# CS167: Machine Learning

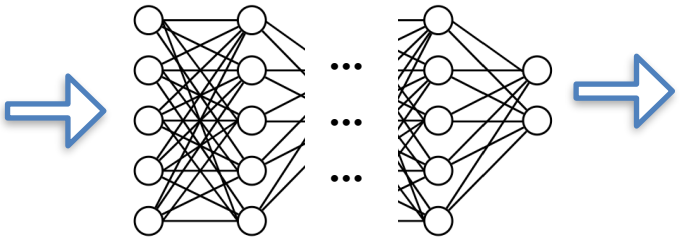
## Convolutional Neural Network (CNN)

Wednesday, April 22nd, 2026

Drake UNIVERSITY

# What's Next?

- A **multilayer perceptron** (MLP) is the simplest type of neural network. It consists of perceptrons (aka nodes, neurons) arranged in layers



- A **multilayer perceptron** (MLP) is just the tip of the iceberg; plenty of other neural network variants exist.

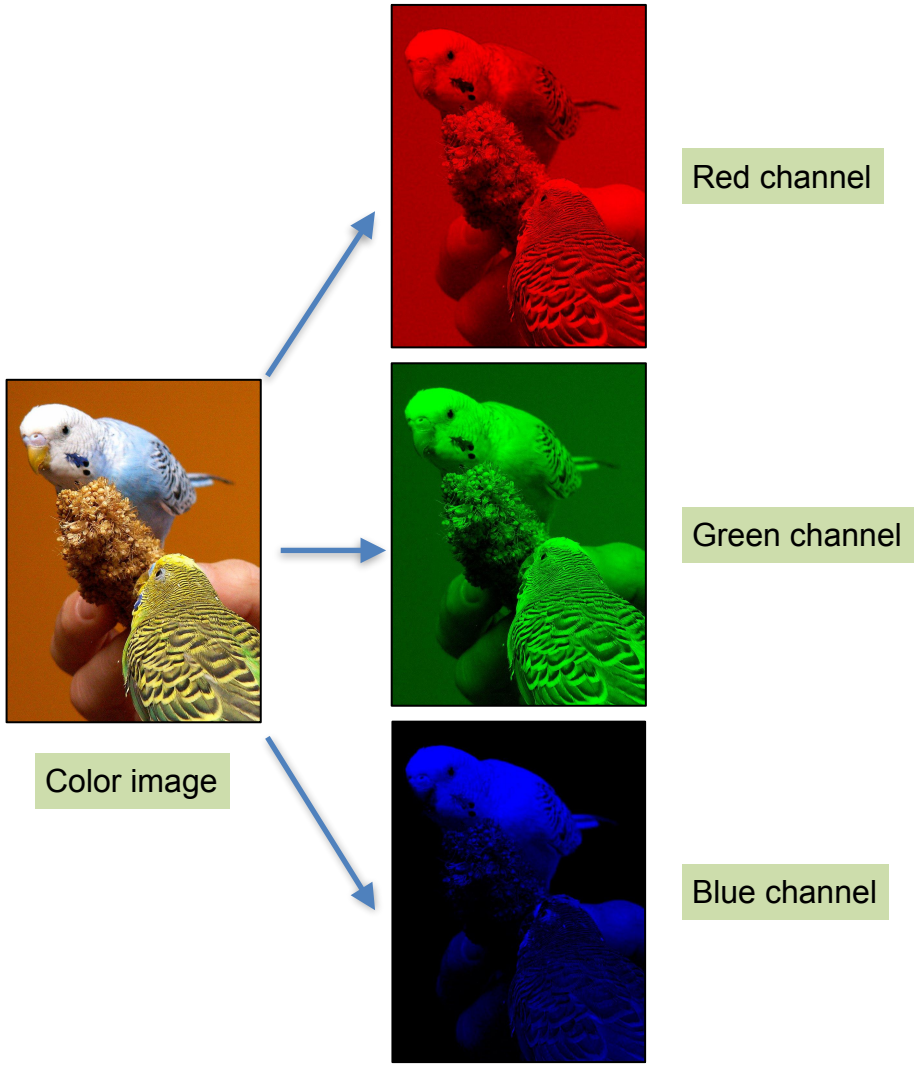
# Today's Agenda

- Convolutional Neural Network (CNN): another type of neural network
  - Convolution operation
  - Nonlinearity
  - Pooling operation
  - CNN: convolutional layer + nonlinearity + pooling layer

# Convolutional Neural Network (CNN)

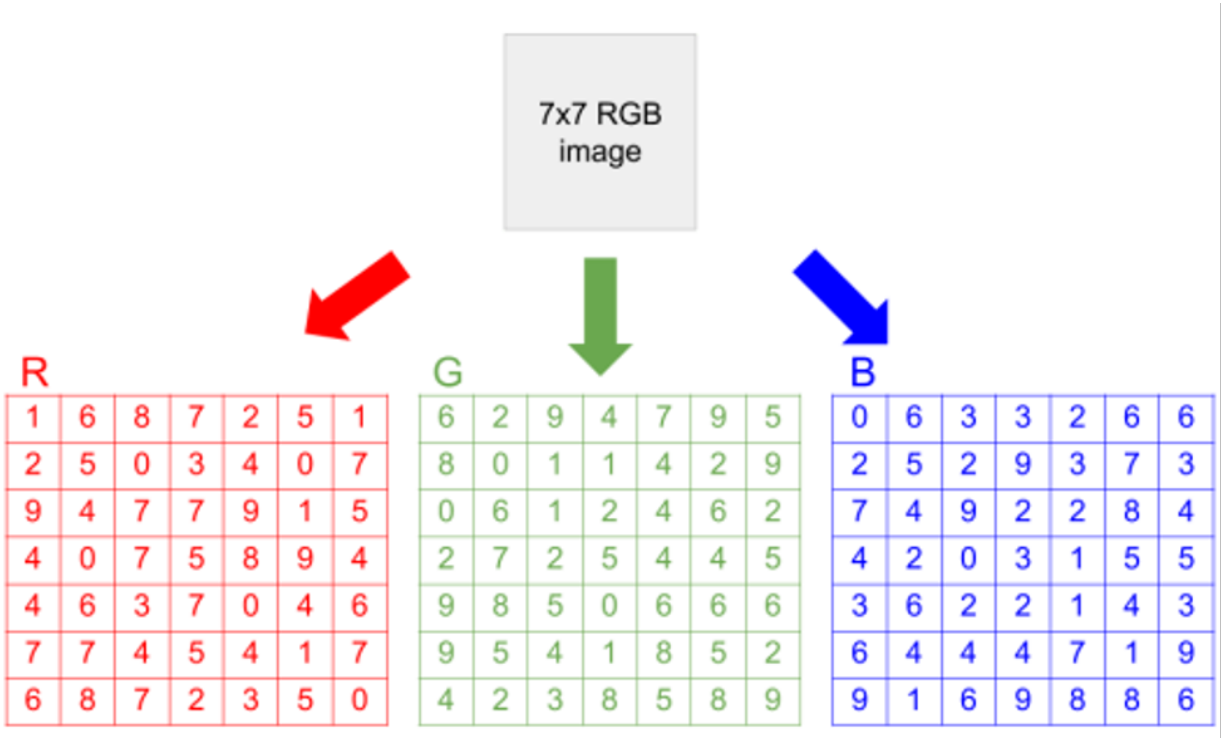
- A **convolutional neural network** that applies **convolutional filters** on grid-like input such as a image

- Image data is represented as a two-dimensional grid of pixels, either grayscale (monochromatic) or color (RBG)
  - each pixel corresponds to one or multiple numeric values: if it's grayscale, it is one number, if it's color, it corresponds to 3 numbers (a red, a green and a blue value)



# Convolutional Neural Network (CNN)

- A **convolutional neural network** that applies **convolutional filters** on grid-like input such as a image



# Convolutional Neural Network (CNN)

- A **convolutional neural network** that applies **convolutional filters** on grid-like input such as a image

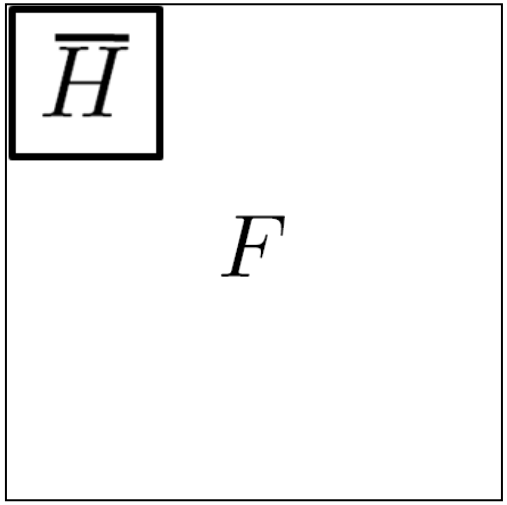

- In order to capture the local dependence of images, we use **convolutional filters.** A convolutional filter, aka kernel:
  - is smaller than the input data (usually 3x3 or 5x5 or 7x7)
  - uses dot product multiplication between a piece of the input that is the size of the filter and the filter
  - scans over the image from the upper left to the bottom right

# Convolution Operation

- What does a **convolution operation** do?

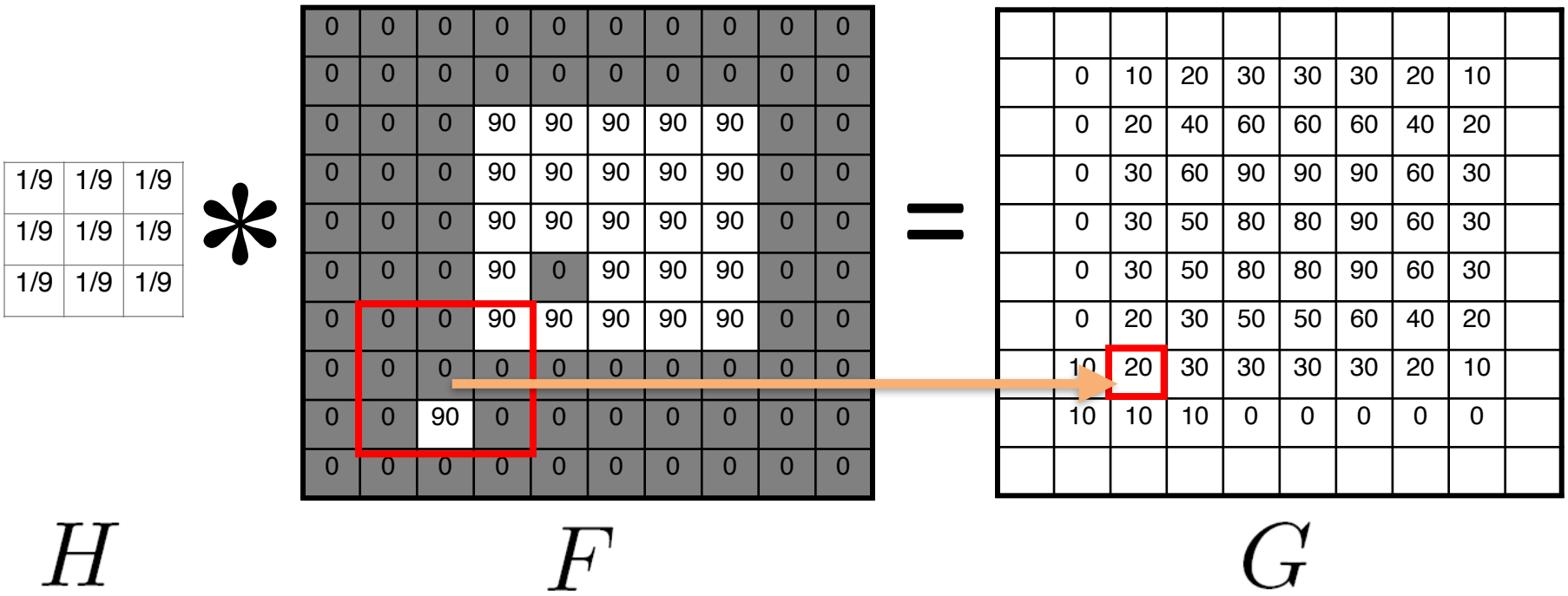
- In an ideal **convolution operation,** a kernel is “flipped” (horizontally and vertically) and then it is applied through the image (from left to right, and top to bottom)


kernel of size
3x3 or 5x5



# Convolution Operation

- What does a **convolution operation** do?



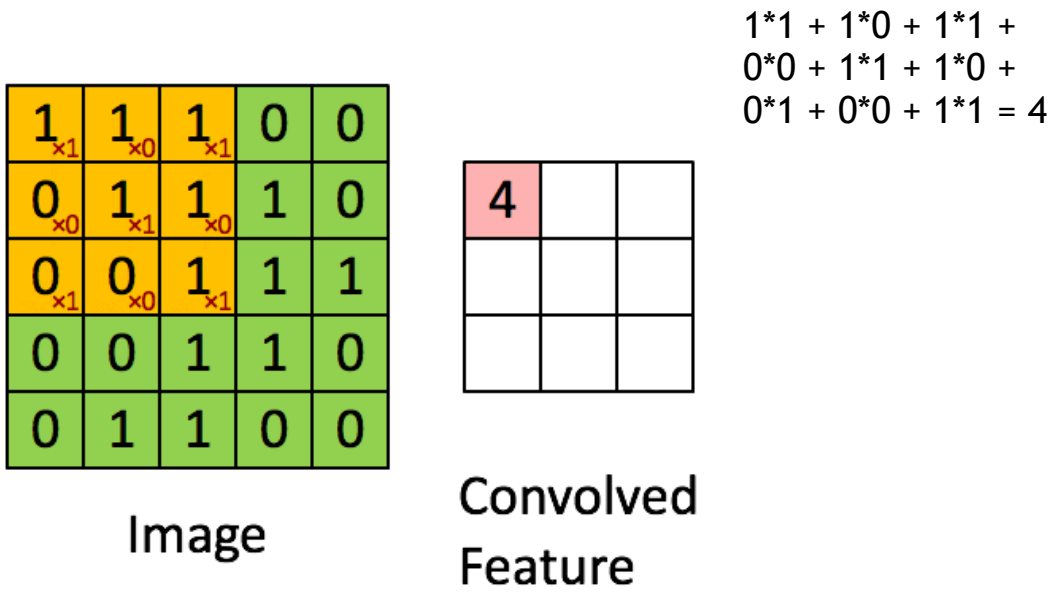

kernel of size 3x3 units

image of 10x10 units

convolved result of 10x10 units

# Convolution Operation

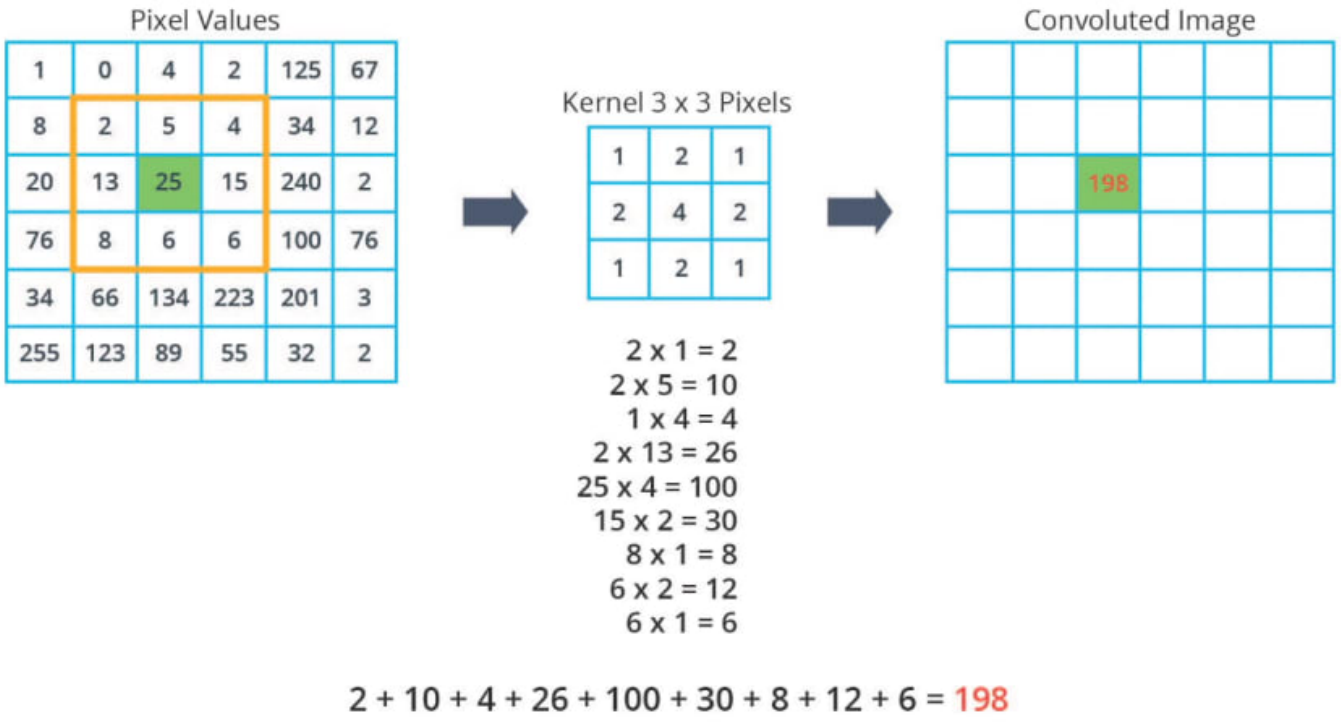
- What does a **convolution operation** do?
- **convolution operation can be achieved with a series of dot products between portions of input feature map and a convolution filter (kernel) weights**




Another visualization shows a yellow convolution filter applied to a green image, resulting in the convolved feature

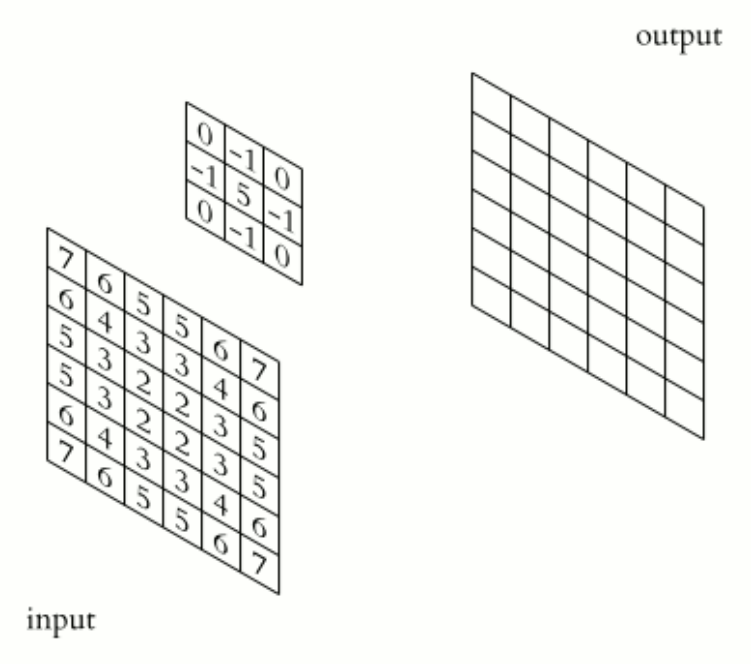
# Convolution Operation

- What does a **convolution operation** do?
- **convolution operation can be achieved with a series of dot products between portions of input feature map and a convolution filter (kernel) weights**



# Convolution Operation Animation!

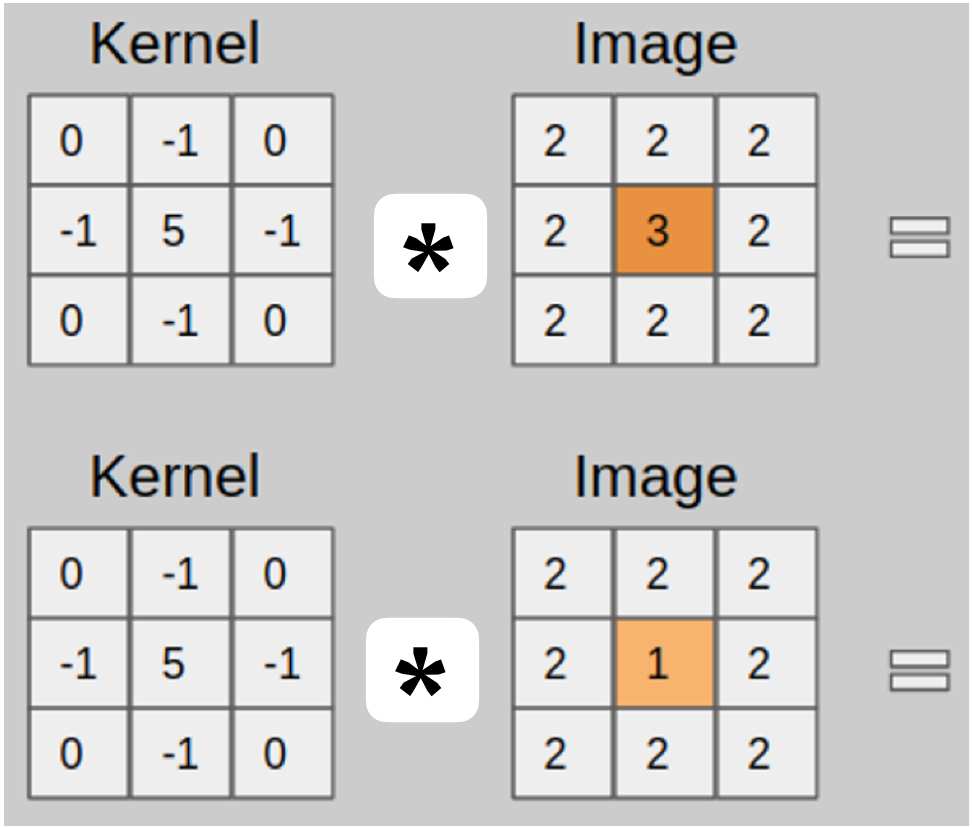
- What does a **convolution operation** do?
- **convolution operation can be achieved with a series of dot products between portions of input feature map and a convolution filter (kernel) weights**



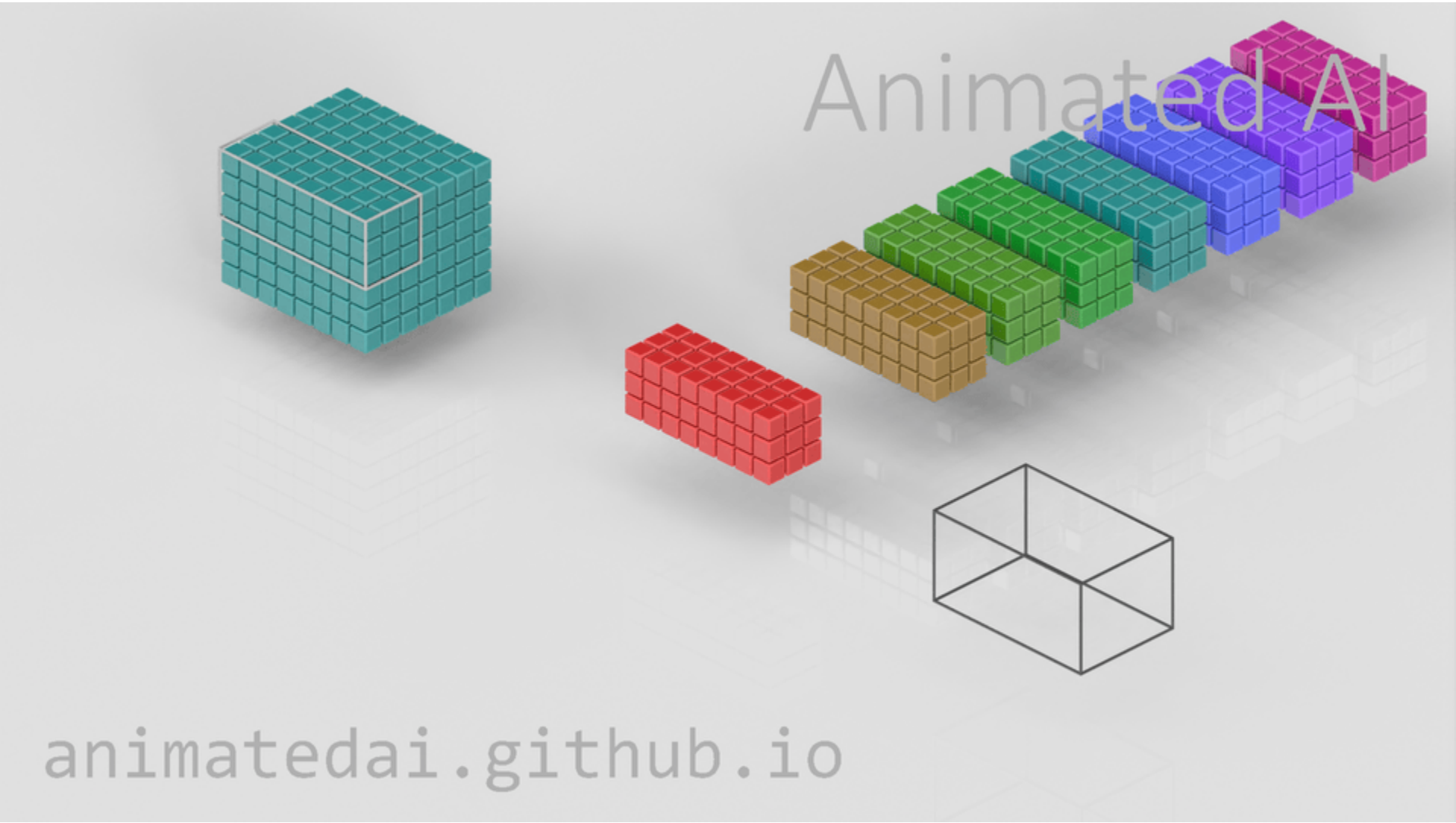

Another visualization shows a convolution filter applied to an image, resulting in the convolved feature

# Group Exercise

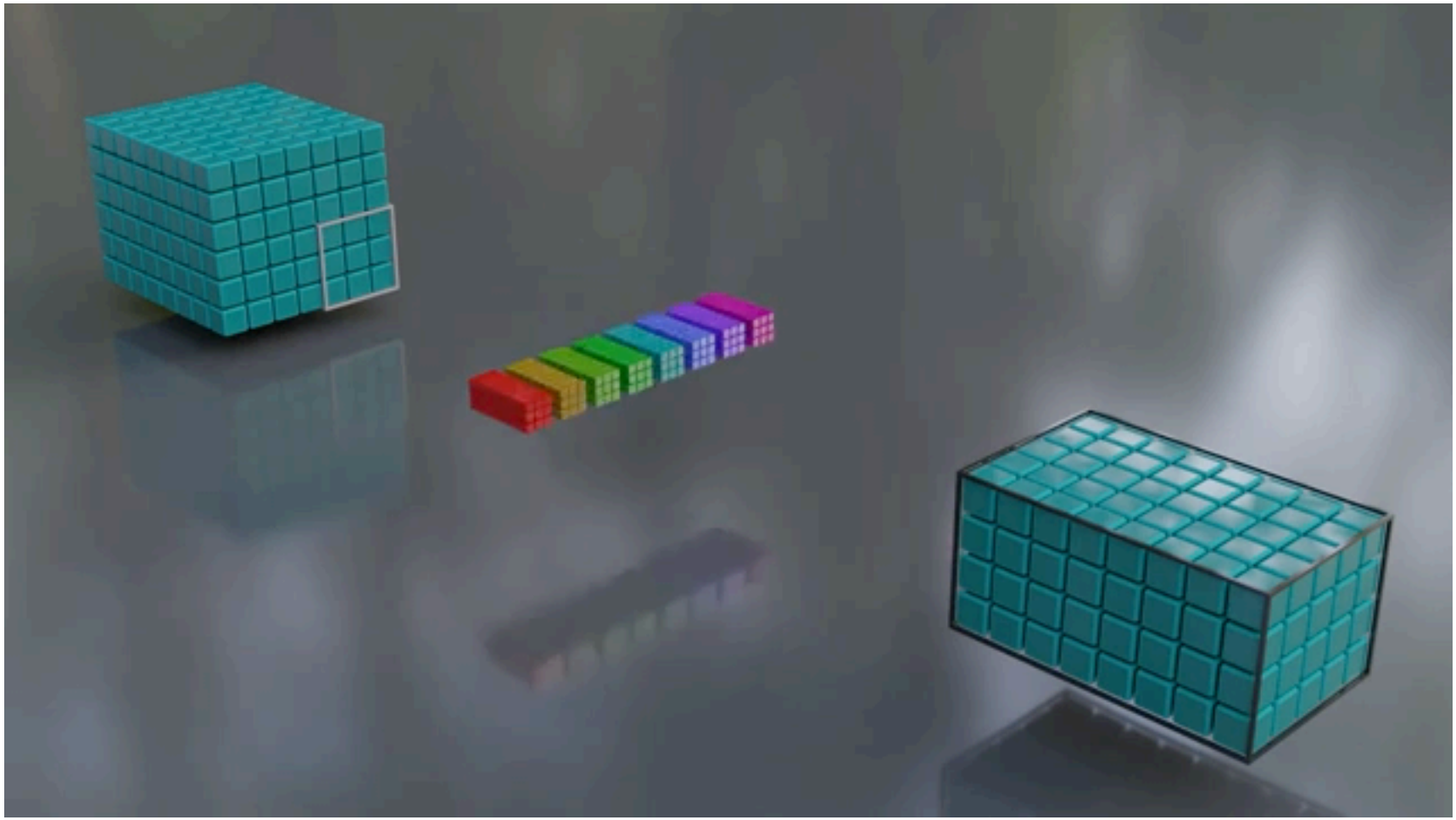
- find the result of the **convolution operation** below



# How to calculate the output volume size?



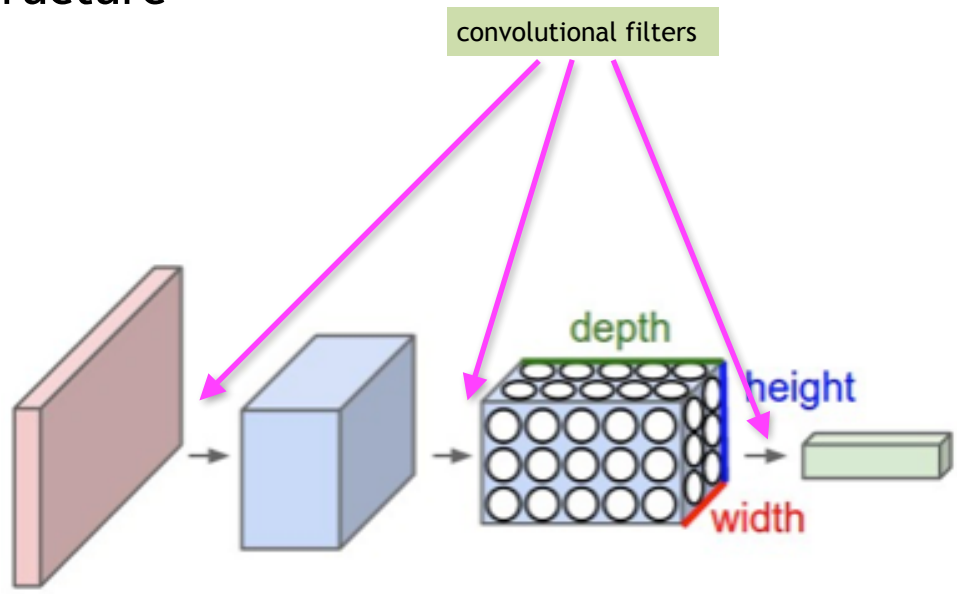
# How to calculate the output volume size?



https://www.youtube.com/watch?v=w4kNHKcBGzA&t=210s

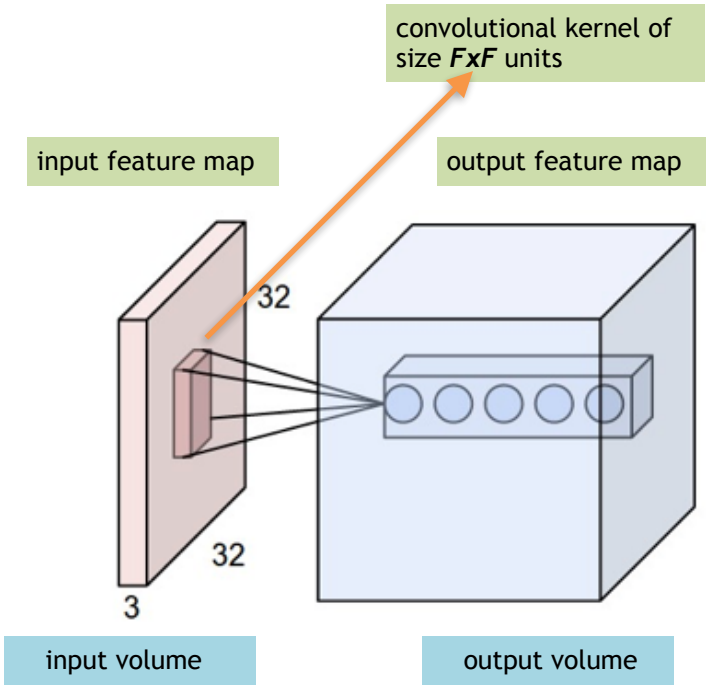
# Convolutional Neural Network (CNN)

- A **convolutional neural network (CNN)** is a neural network with specialized connectivity structure



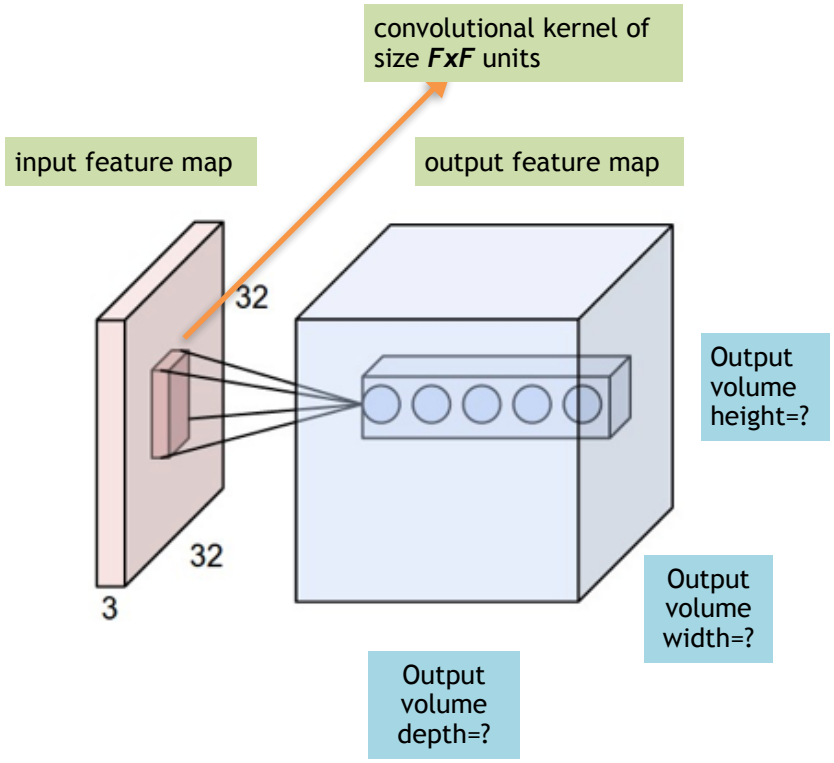

- Every layer of a CNN transforms the input volume to an output volume of neuron activations. The red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels)

# Convolutional Neural Network (CNN)



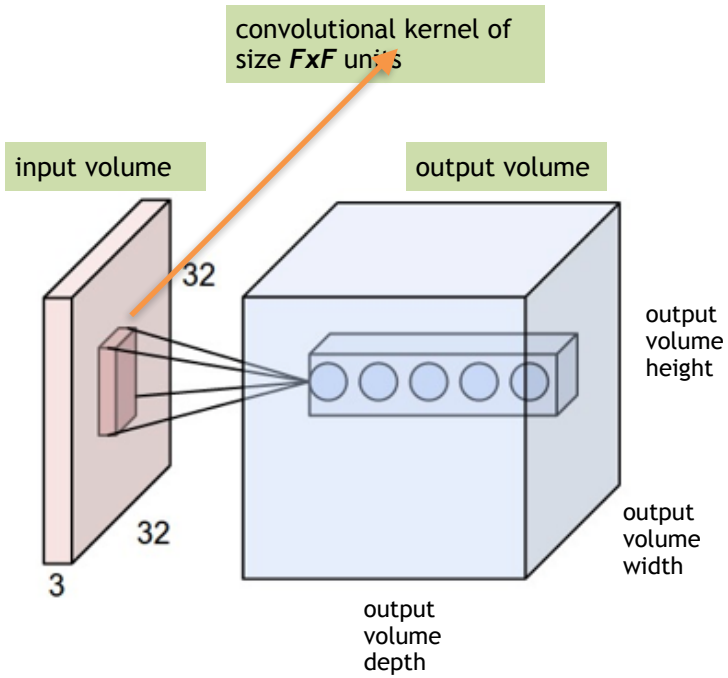

- Weights correspond to the filter (kernel) values
- **Convolutional neural network** can **learn** their own filters!
  - We do not need to provide the values inside the kernel

# CNN: How to calculate the output volume size?



# CNN: How to calculate the output volume size?

- An input volume has size ***(WxWx3)***, eg, (227, 227, 3)
- Filter size/receptive field is ***(FxF)***, eg, (11x11)
- Spatial Stride ***S***, eg, ***S***=4
- Padding size ***P***, eg, ***P***=0
- Number of filters ***K***, eg, ***K***=96

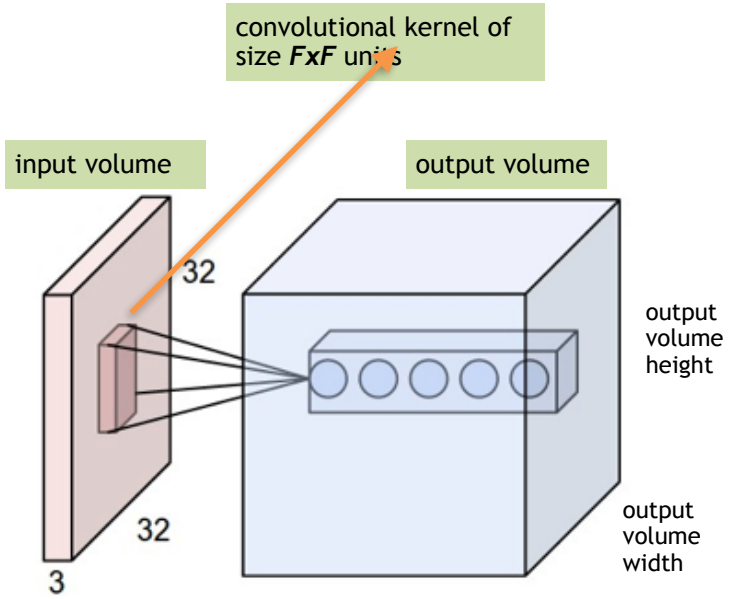



- Size of the output volume width and output volume height as a function of ***W***, ***F***, ***S***,and ***P*** as follows:

$$\text{output volume width/height} = \frac{(W - F + 2P)}{S} + 1 = \frac{(227 - 11 + 2*0)}{4} + 1 = 54+1 = 55$$

# How to calculate the output volume size?

- An input volume has size **($W_1$ x $H_1$ x $D_1$)**
  - Filter size/receptive field is ***(FxF)***
  - Spatial stride size ***S***
  - Padding size ***P***
  - Number of filters ***K***




- Spatial sizes of the output volume ***($W_2$ x $H_2$ x $D_2$)***

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

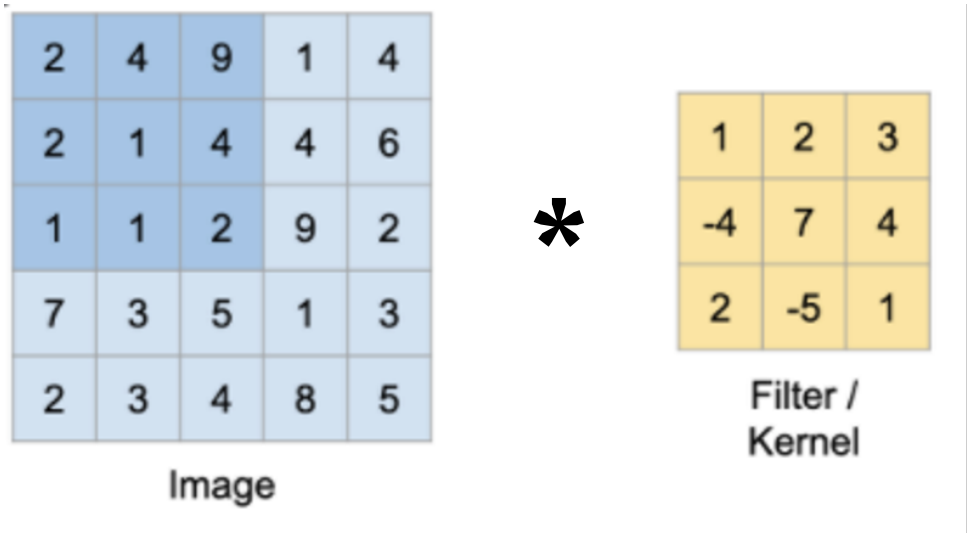
$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

$$D_2 = K$$

- Number of filter weight parameters **= *(F x F x $D_1$) x K***
- Number of bias parameters = ***K***

# Group Exercise

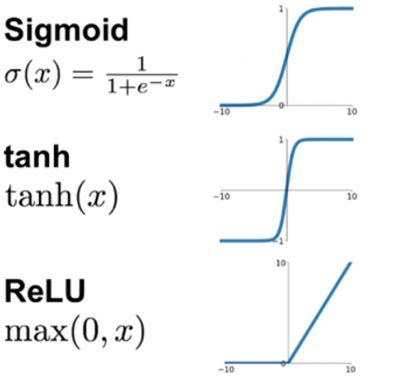
- What will the size of the output of the following convolution be?
  - (5x5x1) * (3x3)



# Today's Agenda

- Convolutional Neural Network (CNN): another type of neural network
  - Convolution operation
  - Nonlinearity
  - Pooling operation
  - CNN: convolutional layer + nonlinearity + pooling layer

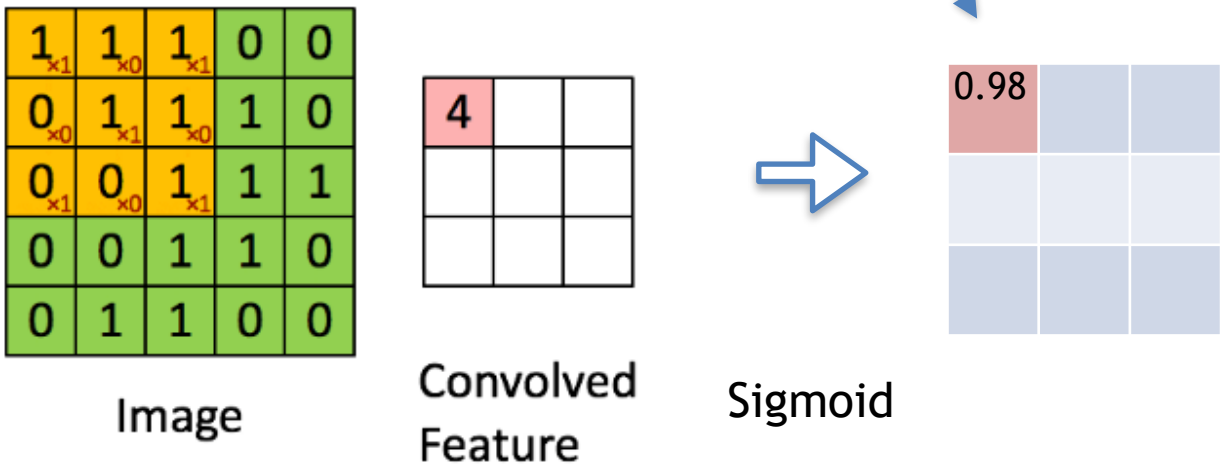
# Nonlinear Function




- Just like an MLP, each convolutional output goes through a non-linear function such as Sigmoid, Tanh, or Rectified Linear Unit (ReLU)

$$convolution = 1*1+1*0+1*1+0*0+1*1+1*0+0*1+0*0+1*1 = 4$$

$$Sigmoid(4) = \frac{1}{1+exp(-4)} = 0.98$$



# Today's Agenda

- Deep Learning

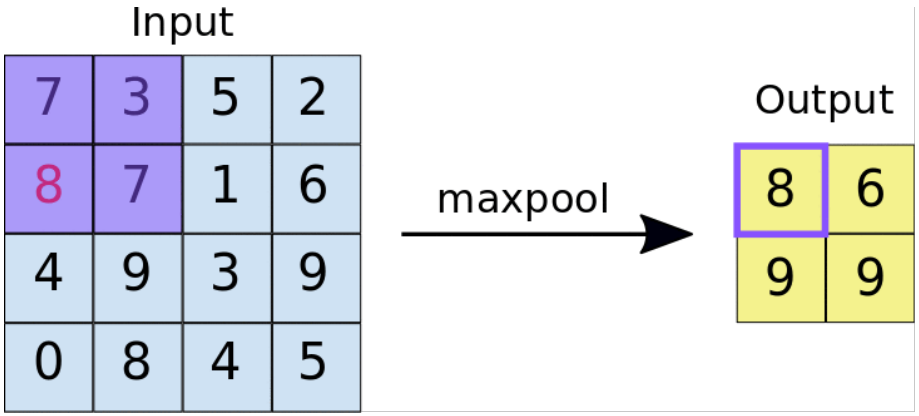
- Convolutional Neural Network (CNN)
  - Convolution operation
  - Nonlinearity
  - Pooling operation
  - CNN: convolutional layer + nonlinearity + pooling layer

# Pooling Operation

- Image data can get computationally inefficient, really quickly. To avoid this, we often toss in a layer that helps us to **summarize** and **downsample** the data

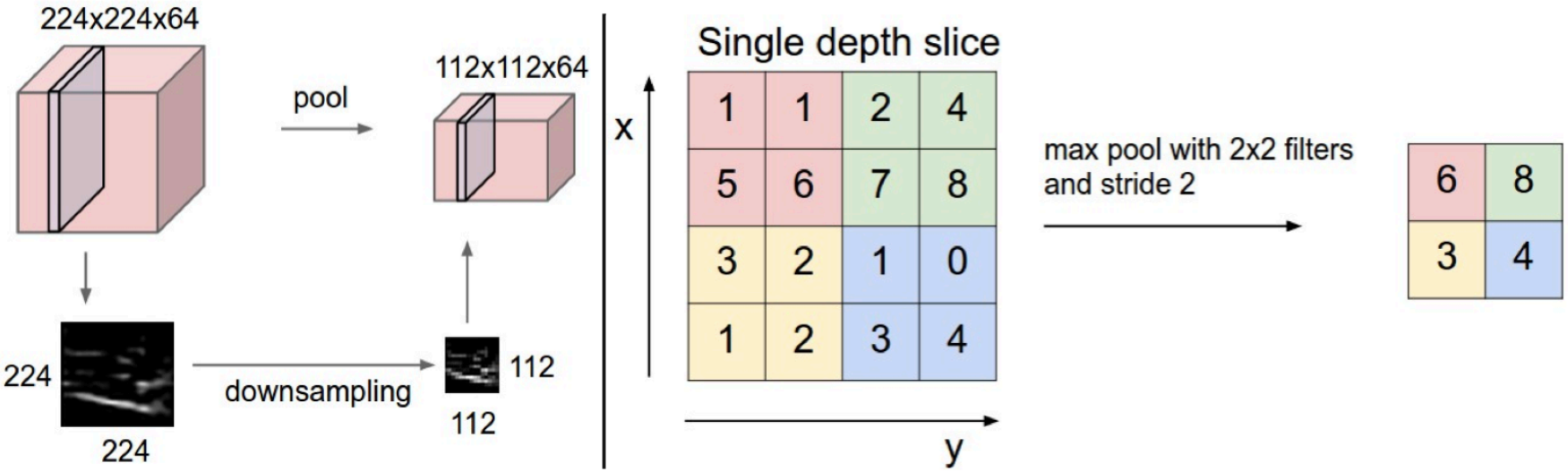
- In classical CNN, we find another useful operation called **pooling operation**

- A common pooling operation is **max pooling**, and its goal is to locally summarize the convolution. It performs something like a convolution, but rather than taking the dot product, it takes the maximum element in the filter area



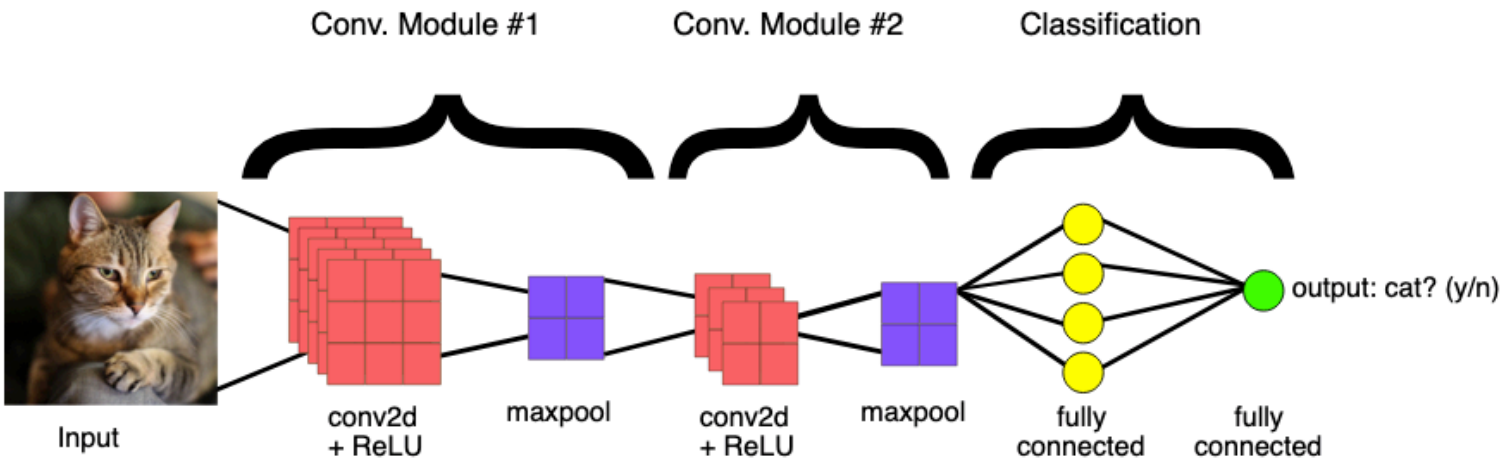
# Pooling Operation

- Pooling operation downsamples the volume spatially, i**ndependently in each depth slice of the input volume**
- Besides max pooling, other pooling operations include: **sum pooling**, average pooling



# CNN: A Composition of Convolutional Layers

- We've talked about **image data**, **convolutions**, **nonlinearity**, **max pooling**, and how they are related to some computer vision tasks. Let's connect the dots
  - input is an image (in this case a color image, so 3 channels—red, green, and blue)
  - there are several filters, not just one.
  - `Conv2D` layers with `ReLU` are often followed by `maxpool`
  - towards the end of the model, we switch to fully connected (`Dense`) layer
  - We have as many output nodes as we have classes to predict




Reference