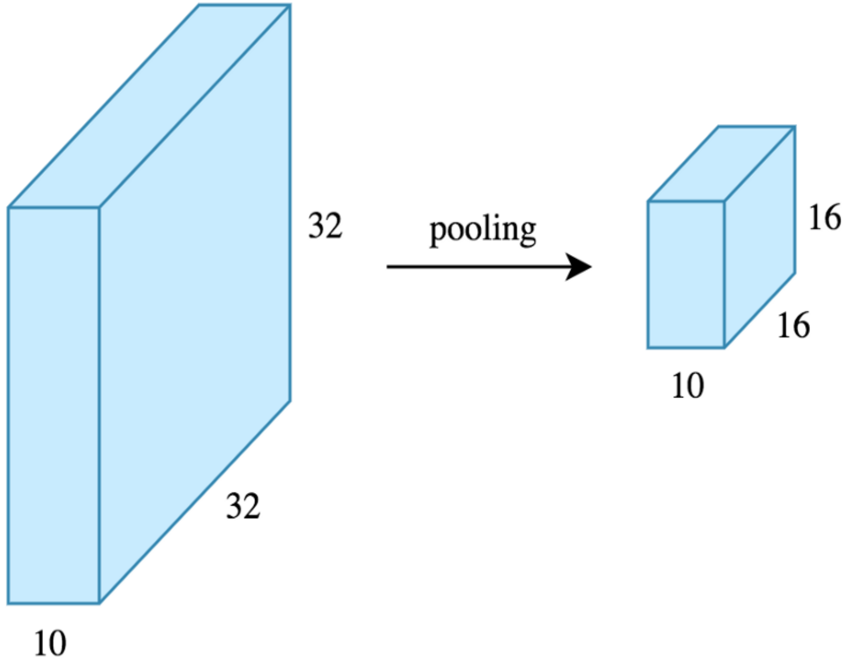
CNN: Pooling



- Consider the **feature map dimensions** before and after pooling:
  - If the input to the pooling layer has the dimensionality 32x32x10, using the same pooling parameters described above, the result will be a 16x16x10 feature map.
  - Both the height and width of the feature map are **halved**, but the **depth** is **not** changed because pooling works independently on each depth slice the input.

AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Pooling



- By **halving** the height and the width, we reduced the number of weights to **1/4** of the input.

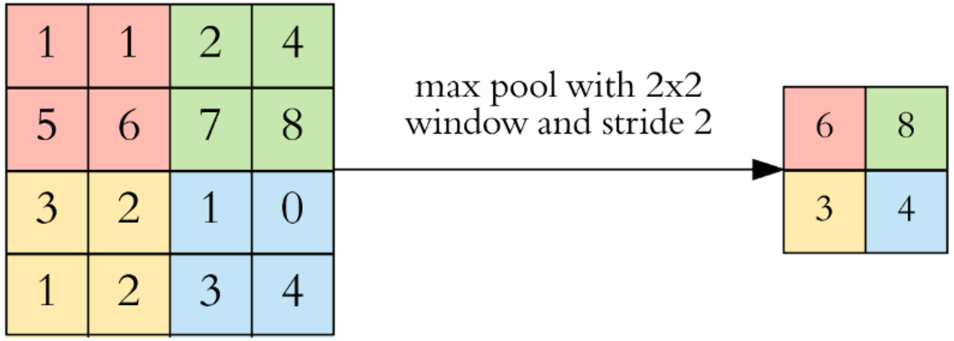
- Consider that to train a deep learning model, it is typical to deal with millions of weights in CNN architectures, this reduction is a big deal.

- In convolutional neural network architectures:
  - Convolution is done with 3x3 windows, stride 1 and with padding.
  - Pooling is typically performed with 2x2 windows, stride 2 and no padding.

AI Deep Learning: Convolutional Neural Networks (CNN)

CNN: Pooling

- Here is the result of max pooling using a 2x2 window and stride 2
  - Each color denotes a different window.
  - Since both the window size and stride are 2, the windows are not overlapping.



- This window and stride configuration **halves the size of the feature map**.
  - This is the main use case of pooling, down-sampling the feature map while keeping the important information.