Convolutional Neural Networks

Reading

- Deep Learning: chapter 9
  - https://www.deeplearningbook.org/contents/convnets.html

Overview

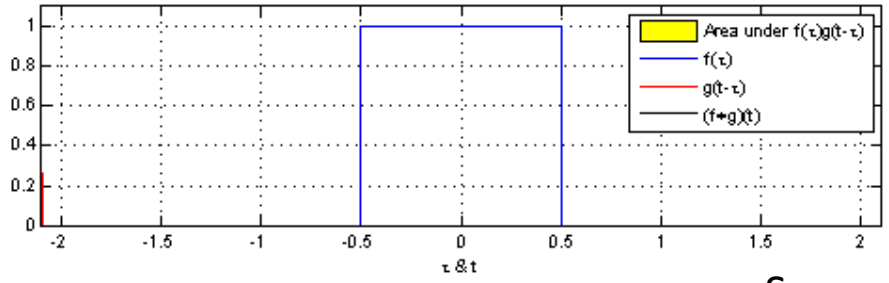
Rensselaer

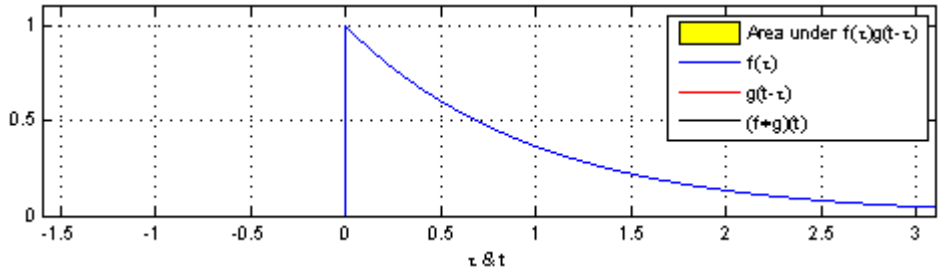
- CNNs are one of the most successful classes of NNs
- Specialized for tasks with known topology and grid-like inputs
  - E.g., image processing, point-cloud processing
- Inspired by the visual cortex of the brain that is known to perform convolution
  - Convolution is very effective at recognizing object edges, shades, etc.
  - David Hubel and Torsten Wiesel received a Nobel prize for these findings
- CNNs can be considered as a subclass of fully-connected NNs, with a bunch of 0 weights and a bunch of shared weights

The convolution operation

- In math, convolution is an operation on two functions that produces a third function
- Typically the functions are signals (i.e., functions of time)
- Convolution can capture many natural phenomena
  - Filtering out noise in data, modeling the response of a circuit to an electrical impulse, etc.
- Definition

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$





Source: wikipedia

Convolution, cont'd

- Also has a discrete-time version

$$(f * g)(n) = \sum_{k=-\infty}^{\infty} f(k)g(n-k)$$

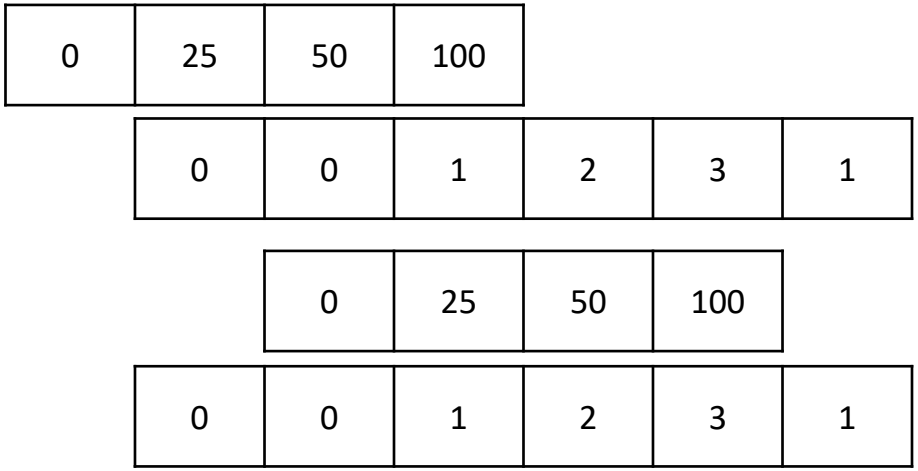
- Mostly used in analyzing control/industrial systems
  - g is the system's response to a "unit" impulse
  - f is the specific input signal (multiplied by the impulse)
  - g is reversed for mathematical convenience
- Convolution is commutative

$$f * g = g * f$$

  – this is the benefit of reversing time

Example

- Suppose during my PhD I publish 0,0,1,2,3,1 papers each year
  - Each paper is cited 100,50,25,0 times a year at 0,1,2,3 years
- How do I calculate my total citations per year?
  - Convolution!
- Citations per paper is the system's response, g, to my input, f
- To calculate number, reverse g and convolve
- In year 3
  - 100
- In year 5
  - 300 + 100 + 25 = 425

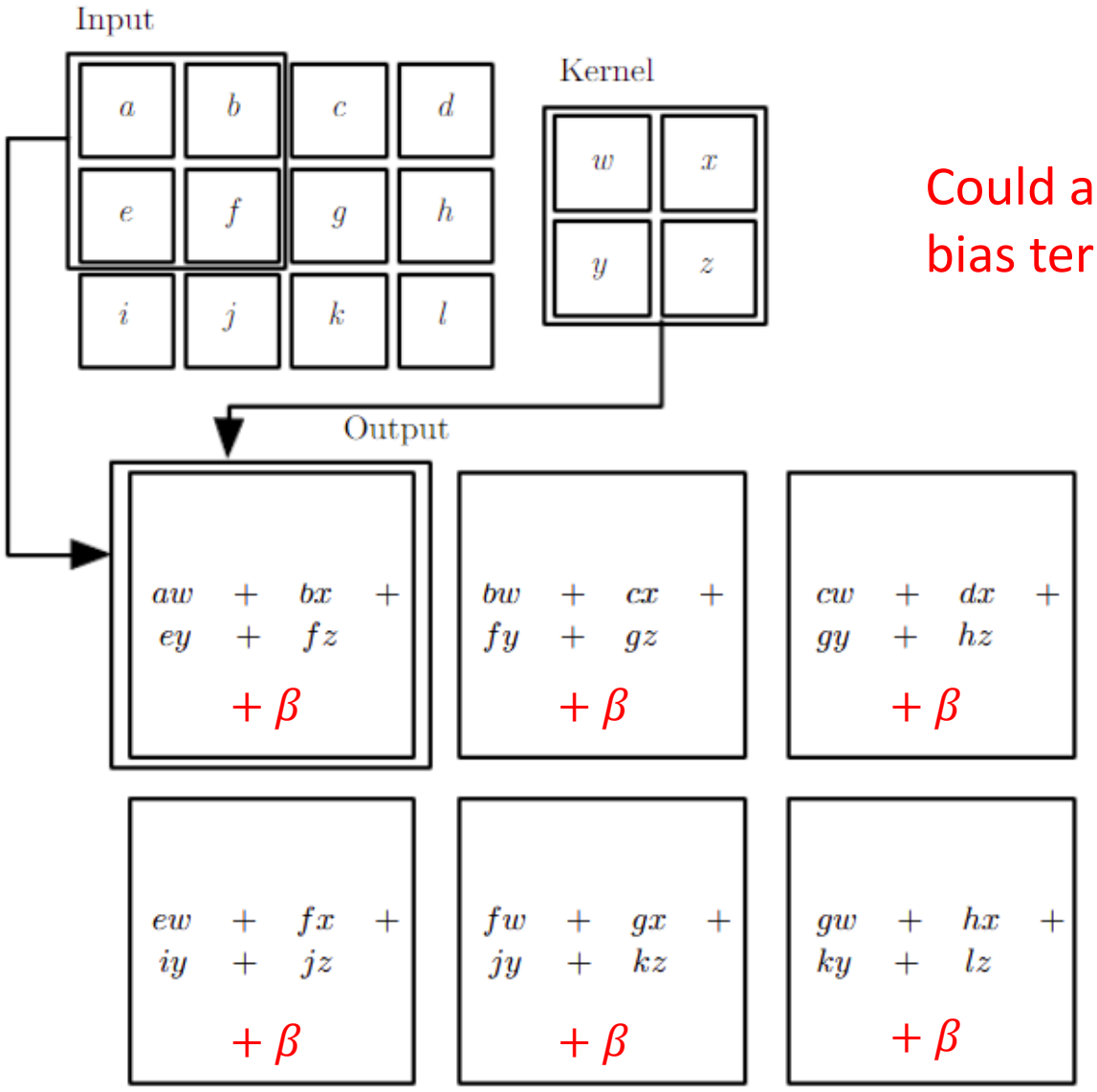


Motivation

Rensselaer

- Pattern matching
  - signal f is the input example (e.g., image)
  - signal g is also known as a kernel in ML
  - a convolutional filter tries to find similar patterns in the data (e.g., cats, dogs, etc.)
- Training
  - Each kernel has few parameters, so easier to train
  - Similar to a (very) sparse fully-connected NN
- Equivariant representation
  - More on this later

Convolution example



Could also add a bias term β

Convolutional Kernel Structure

- Dimensions: $m \times n$
  - Usually 2D, but can also be 1D
  - A total of $m * n$ weights (plus one optional bias parameter)
- Images usually have c channels (i.e., a 3rd dimension)
  - e.g., RGB (red, green, blue)
  - one weight per channel
  - total number of weights becomes $m * n * c$
- Kernel dimensions usually much smaller than input
  - Need to slide it right and down, using same weights – **shared parameters**

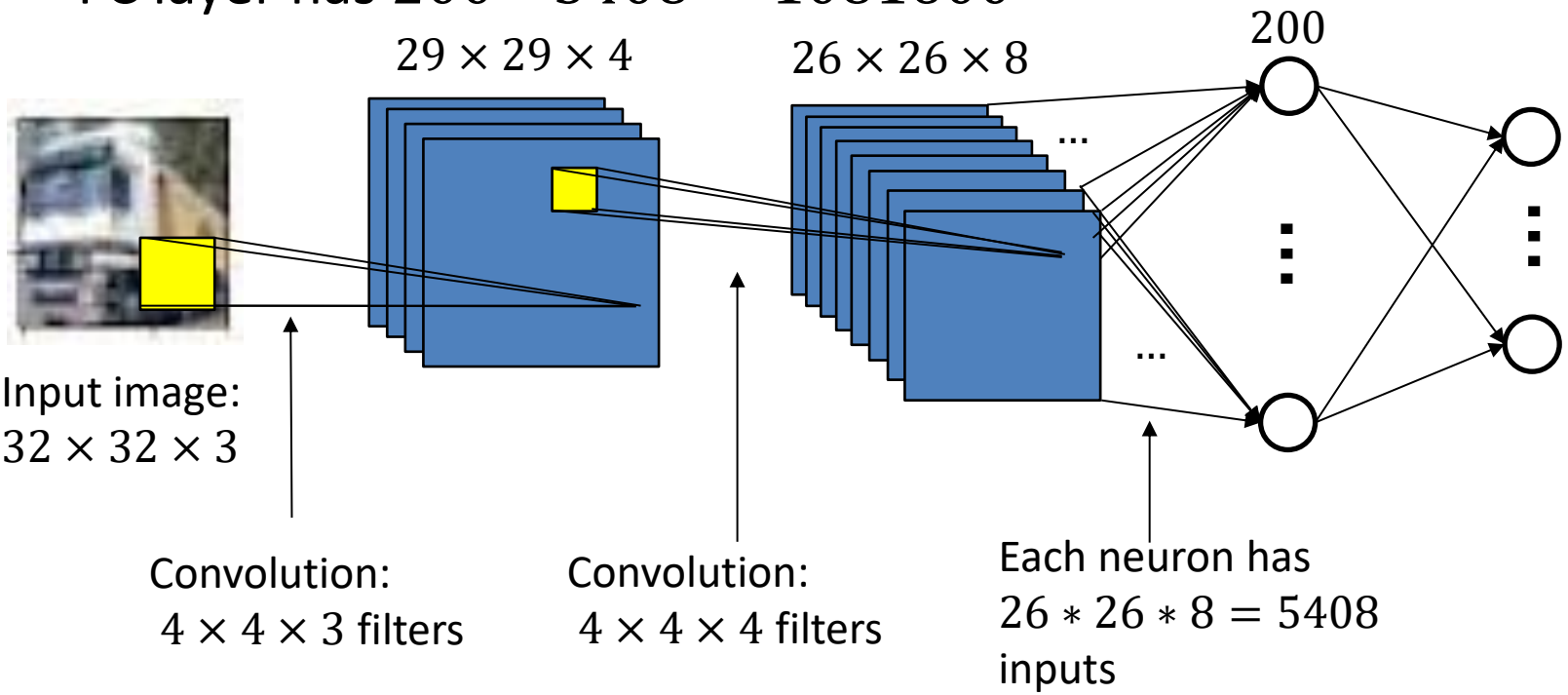
Convolutional Kernel Structure, cont'd

Rensselaer

- Stride
  - Step size when sliding (right and down)
  - Typical values are 1 and 2, though others possible, too
- Padding
  - If kernel doesn't fit on the last step(s), extend image by 0s
  - Could distribute the 0s on both sides evenly
- Computing output dimension of a kernel is tricky because of these issues
  - E.g., suppose input is a 32×32 image and kernel is 4×4 with a stride of 1
  - Can slide it across and down 29 times, so output is 29×29
  - You need to know these details when constructing your NN

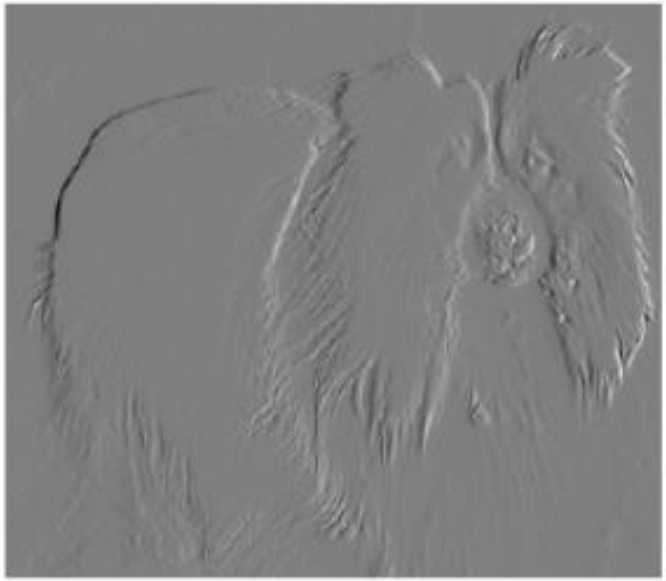
CNN Example: CIFAR10

- Simple CNN architecture on CIFAR10
  - Each filter's output is one channel in the next layer
  - How many weights does the 2nd hidden layer have?
    - Each filter has 4^3 weights, so total number is $8 * 4^3 = 512$
  - FC layer has $200 * 5408 = 1081600$



A very simple convolution

- Take an image and form a new image by subtracting from each pixel its neighboring pixel to the left



What does the kernel look like?

[-1 1]

Convolutional Layers as Sparse "Fully"-Connected Layers

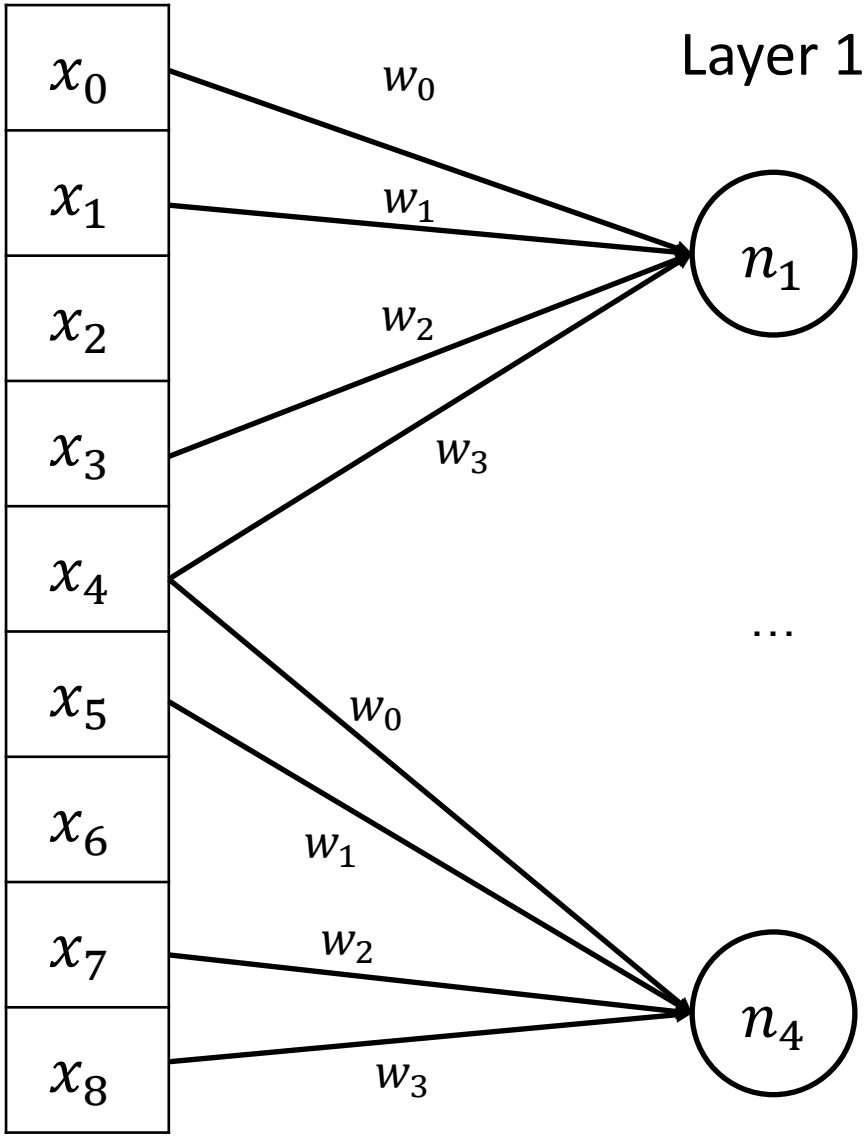
Input

x_0	x_1	x_2
x_3	x_4	x_5
x_6	x_7	x_8

Kernel

w_0	w_1
w_2	w_3

Convolutional Layers as Sparse "Fully"-Connected Layers



Note the drastic savings in number of parameters!

- A fully connected layer would have $4 * 9$ parameters

Benefits of convolutional layers over fully connected layers

Rensselaer

- Very few parameters for similar representation complexity
  - Fully-connected NNs are universal approximators but finding the right parameters is hard in high-dimensional spaces
  - CNNs are tailored for grid-like inputs and have sufficient expressive power with fewer parameters
- Easier to train
  - Both faster and better optimization since searching in lower-dimensional spaces
- Require less memory
  - Not terribly important but could make a difference on an embedded device

Translational equivariance

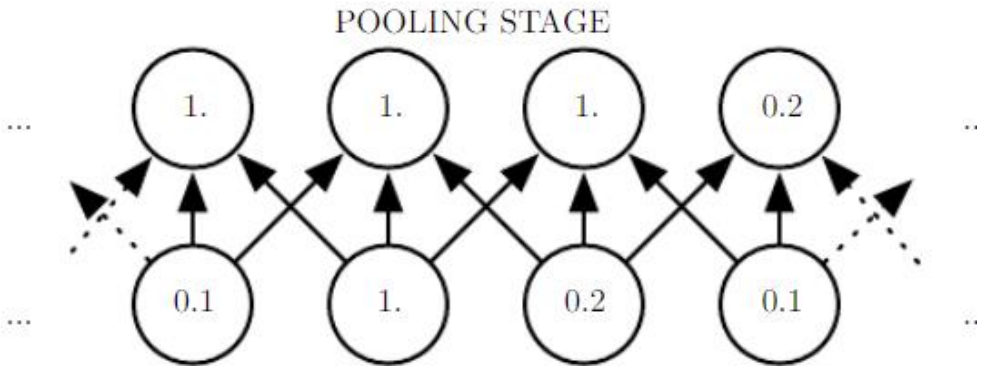
- If we translate the image (e.g., move to the right), the convolved image is similarly translated

$$f\big(g(x)\big) = g(f(x))$$

- Not quite invariant to translation, but next best thing
  - Useful image features still propagated to the next layer
  - For example, first layer may be trained to detect edges
- Convolution is not naturally invariant to rotation or scale
  - Other mechanisms are necessary for these
  - E.g., data-augmentation

Pooling

- A small layer that usually goes hand in hand with a convolutional layer
  - Newer architectures do not use it as much, so it doesn't seem to be essential
- Usually convolution is followed by an activation as before
  - Pooling comes after the activation
- Pooling is a local function that we slide across the image
  - E.g., take the mean/max of nearby pixels



Pooling Motivation

Rensselaer

- Local translation invariance
  - If we shift pixels by a few places, max pooling will not be affected
- Useful for downsampling
  - If input image is too large, can use max pooling to reduce it to required size while preserving high-level features
  - If used in the middle of the NN, can reduce the number of parameters in the rest of the NN

Typical Convolutional Layer Summary

- Picking kernel size depends on application
  - Standard choices are $8 \times 8, 16 \times 16$
  - Might need bigger if image is very big
- Stride is usually 1 but bigger values may be more efficient (fewer outputs)
- Padding is not very important, but you need to be careful since it affects the number of outputs
- Pooling does not help tremendously but may lead to more stable training
- Typically, a CNN ends with a fully-connected layer or two, partly to reshape the output