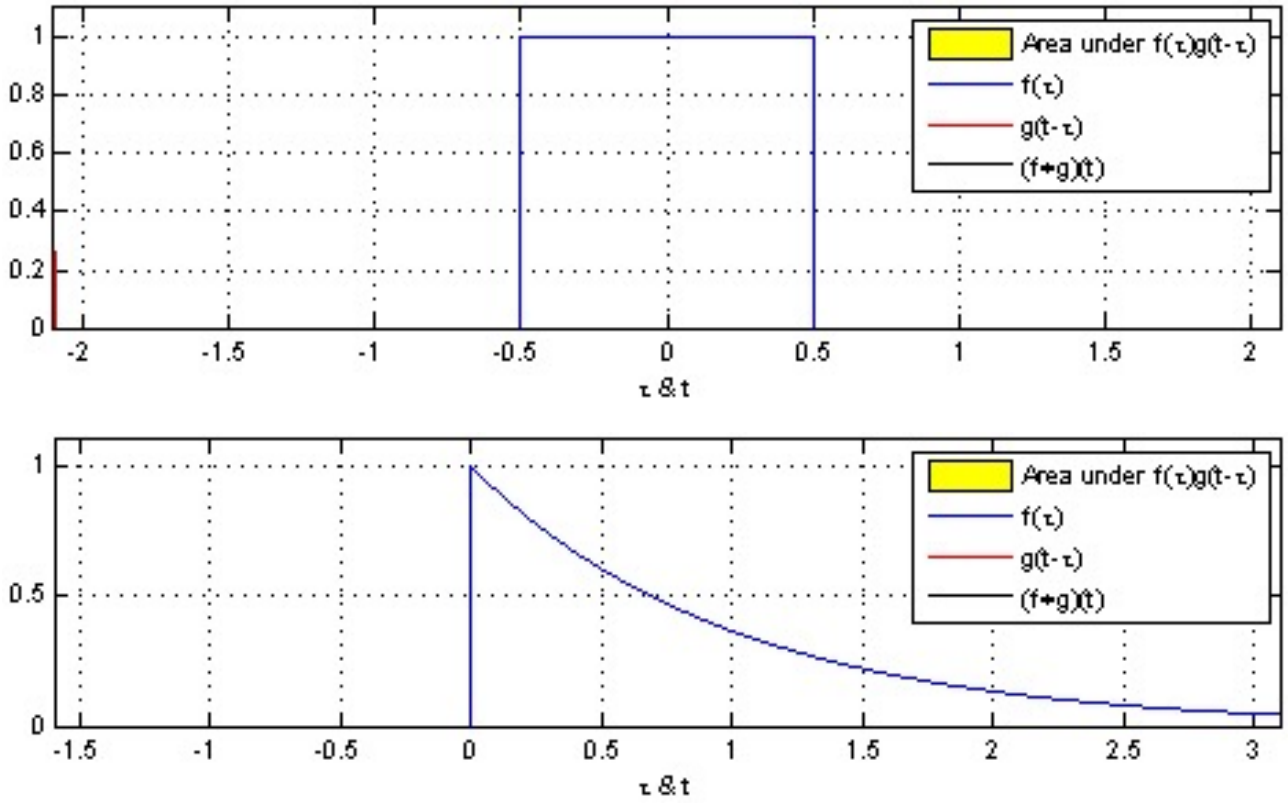
Convolution

What Is a Convolution?

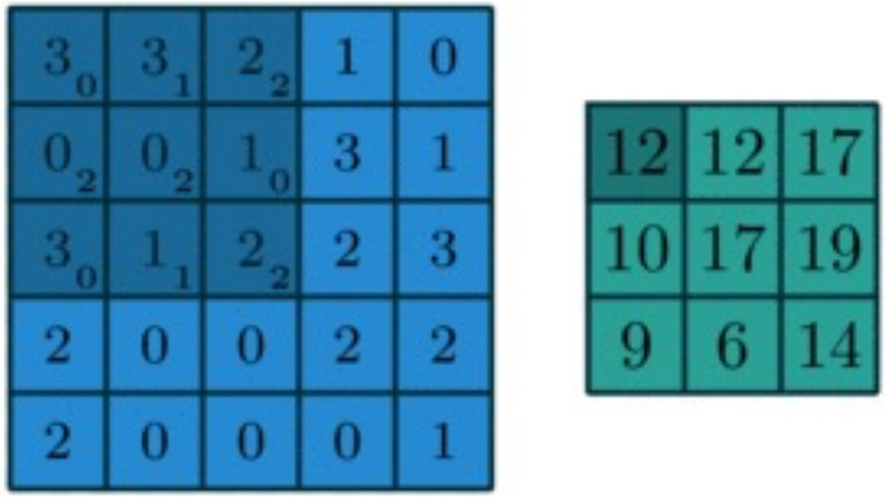
A convolution is the overlap of two functions. We slide one function, called the filter, over another. The output is the area of overlap between the two functions.



Two visual examples of convolutions. Images courtesy of Wikipedia.

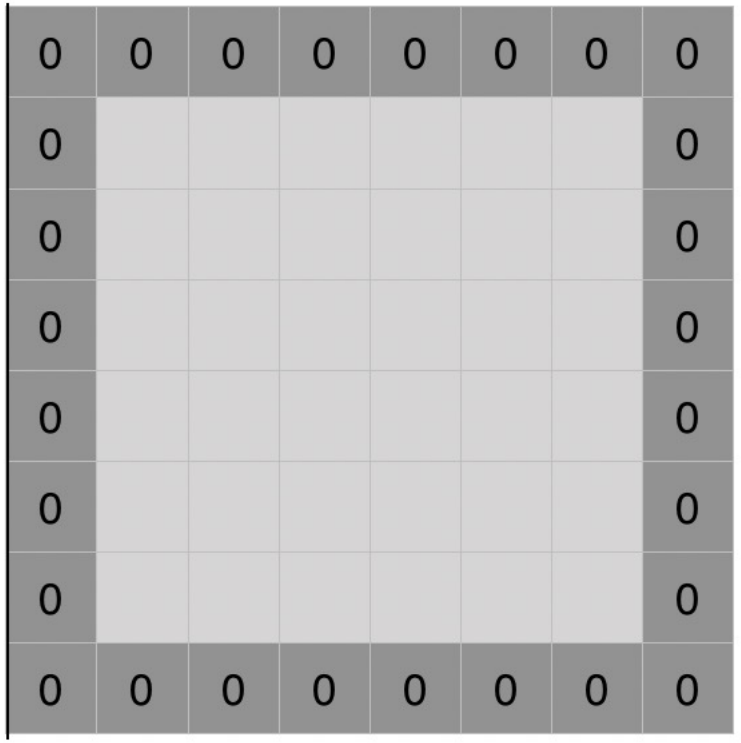
Convolutions in Neural Networks

- We will slide our filter (numbers located in the lower right) over the data (typically an image) to produce a new “feature.” We will use multiple filters to create new features. In a convolution neural network, what is being learned is the filter.
- The size of the filter (here 3x3) is something to “tune” in your architecture.



Padding

- Notice that the output of the previous slide was smaller by 1 in each dimension. In order to get the output the exact same size, we employ “padding.”
- Padding adds an extra “0” along the border so that the output size matches the input size.
- Padding also prevents edge distortion.



Zero-padding added to image

Strides

- Strides are how far the filter moves each step.
- In the example shown on the two previous slides, the stride is 1.
- Stride is typically small, but is also a tunable parameter.

Example Convolutions

Original

Averaging

$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Original

Edge Detection

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

DataScience@SMU

Convolutional Neural Networks

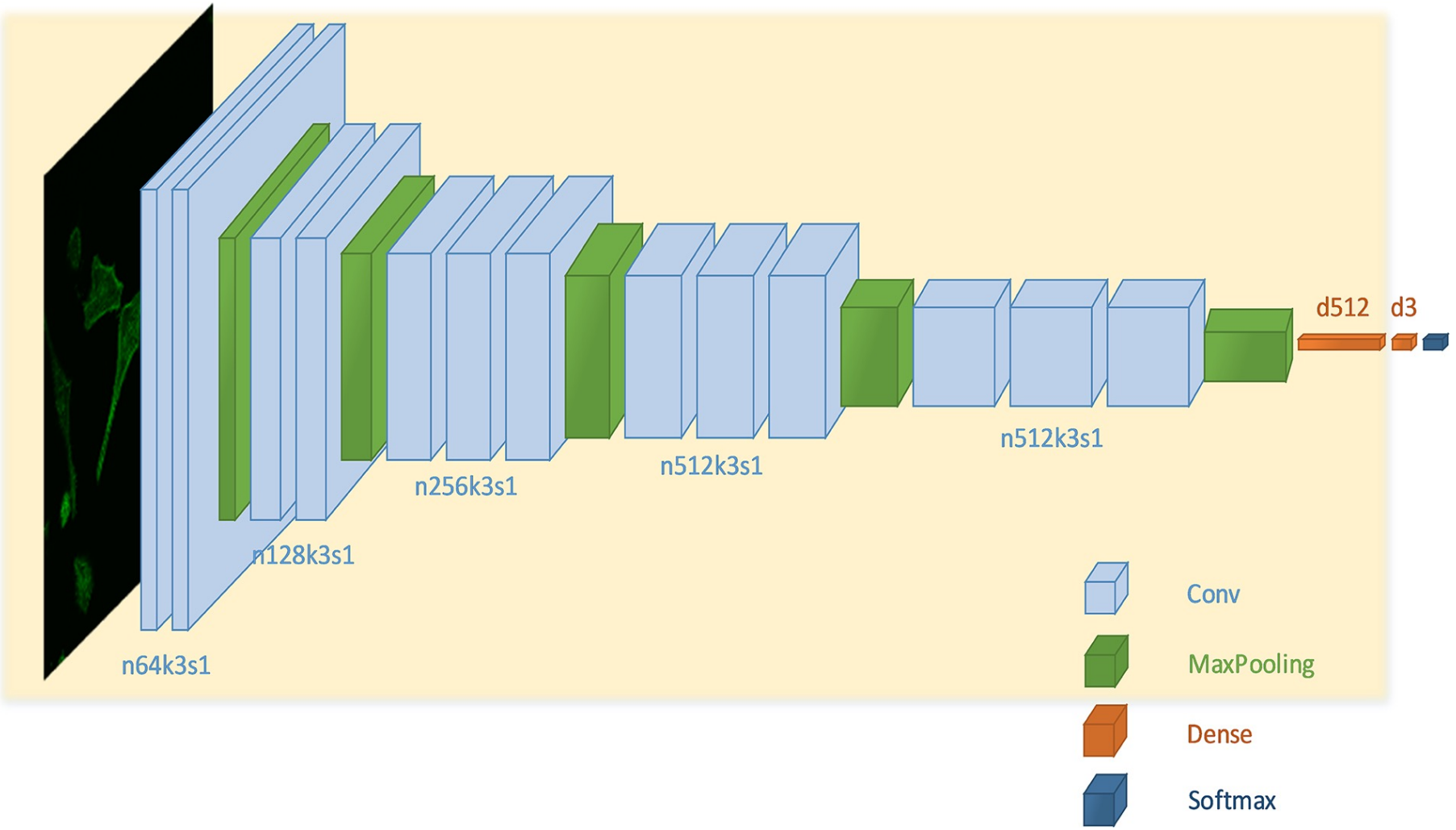
Convolutional Neural Networks (CNN)

- Uses convolutions we previously studied
- Adds a pooling layer
- The idea is based on the human eye
  - Nearby neurons are connected to other nearby neurons only, rather than all neurons (dense layer)
  - While the CNN is modeled on the eye—in the end, its version of “vision” is quite different
  - Still very good at image detection

High Level View of CNN

- Think of your input data as an array or matrix.
  - Color pictures are 3 arrays, 1 for each primary channel (red, green, blue).
  - Thus, a color picture is a 3D matrix of data (height, width, channels).
- Each time a filter is passed of the array, it creates a transformed array.
  - Remember, a convolution layer has multiple filters. If it has 10 filters, for a color picture it will produce 10 3D data arrays or 30 2D arrays (10 for each color channel).
- A pooling layer (discussed soon) shrinks the height and width. So if we pool after the above, we still have 30 2D arrays, but **fewer** rows and columns.
- Thus a CNN can be thought of as taking a large (height, width) image and creating many small “feature” or compressed images. Instead of 3 channels, we may end up with hundreds!

Visual Example of a CNN

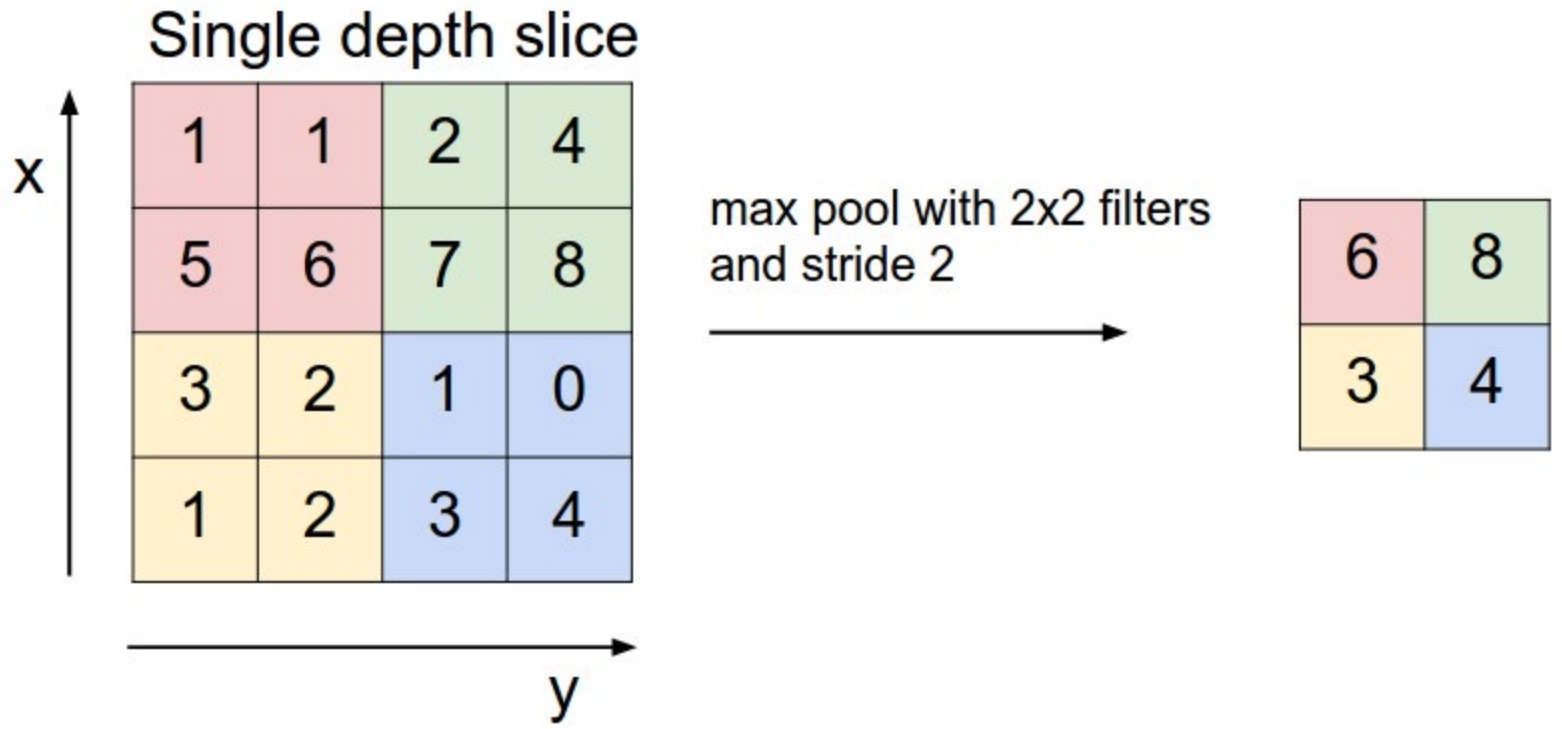


Final Parts of a CNN

- Once we have a large number of feature maps, they are projected to a 1D dense layer.
- That dense layer is then connected to an output (typically a classifier).

Pooling Layer

Pooling shrinks the data array by selecting either the max or average of a group of pixels.



Flatten Layer

Takes a 2D array and converts it into a 1D

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \end{bmatrix}$$

Note that this is done for every 2D array, so even though the data is “stacked,” the flatten layer converts all the 2D arrays into a single giant 1D array.

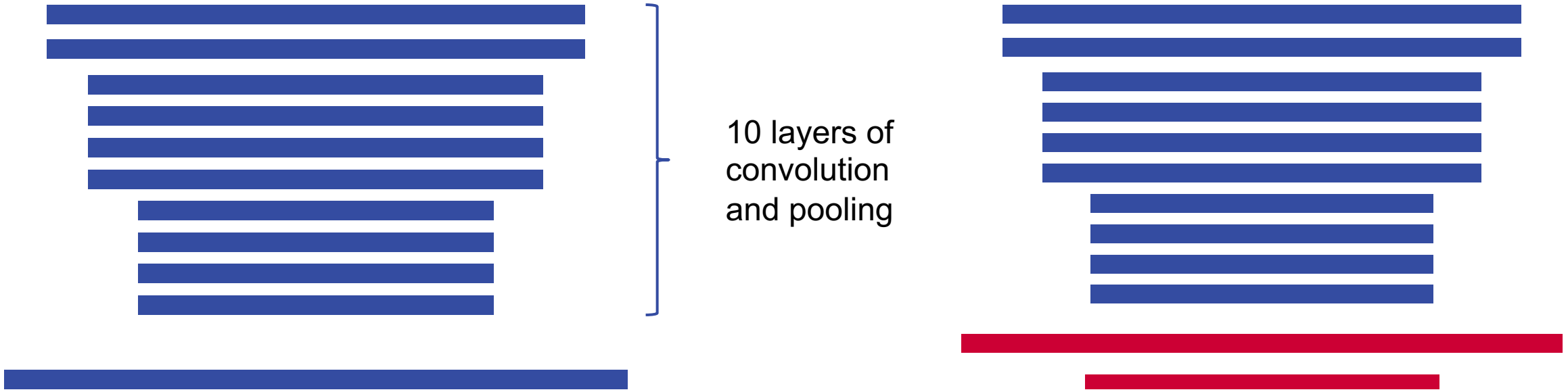
DataScience@SMU

Transfer Learning

Transfer Learning

- Remember the vanishing gradient? What if we could take advantage of it?
- Over time, it has been learned that layers near the input learn more general features.
  - They also learn very slowly.
- Often, new released models are run on supercomputers with tons of training time.
  - Since the early layers in the network are unlikely to change, our model needs to learn only the later layer weights.

Transfer Learning Diagram



Instead of relearning over thousands or even millions of examples, we put in two new dense layers and use the previous trained network as an input. We can even do different problems!

Transfer Learning

- Initially started in images
- Works for any deep neural network
  - Now in images and language processing
- Research groups release pretrained models
  - Use that model to produce “features” that train your network for your particular problem
  - Utilize models trained on hardware unavailable to the average person
- Many examples now included with libraries or released to public
  - TensorFlow Hub

DataScience@SMU