ECE 449/590 – OOP and Machine Learning
Lecture 17 Convolutional Networks

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

October 24, 2022

Outline

Convolution

Pooling

Reading Assignment

- This lecture: Deep Learning 9
- Next lecture: Deep Learning 5, 6

Outline

Convolution

Pooling

Convolutional Networks

- A.k.a. convolutional neural networks, or CNNs.
- For data that has a known grid-like topology, e.g.
  - 1-D: time-series
  - 2-D: images
- Make use of convolution in at least one of the neural network layers.
  - Specialized kind of linear operation.

The Convolution Operation

- Convolution: average input x with kernel w.

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

  - Typically written as $s(t) = (x * w)(t)$
  - From aspects of signals and systems, w is the impulse response of the associated linear and time-invariant (LTI) system.
- Most neural network libearies implement convolution as cross-correlation.
  - E.g. 2-D: $S(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$
  - Note that $S(i,j)$, $I(i,j)$ could be vectors and $K(m,n)$ are therefore matrices, requiring S, I, K to be tensors themselves.

2D Convolution

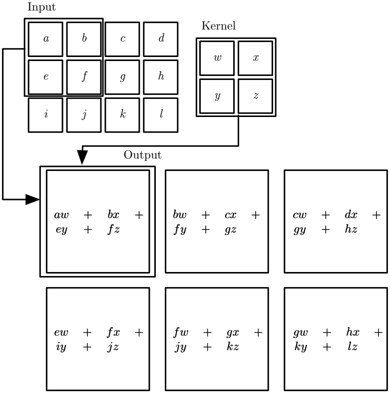


Figure 9.1

(Goodfellow 2016)

- Only keep an output cell when all input cells are defined.
  - Neural network libearies usually use padding to expand the input so that output could remain the same size.

Zero Padding Controls Size

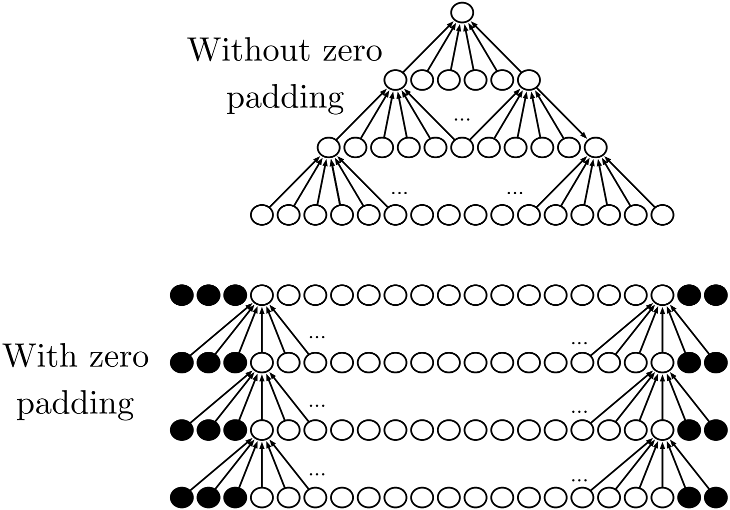


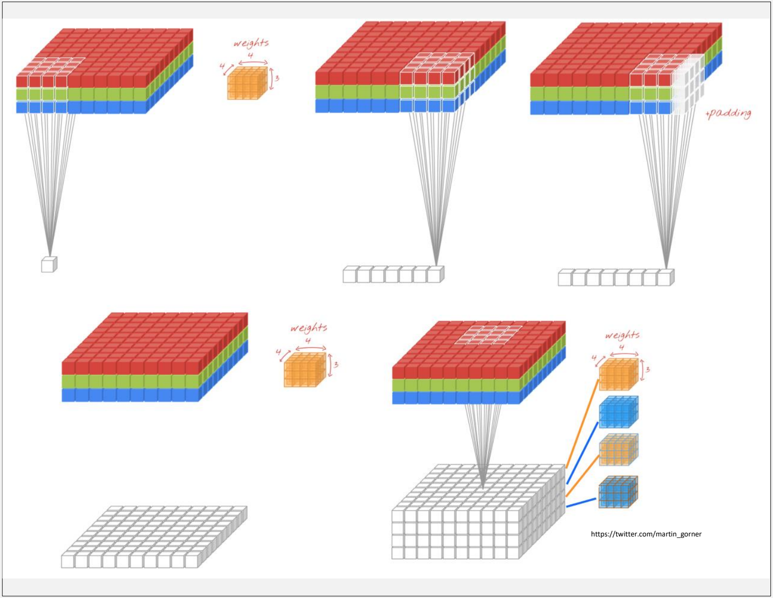
Figure 9.13

(Goodfellow 2016)

Convolution with Input and Output Channels

- In practice, convolutions are usually applied to a group of input channels to generate a group of output channels.
- For a 2D image,
  - With color channels, the input to the first layer is a 3D tensor (W, H, input channels).
  - A kernel as a 3D tensor will only generate a 2D output.
  - Instead, we use a kernel that is a 4D tensor so the output is a 3D tensor (W, H, output channels)
  - Layers using 4D kernels can now be easily composed together.

Convolution with Input and Output Channels (Cont.)



Why convolution?

- Sparse interactions: by using a small kernel, interaction between inputs and outputs are limited.
- Parameter sharing: same kernel is applied many times.
- Equivariant representations: help to learn representations that are invariant to input locations.
- Compared to the (fully-connected) linear layer,
  - Able to work with inputs with variable size
  - Much less capacity and storage $O(k)$ (vs $O(n^2)$).
  - Less computation $O(kn)$ (vs $O(n^2)$).

Sparse Connectivity

Sparse connections due to small convolution kernel

Dense connections

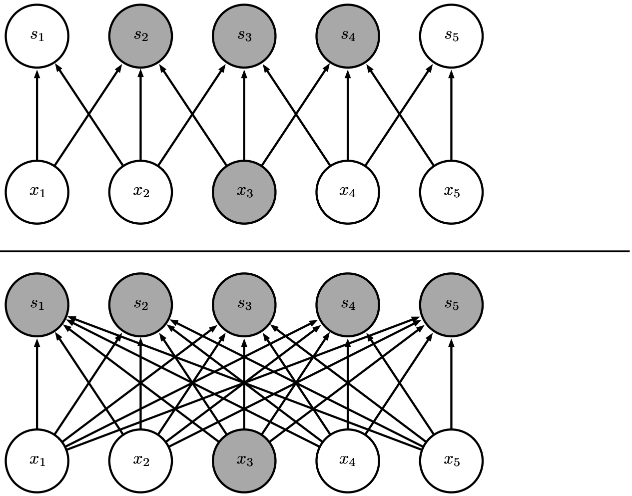


Figure 9.2

(Goodfellow 2016)

Sparse Connectivity

Sparse connections due to small convolution kernel

Dense connections

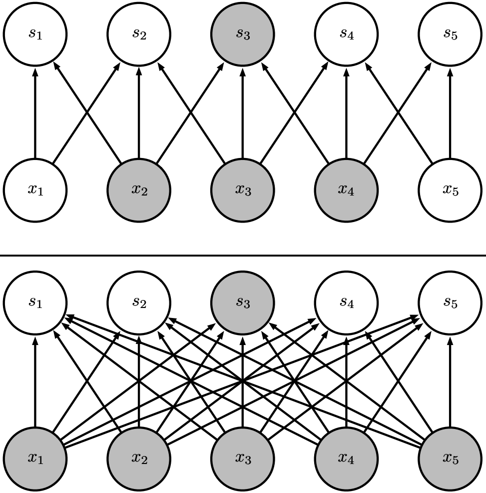


Figure 9.3

(Goodfellow 2016)

Growing Receptive Fields

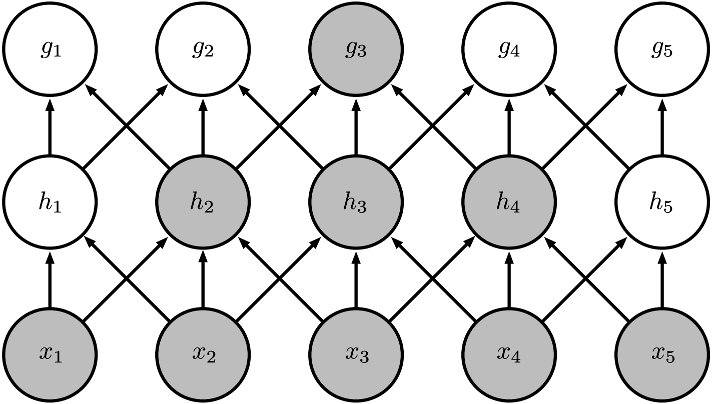
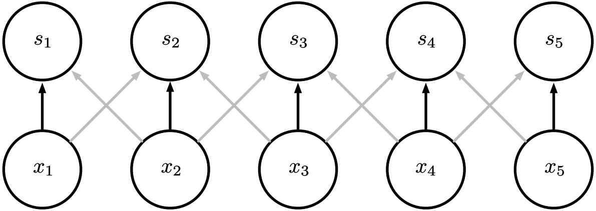


Figure 9.4

(Goodfellow 2016)

Parameter Sharing

Convolution shares the same parameters across all spatial locations



Traditional matrix multiplication does not share any parameters

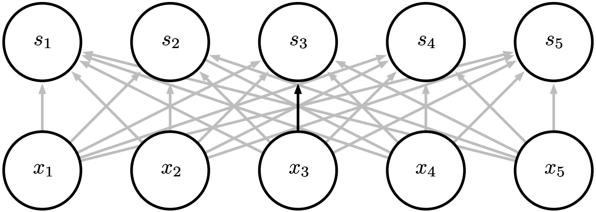


Figure 9.5

(Goodfellow 2016)

Edge Detection by Convolution

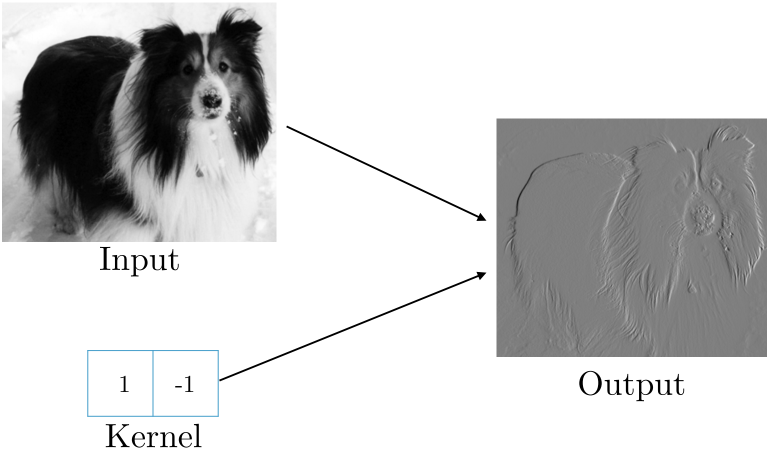
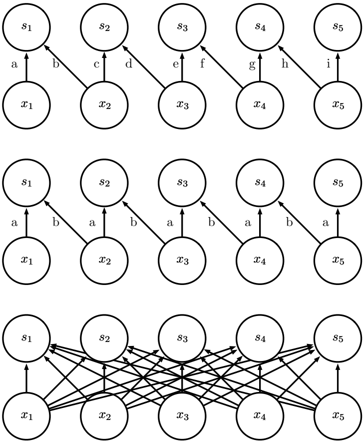


Figure 9.6

(Goodfellow 2016)

Kinds of Connectivity



Local connection: like convolution, but no sharing

Convolution

Fully connected

Figure 9.14

(Goodfellow 2016)

Efficiency of Convolution

Input size: 320 by 280

Kernel size: 2 by 1

Output size: 319 by 280

	Convolution	Dense matrix	Sparse matrix
Stored floats	2	319*280*320*280 $>$ 8e9	2*319*280 = 178,640
Float muls or adds	319*280*3 = 267,960	$>$ 16e9	Same as convolution (267,960)

(Goodfellow 2016)

Gabor Functions

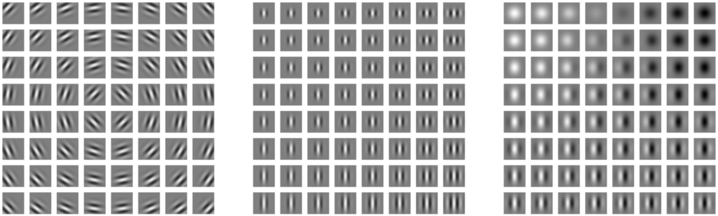


Figure 9.18

(Goodfellow 2016)

- Primary visual cortex (V1), where our brain begins to process visual input, can be described by Gabor functions.

Gabor-like Learned Kernels

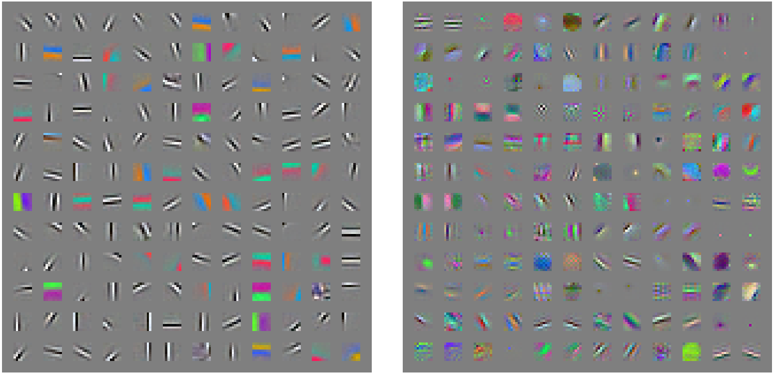


Figure 9.19

(Goodfellow 2016)

Outline

Convolution

Pooling

Convolutional Network Components

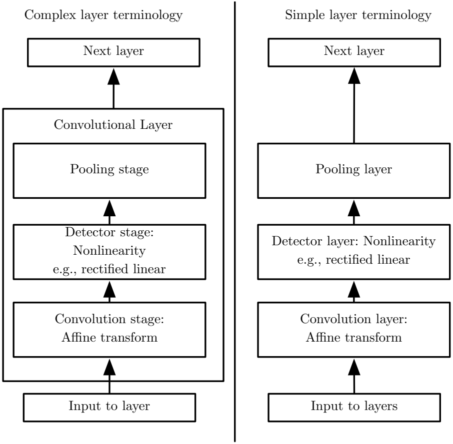


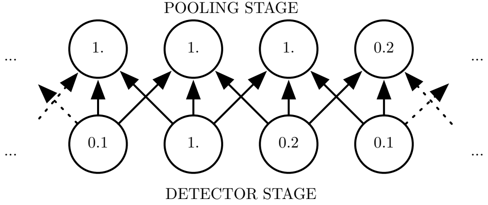
Figure 9.7

(Goodfellow 2016)

Pooling

- Summarize statistic of the nearby outputs.
  - Max pooling
  - Average pooling
  - Help to learn representations that are invariant to small translations, e.g. noises on where an eye is located.
- Pooling may be applied beyond convolution to learn invariant from a group of inputs.
  - E.g. to learn representations invariant to rotation and scaling from different rotations and scalings of input.
- Usually reduce input sizes to speed up following layers.

Max Pooling and Invariance to Translation



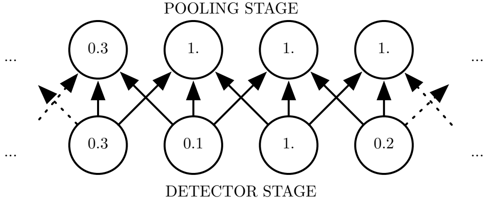


Figure 9.8

(Goodfellow 2016)

Cross-Channel Pooling and Invariance to Learned Transformations

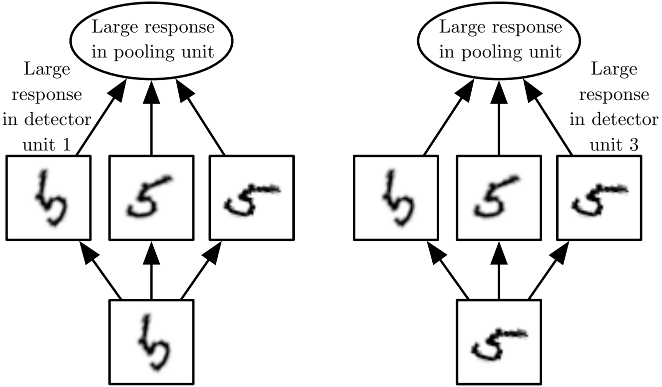


Figure 9.9

(Goodfellow 2016)

Pooling with Downsampling

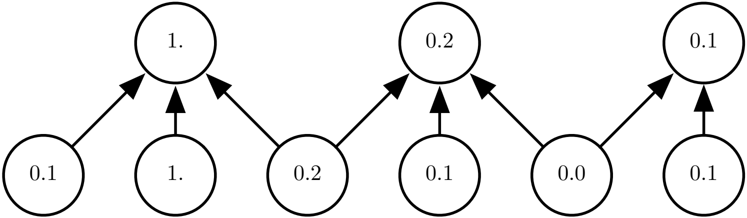


Figure 9.10

(Goodfellow 2016)

Example Classification Architectures

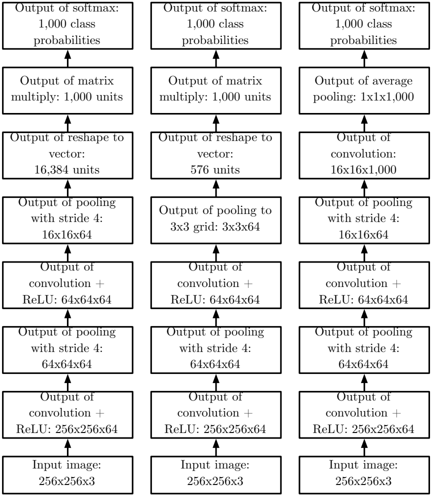


Figure 9.11

(Goodfellow 2016)

Summary

- CNNs apply the same kernel over many parts of the input.
- Pooling help to summarize statistic of the nearby outputs.