The Convolution Operation

Sargur Srihari
srihari@buffalo.edu

Topics in Convolutional Networks

- Overview

1. The Convolution Operation
2. Motivation
3. Pooling
4. Convolution and Pooling as an Infinitely Strong Prior
5. Variants of the Basic Convolution Function
6. Structured Outputs
7. Data Types
8. Efficient Convolution Algorithms
9. Random or Unsupervised Features
10. The Neuroscientific Basis for Convolutional Networks
11. Convolutional Networks and the History of Deep Learning

Plan of discussion

1. What is convolution?
2. Convolution: continuous and discrete cases
3. Convolution in two dimensions
4. Discrete convolution viewed as matrix multiplication

What is convolution?

- Convolution is an operation on two functions of a real-valued argument
- Examples of the two functions
  - Tracking location of a spaceship by a laser sensor
    - A laser sensor provides a single output $x(t)$, the position of spaceship at time t
  - w a function of a real-valued argument
    - If laser sensor is noisy, we want a weighted average that gives more weight to recent observations
    - Weighting function is $w(a)$ where a is age of measurement
- Convolution is the weighted average or smoothed estimate of the position of the spaceship
  - A new function s

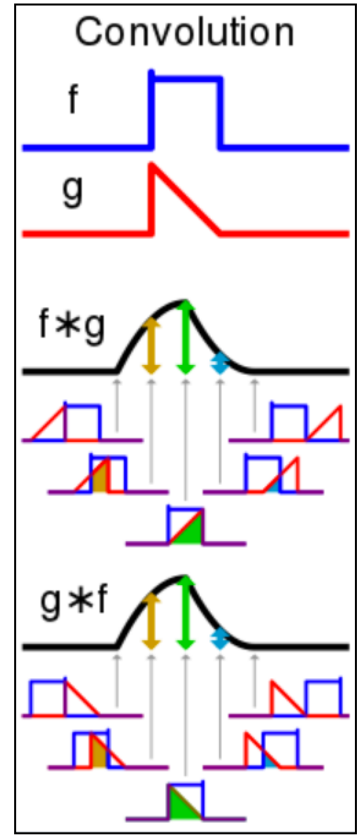
$$\boxed{s(t) = \int x(a)w(t-a)\,da}$$

What is Convolution?

- One-dimensional continuous case
  - Input $f(t)$ is convolved with a kernel $g(t)$

$$\boxed{(f * g)(t) \equiv \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau}$$

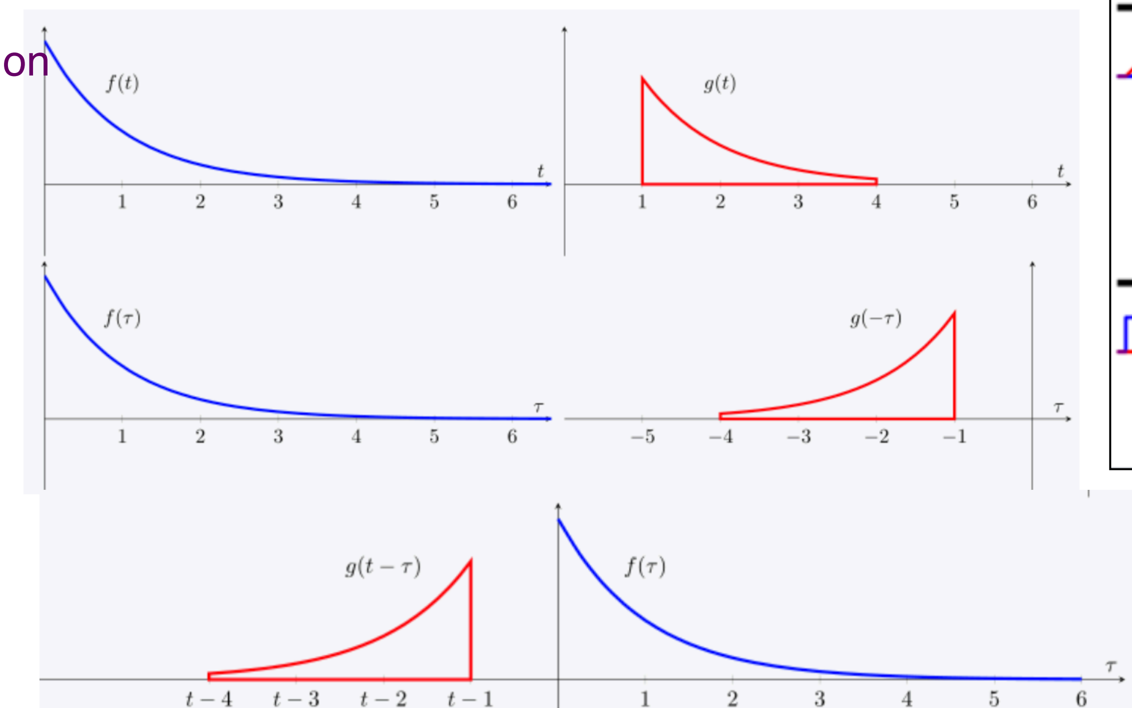
Note that $(f * g)(t) = (g * f)(t)$



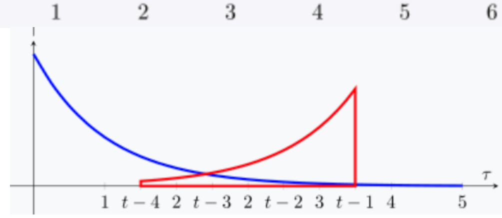
1.Express each function in terms of a dummy variable τ

2. Reflect one of the functions $g(\tau) \rightarrow g(-\tau)$

3. Add a time offset t, which allows $g(t-\tau)$ to slide along the τ axis

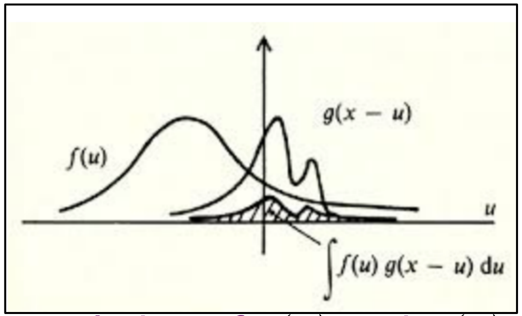


4. Start t at $-\infty$ and slide it all the way to $+\infty$
Wherever the two functions intersect find the integral of their product



https://en.wikipedia.org

Definition of convolution of input and kernel



Convolution of $f(u)$ and $g(u)$

- Convolution is a new function s, the weighted average of x

$$s(t) = \int x(a)w(t-a)\,da$$

  - This operation is typically denoted with an asterisk

$$s(t) = (x * w)(t)$$

  - w needs to be a valid pdf, or the output is not a weighted average
  - w needs to be 0 for negative arguments, or we will look into the future
- In convolution network terminology the first function x is referred to as the *input*, the second function w is referred to as the *kernel*
- The output s is referred to as the feature map

Convolution with Discrete Variables

- Laser sensor may only provide data at regular intervals
- Time index t can take on only integer values
  - x and w are defined only on integer t

$$\boxed{s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)}$$

- In ML applications, input is a multidimensional array of data and the kernel is a multidimensional array of parameters that are adapted by the learning algorithm
  - These arrays are referred to as tensors
- Input and kernel are explicitly stored separately
  - The functions are zero everywhere except at these points

Convolution in discrete case

- Here we have discrete functions f and g

$$(f * g)[t] = \sum_{\tau=-\infty}^{\infty} f[\tau] \cdot g[t-\tau]$$

$f[t]$							
	0	1	2	-1	1	-3	0

$g[t-\tau]$			
	1	0	-1

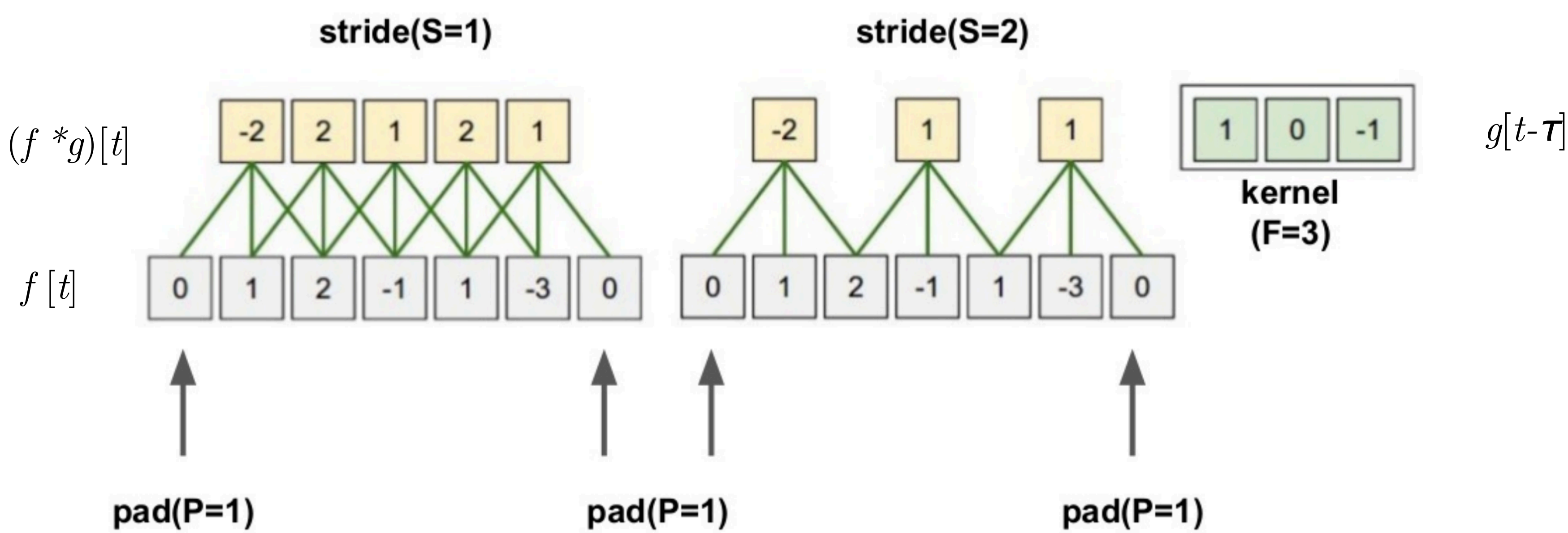
Computation of 1-D discrete convolution

Parameters of convolution:

Kernel size (F)

Padding (P)

Stride (S)



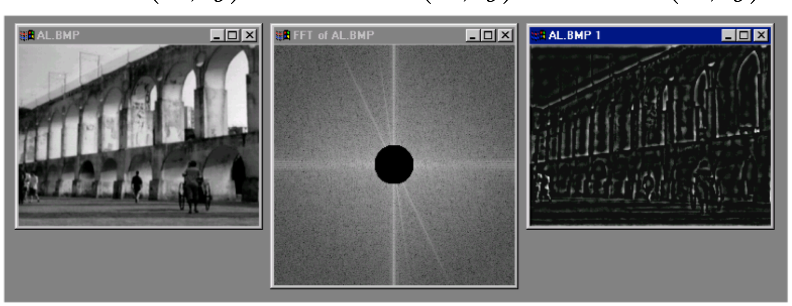
Two-dimensional convolution

- Convolutions over more than one axis
- If we use a 2D image I as input and use a 2D kernel K we have

$$\boxed{S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)}$$

Sharply peaked kernel K for edge detection

$I(i\ ,\ j)$ $K(i\ ,\ j)$ $S(i\ ,\ j)$



Kernels K_1-K_4 for line detection

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal lines

-1	2	-1
-1	2	-1
-1	2	-1

Vertical lines

-1	-1	2
-1	2	-1
2	-1	-1

45 degree lines

2	-1	-1
-1	2	-1
-1	-1	2

135 degree lines

Commutativity of Convolution

- Convolution is commutative. We can equivalently write:

$$\boxed{S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)}$$

- This formula is easier to implement in an ML library since there is less variation in the range of valid values of m and n
- Commutativity arises because we have flipped the kernel relative to the input
  - As m increases, index to the input increases, but index to the kernel decreases

Cross-Correlation

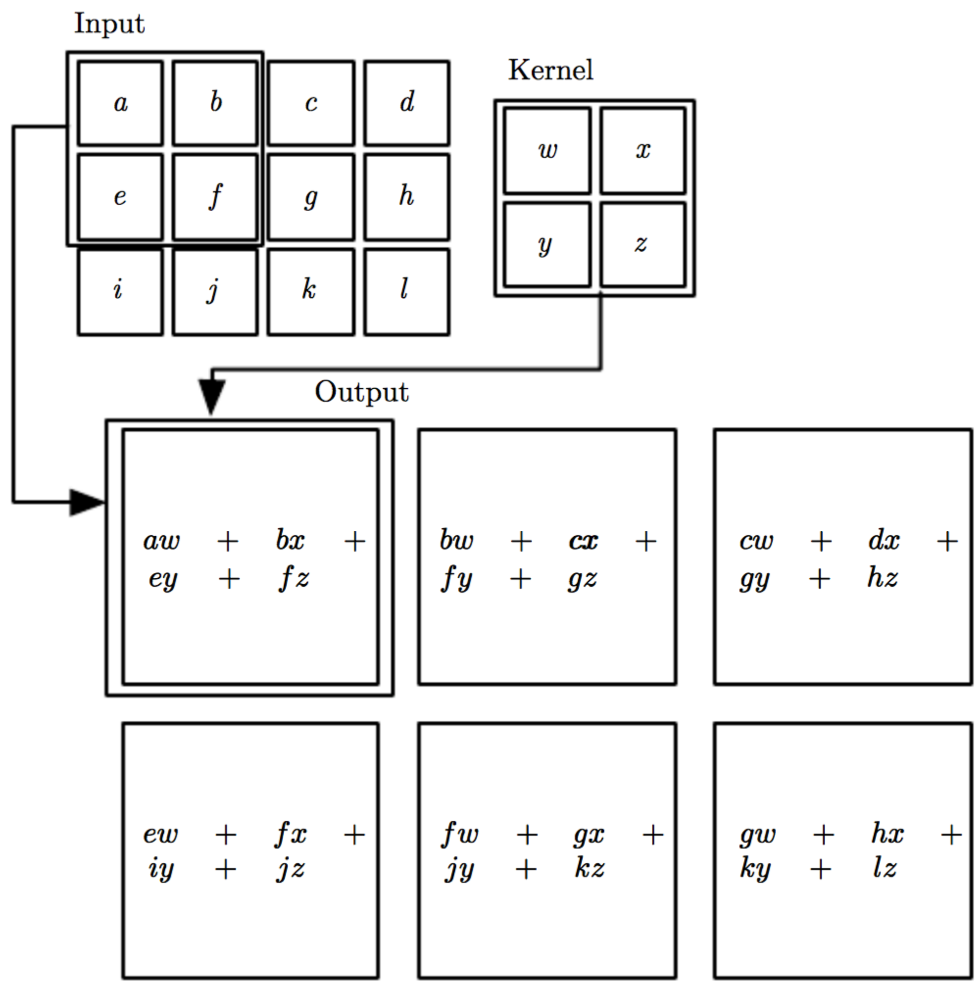
- Same as convolution, but without flipping the kernel

$$\boxed{S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)}$$

- Both referred to as convolution, and whether kernel is flipped or not
- In ML, learning algorithm will learn appropriate values of the kernel in the appropriate place

Example of 2D convolution

- Convolution without kernel flipping applied to a 2D tensor
- Output is restricted to case where kernel is situated entirely within the image
- Arrows show how upper-left of input tensor is used to form upper-left of output tensor



Discrete Convolution Viewed as Matrix multiplication

- Convolution can be viewed as multiplication by a matrix
- However the matrix has several entries constrained to be zero
- Or constrained to be equal to other elements
  - *For univariate discrete convolution: Univariate Toeplitz* matrix:
    - Rows are shifted versions of previous row

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & \cdots & a_{-(n-1)} \\ a_1 & a_0 & a_{-1} & \ddots & & \vdots \\ a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a_{-2} \\ \vdots & & \ddots & a_1 & a_0 & a_{-1} \\ a_{n-1} & \cdots & \cdots & a_2 & a_1 & a_0 \end{bmatrix}$$

  - 2D case: *doubly block circulant matrix*
    - It corresponds to convolution

$$C = \begin{bmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{bmatrix}$$