Introduction to Deep Learning

Arijit Mondal
Dept. of Computer Science & Engineering
Indian Institute of Technology Patna
arijit@iitp.ac.in

Convolutional Neural Network

Introduction

- Specialized neural network for processing data that has grid like topology
  - Time series data (one dimensional)
  - Image (two dimensional)
- Found to be reasonably suitable for certain class of problems eg. computer vision
- Instead of matrix multiplication, it uses convolution in at least one of the layers

Convolution operation

- Consider the scenario of locating a spaceship with a laser sensor
- Suppose, the sensor is noisy
  - Accurate estimation is not possible
- Weighted average of location can provide a good estimate $s(t) = \int x(a)w(t-a)da$
  - $x(a)$ — Location at age a by the sensor, t — current time, w — weight
  - This is known as convolution
  - Usually denoted as $s(t) = (x * w)(t)$
- In neural network terminology x is input, w is kernel and output is referred as feature map

Convolution operation (contd)

- Discrete convolution can be represented as

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

- In neural network input is multidimensional and so is kernel
  - These will be referred as tensor
- Two dimensional convolution can be defined as

$$s(i,j) = (I * K)(i,j) = \sum_{m,n} I(m,n)k(i-m, j-n) = \sum_{m,n} I(i-m, j-n)k(m,n)$$

  - Commutative
- In many neural network, it implements as cross-correlation

$$s(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I(i+m, j+n)k(m,n)$$

  - No kernel flip is possible

2D convolution

a	b	c	d
e	f	g	h
i	j	k	l

w	x
y	z

aw+bx +ey+fz	bw+cx +fy+gz	cw+dx +gy+hz
ew+fx +iy+jz	fw+gx +jy+kz	gw+hx +ky+lz

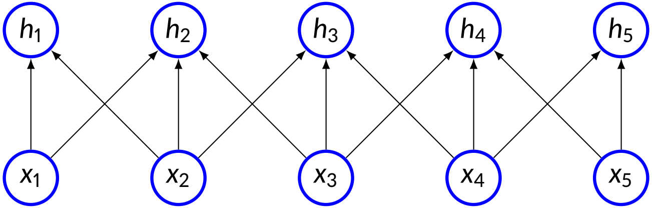
Advantages

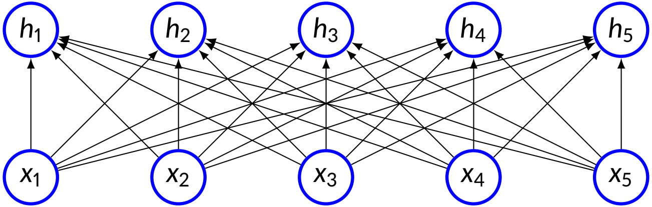
- Convolution can exploit the following properties
  - Sparse interaction (Also known as sparse connectivity or sparse weights)
  - Parameter sharing
  - Equivariant representation

Sparse interaction

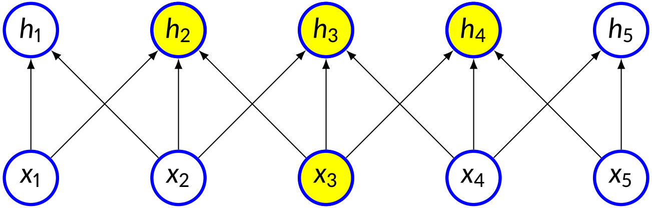
- Traditional neural network layers use matrix multiplication to describe how outputs and inputs are related
- Convolution uses a smaller kernel
  - Significant reduction in number of parameters
  - Computing output require few comparison
- For example, if there is m inputs and n outputs, traditional neural network will require $m \times n$ parameters
- If each of the output is connected to at most k units, the number of parameters will be $k \times n$

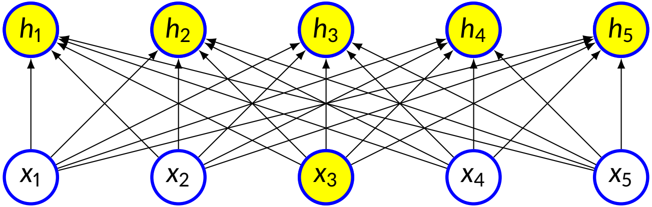
Sparse connectivity



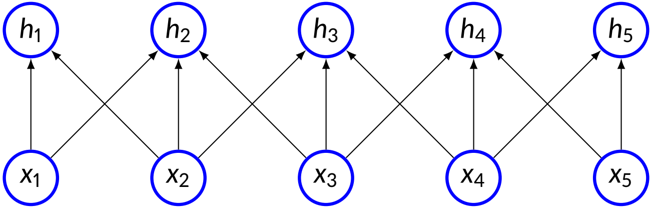


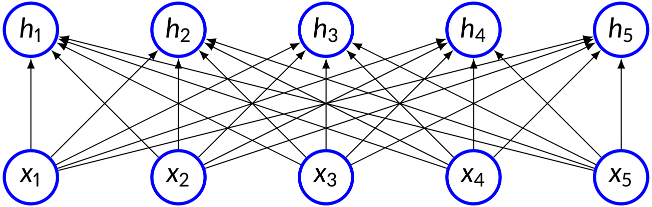
Sparse connectivity



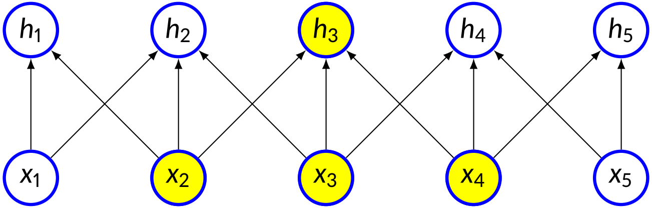


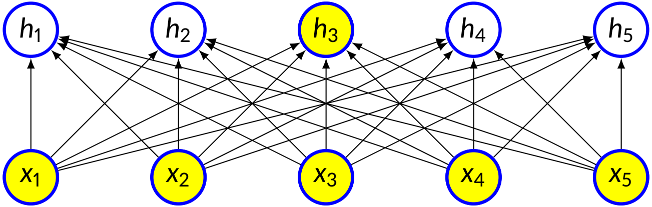
Sparse connectivity



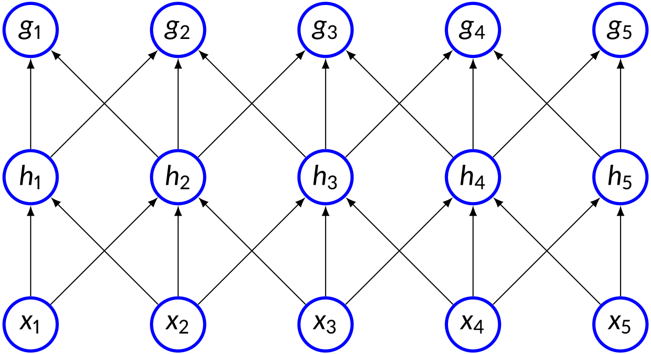


Sparse connectivity

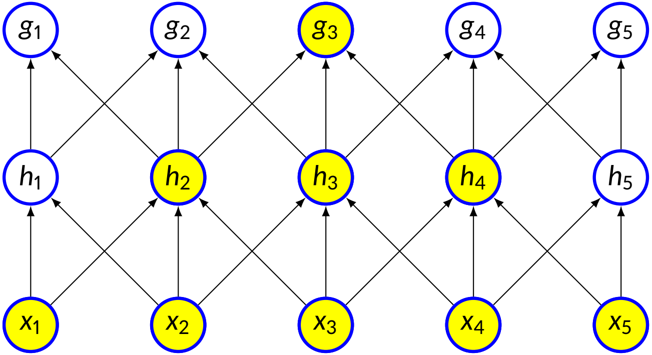




Receptive field



Receptive field



Parameter sharing

- Same parameters are used for more than one function model
- In tradition neural network, weight is used only once
- Each member of kernel is used at every position of the inputs
- As $k \ll m$, the number of parameters will reduced significantly
- Also, require less memory

Edge detection

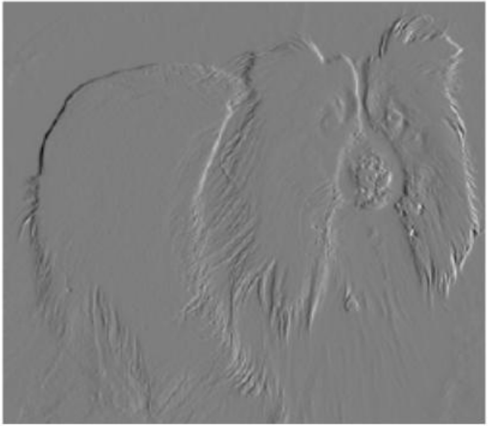


Image source: Deep Learning Book

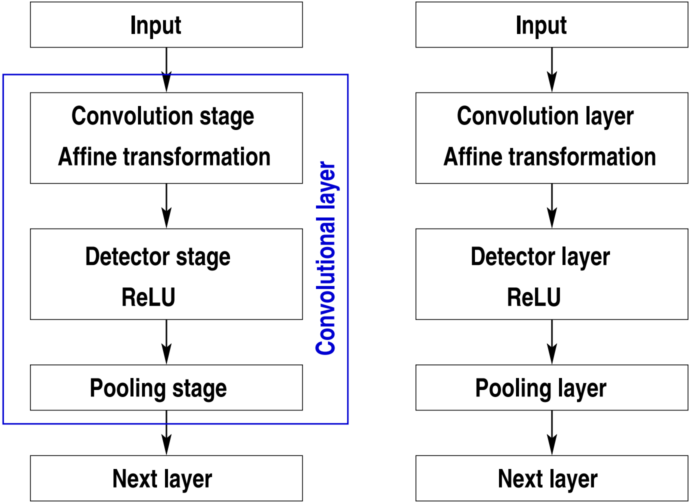
Equivariance

- If the input changes, the output changes in the same
- Specifically, a function $f(x)$ is equivariant to function g if $f(g(x)) = g(f(x))$
  - Example, g is a linear translation
  - Let B be a function giving image brightness at some integer coordinates and g be a function mapping from one image to another image function such that $I' = g(I)$ with $I'(x, y) = I(x-1, y)$
- There are cases sharing of parameters across the entire image is not a good idea

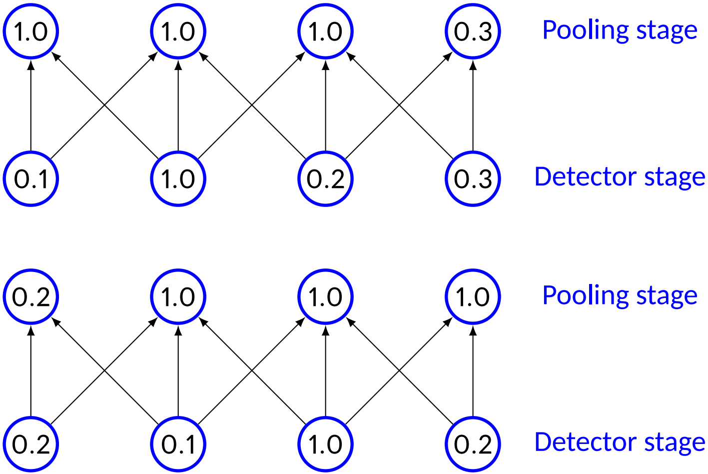
Pooling

- Typical convolutional network has three stages
  - Convolution — several convolution to produce linear activation
  - Detector stage — linear activation runs through the non-linear unit such as ReLU
  - Pooling — Output is updated with a summary of statistics of nearby inputs
    - Maxpooling reports the maximum output within a rectangular neighbourhood
    - Average of rectangular neighbourhood
    - Weighted average using central pixel
- Pooling helps to make representation invariant to small translation
  - Feature is more important than where it is present
- Pooling helps in case of variable size of inputs

Typical CNN


Input
Convolution stage
Affine transformation
Detector stage
ReLU
Pooling stage
Convolutional layer
Next layer
Input
Convolution layer
Affine transformation
Detector layer
ReLU
Pooling layer
Next layer

Invariance of maxpooling



Learned invariances

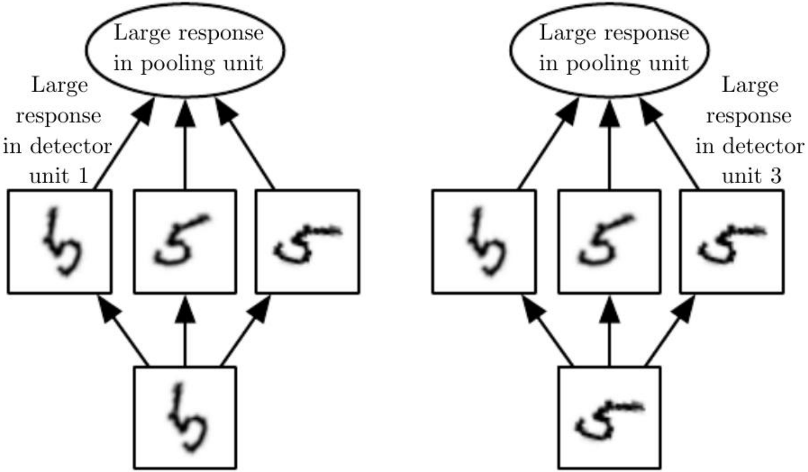
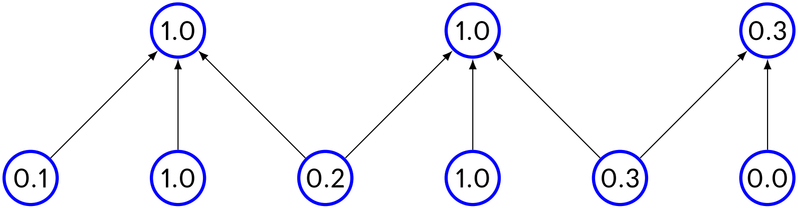
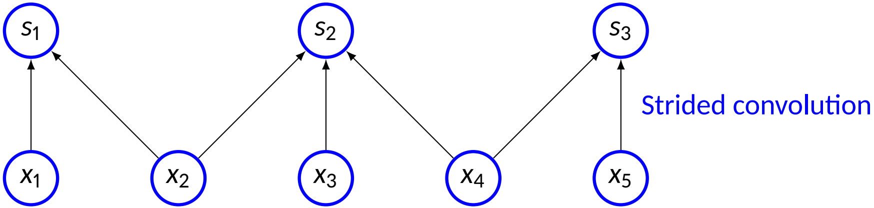


Image source: Deep Learning Book

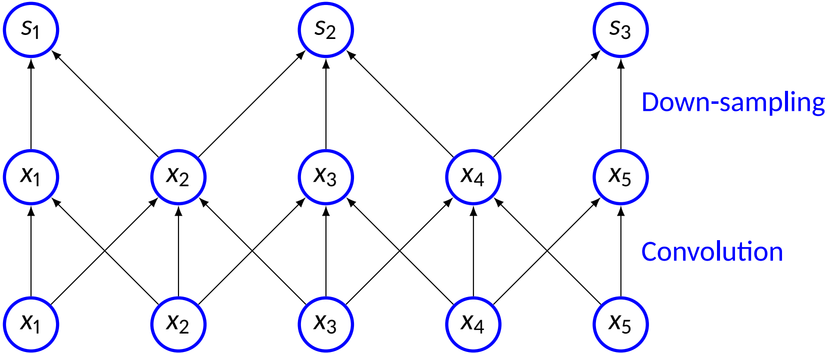
Pooling with downsampling



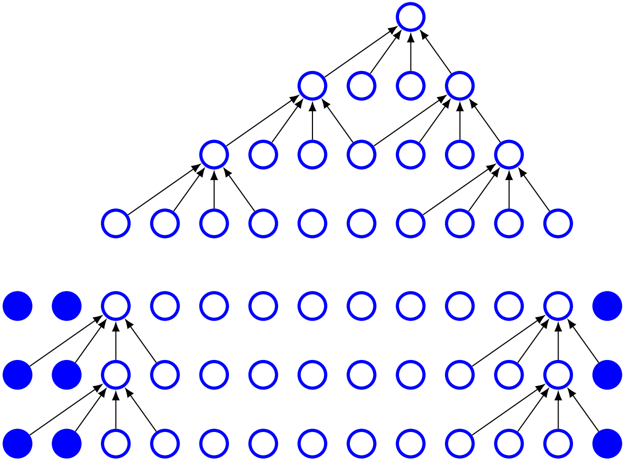
Strided convolution



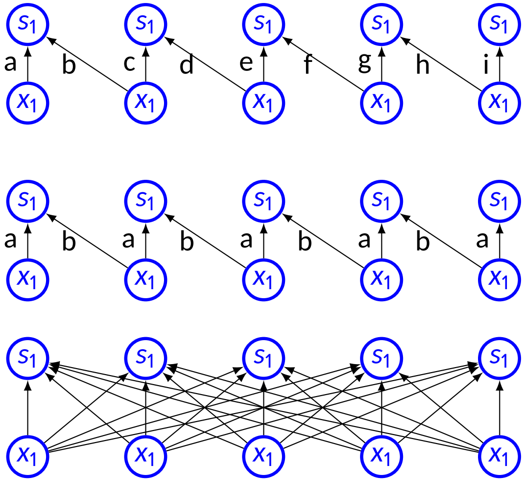
Strided convolution (contd)



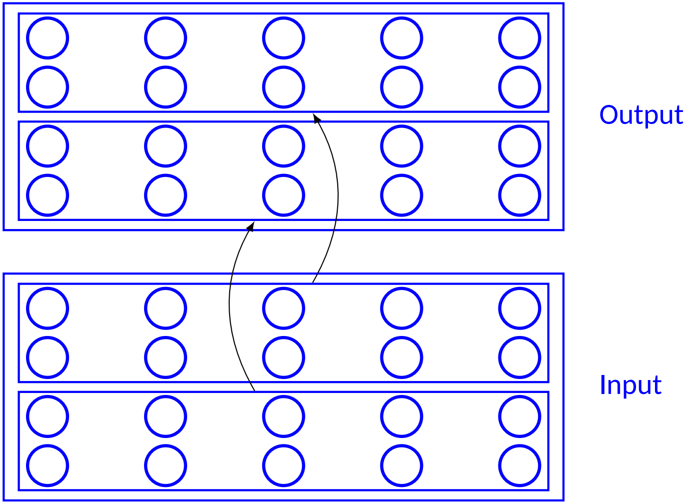
Zero padding



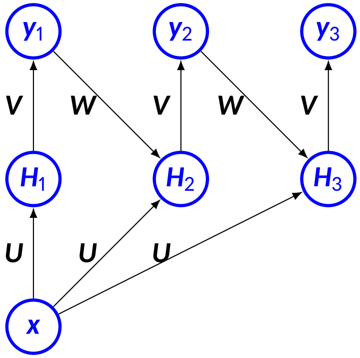
Connections



Local convolution



Recurrent convolution network



AlexNet

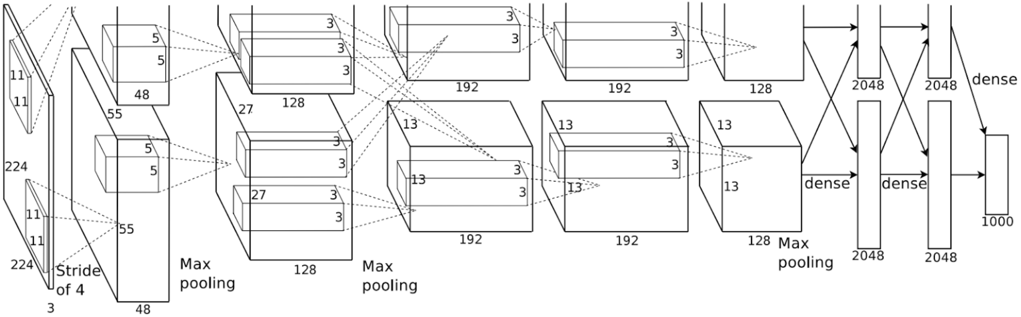


Image source: https://worksheets.codalab.org

GoogleNet

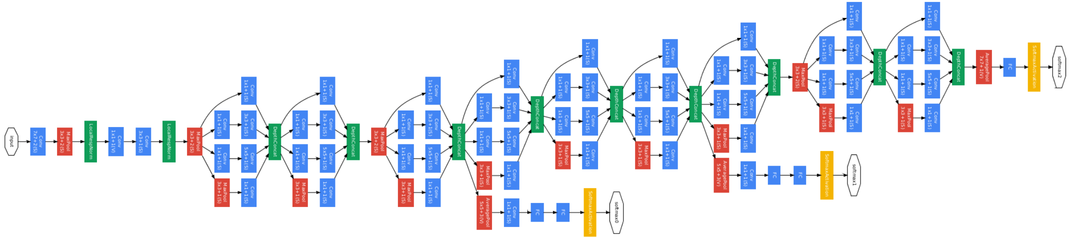
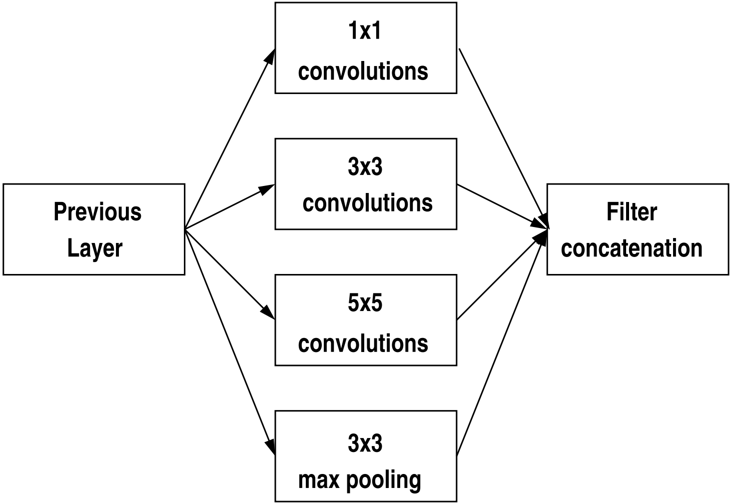
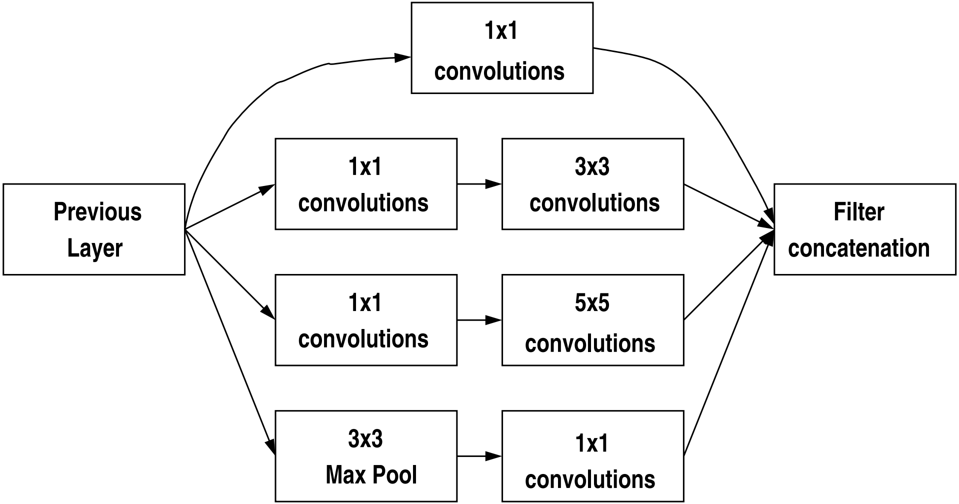


Image source: http://joelouismarino.github.io

Naive inception



Inception


1x1
convolutions
Previous
Layer
1x1
convolutions
3x3
convolutions
Filter
concatenation
1x1
convolutions
5x5
convolutions
3x3
Max Pool
1x1
convolutions