Lecture 12: Convolutional Neural Networks
CS480/680 Intro to Machine Learning

2023-2-16

Pascal Poupart
David R. Cheriton School of Computer Science

UNIVERSITY OF WATERLOO

Large networks

- What kind of neural networks can be used for large or variable length input vectors (e.g., time series)?

- Common networks:
  - Convolutional networks
  - Recurrent networks
  - Transformers

Convolution

- Convolution: mathematical operation on two functions $x()$ and $w()$ that produces a third function $y()$ that can be viewed as a modified version of the original function $x()$

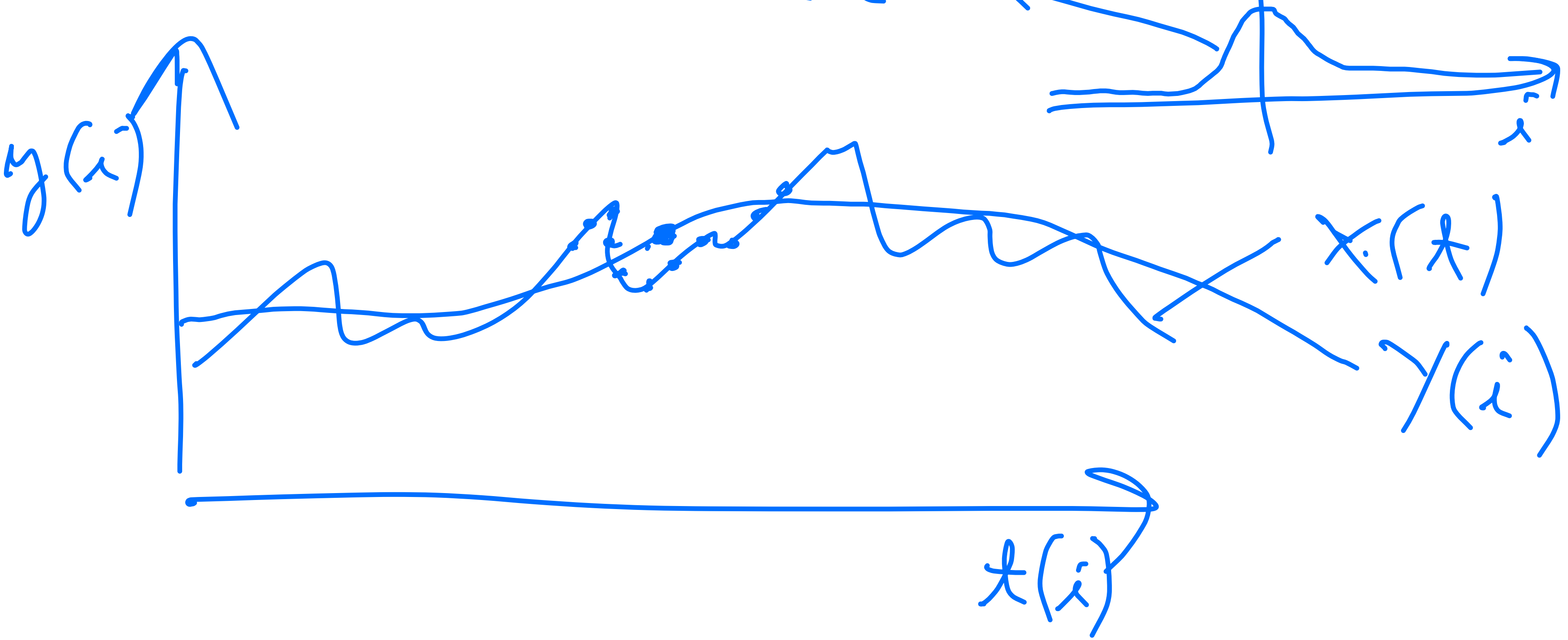
$$y(i) = \int_t x(t)w(i-t)dt$$

$$y(i) = (x * w)(i)$$

where $*$ is an operator denoting a convolution

Example Smoothing

$$e^{\frac{(t-i)^2}{2\sigma^2}} = w(i)$$



Discrete convolution

- Discrete convolution

$$y(i) = \sum_{t=-\infty}^{\infty} x(t)w(i-t)$$

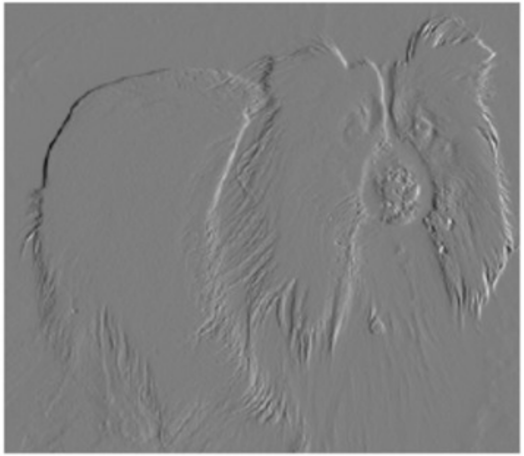
- Multidimensional convolution

$$y(i,j) = \sum_{t_1=-\infty}^{\infty} \sum_{t_2=-\infty}^{\infty} x(t_1,t_2)w(i-t_1,j-t_2)$$

Example: Edge Detection

- Consider a grey scale image
- Detect vertical edges: $y(i,j) = x(i,j) - x(i-1,j)$

$$w(i-t_1, j-t_2) = \begin{cases} 1 & t_1 = i, t_2 = j \\ -1 & t_1 = i-1, t_2 = j \\ 0 & \text{otherwise} \end{cases}$$



Convolutions for feature extraction

- In neural networks
  - A **convolution** denotes the linear combination of a **subset of units** based on a **specific pattern of weights.**

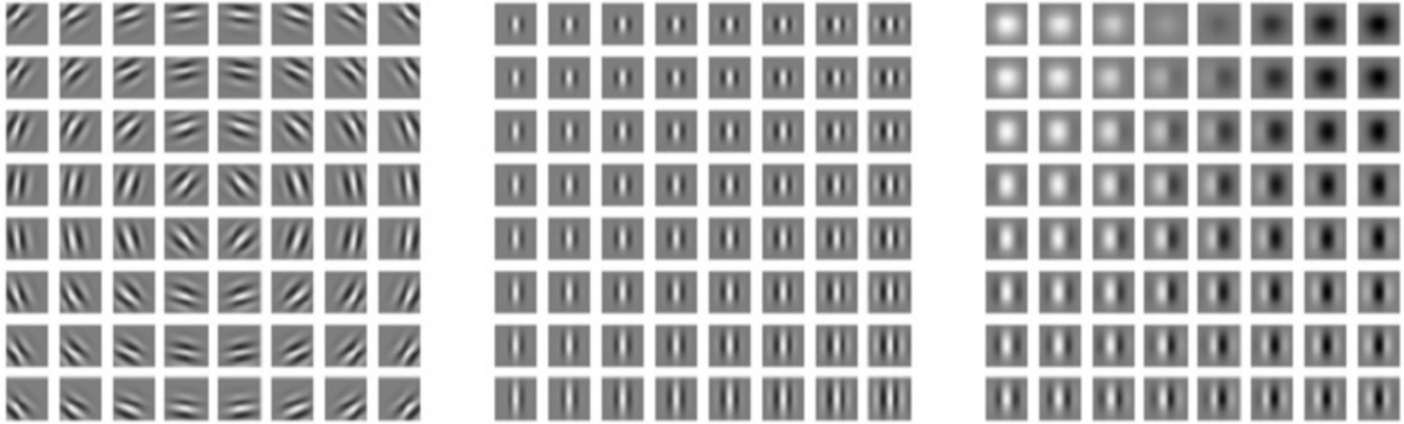
$$a_j = \sum_i w_{ji} z_i$$

  - Convolutions are often combined with an activation function to produce a feature

$$z_j = h(a_j) = h\left(\sum_i w_{ji} z_i\right)$$

Gabor filters

- Gabor filters: common feature maps inspired by the human vision system

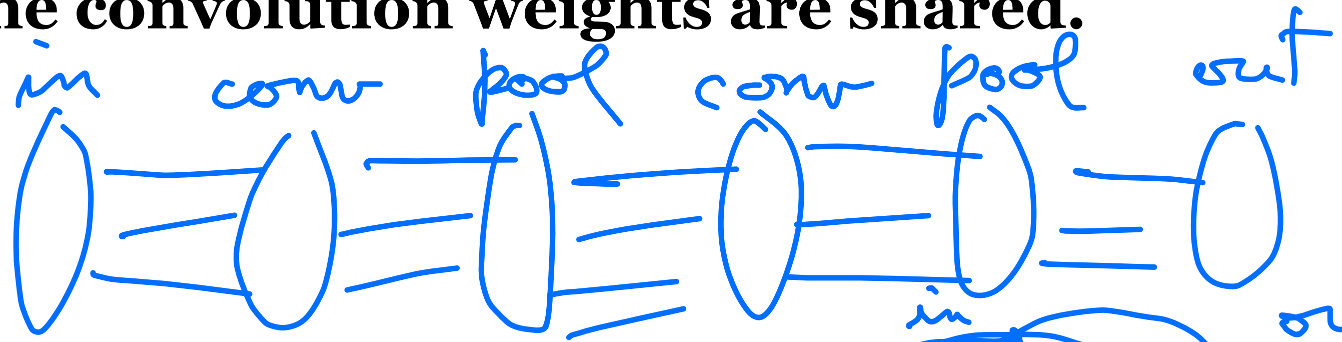


- Weights:

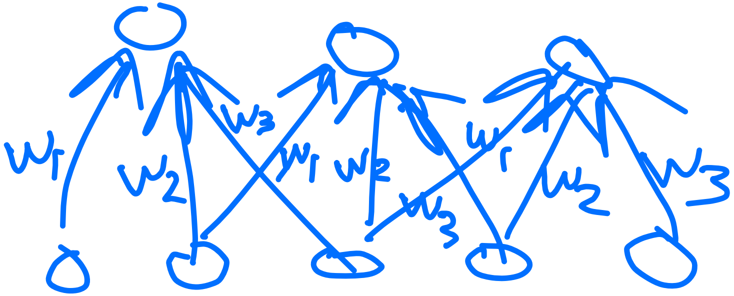
  Grey: zero White: positive Black: negative

Convolution Neural Network

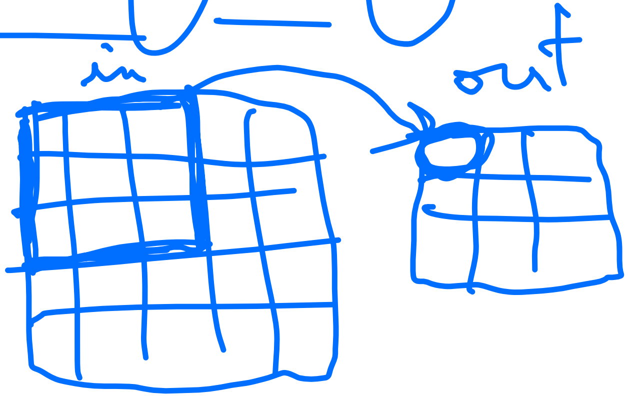
- A **convolutional neural network** refers to any network that includes an **alternation of convolution and pooling layers**, where **some of the convolution weights are shared.**
- Architecture:



convolution



filter of size 3
w_1, w_2, w_3

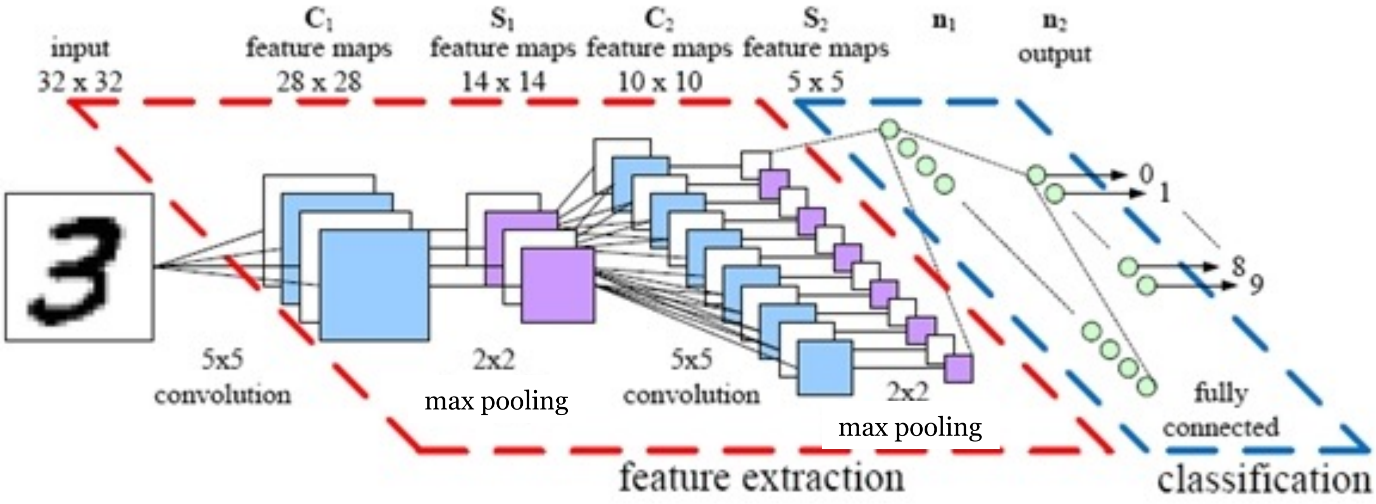


Pooling

- Pooling: **commutative** mathematical operation that combines several units
- Examples:
  - max, sum, product, average, Euclidean norm, etc.
- Commutative property (order does not matter):

$$\text{Ex.: } \max(a, b) = \max(b, a)$$

Example: Digit Recognition



UNIVERSITY OF WATERLOO

Benefits

- Sparse interactions
  - Fewer connections
- Parameter sharing
  - Fewer weights
- Locally equivariant representation
  - Locally invariant to translations
  - Handle inputs of varying length

Parameters

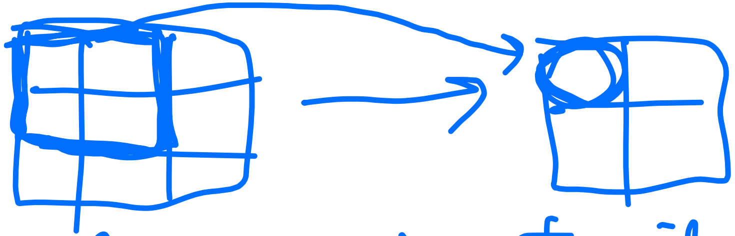
- **# of filters**: integer indicating the # of filters applied to each window.

filter

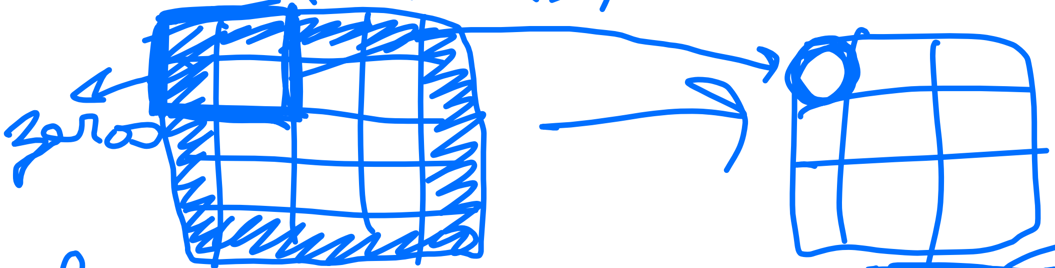
- **kernel size:** tuple (width, height) indicating the size of the window.
- **Stride:** tuple (horizontal, vertical) indicating the horizontal and vertical shift between each window.
- **Padding:** "valid" or "same". Valid indicates no input padding. Same indicates that the input is padded with a border of zeros to ensure that the output has the same size as the input.

Examples

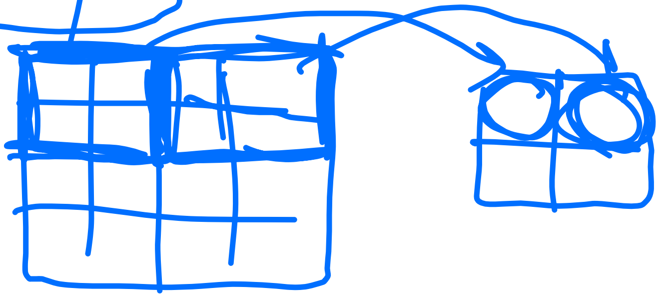
kernel (filter) (2x2) stride (1x1) padding "valid"



kernel (3x3) stride (1x1) padding "same"



kernel (2x2)
stride (2x2)
padding "valid"



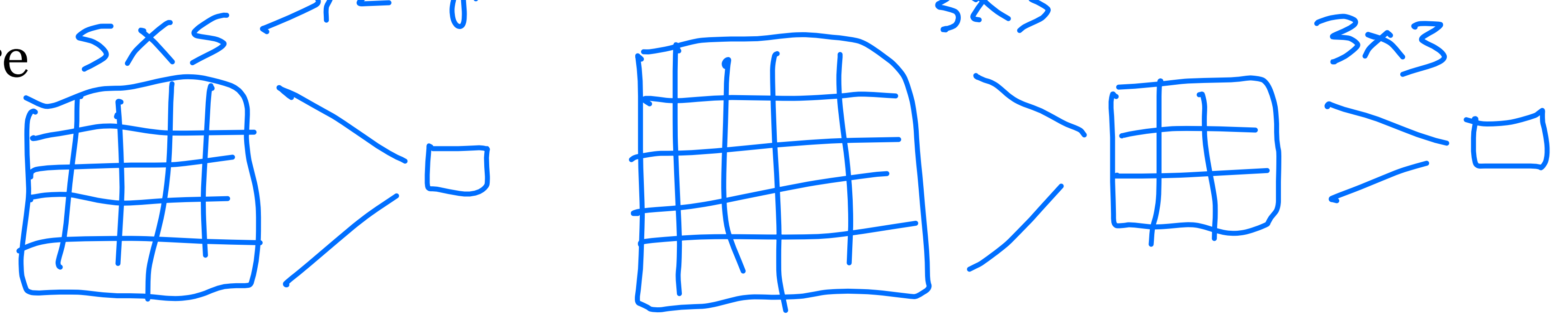
Training

- Convolutional neural networks are trained in the same way as other neural networks
  - E.g., backpropagation

- Weight sharing:
  - Combine gradients of shared weights into a single gradient

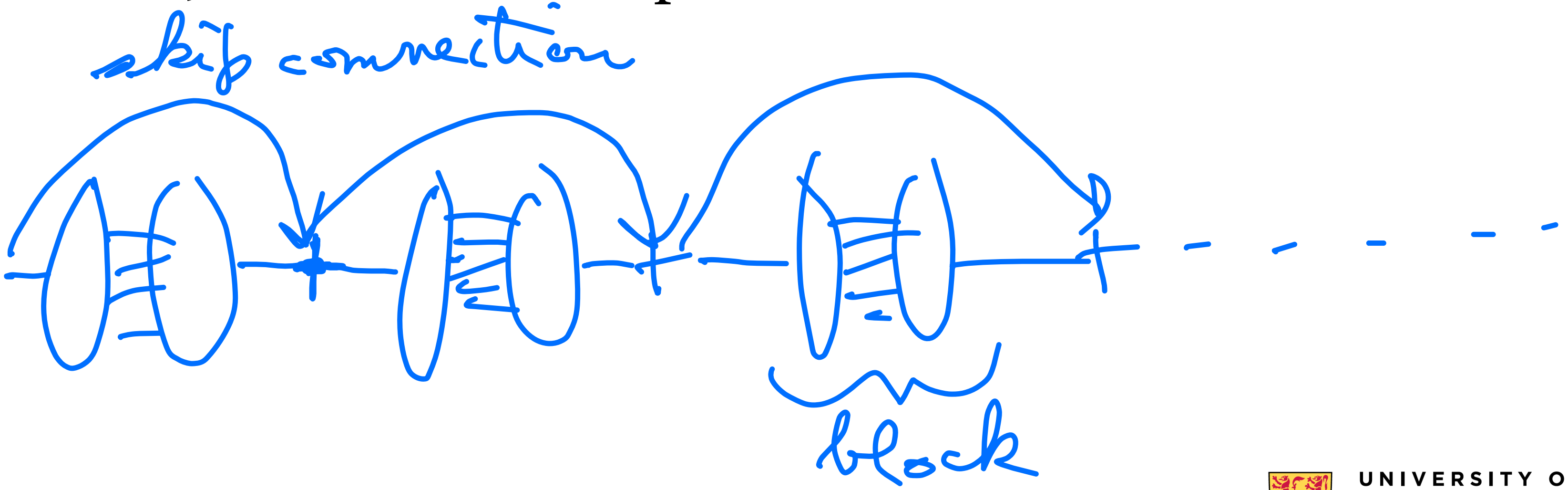
Architecture design

- What is the preferred filter size?
- VGG (Visual Geometry Group at Oxford, 2014): stack of small filters is often preferred to a single large filter
  - Fewer parameters
  - Deeper network
- Picture



Residual Networks

- Problem: very deep networks can perform worse than shallower networks (due to local optima & other stationary non-optimal points)
- Solution [He et al., 2015]: introduce **residual connections** (a.k.a. skip connections) to make blocks optional
- Picture:



UNIVERSITY OF WATERLOO

Residual Learning

- Consider a block $b(\boldsymbol{x})$ that ends with a linear layer
  - i.e., $b(\boldsymbol{x}) = \boldsymbol{w}^T g(\boldsymbol{x})$ where $g(\boldsymbol{x})$ can be anything
- We can nullify $b(\boldsymbol{x})$ simply by setting $\boldsymbol{w}$ to $\boldsymbol{0}$
  - i.e., $b(\boldsymbol{x}) = \boldsymbol{0}^T g(\boldsymbol{x}) = \boldsymbol{0}$
- Hence, when $b(\boldsymbol{x}) = \boldsymbol{0}$,
  then $f(\boldsymbol{x}) = b(\boldsymbol{x}) + \boldsymbol{x} = \boldsymbol{x}$ computes the identity

Applications

- Image processing
- Data with sequential, spatial, or tensor patterns