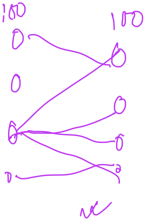
Introduction to Deep Learning

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

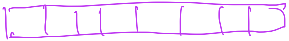
Convolutional Neural Networks

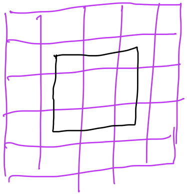


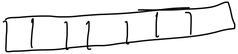
$\boxed{100 \times 100}$

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



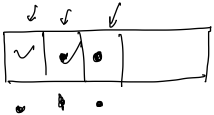




Convolution operation

- 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

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

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

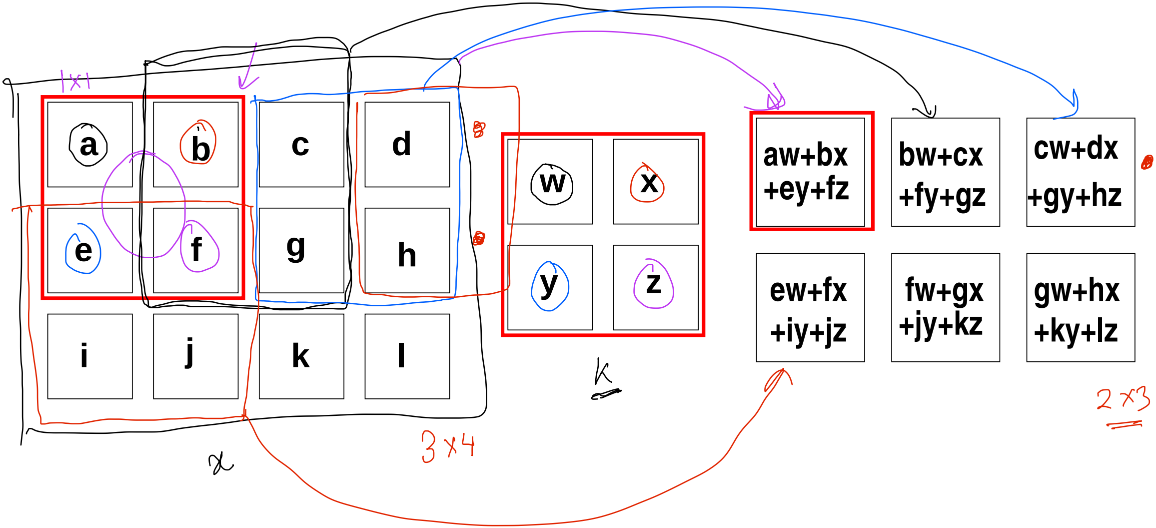
$$s(i,j) = (I * K)(i,j) = \sum_{m,n} I(m,n)k(i-m,j-n) = \sum_{m,n} I(i-m,j-n)k(m,n)$$

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

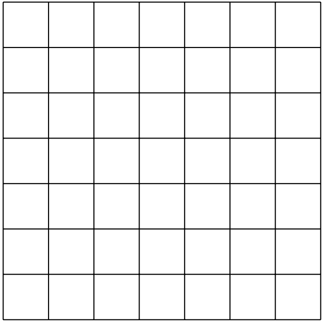
$$s(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(i+m,j+n)k(m,n)$$

  - No kernel flip is possible

2D convolution

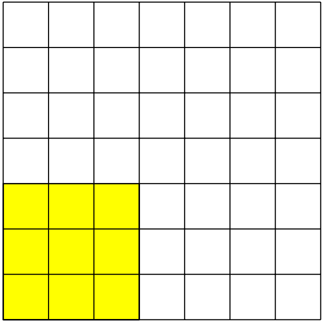


2D Convolution



Grid size: 7×7

2D Convolution

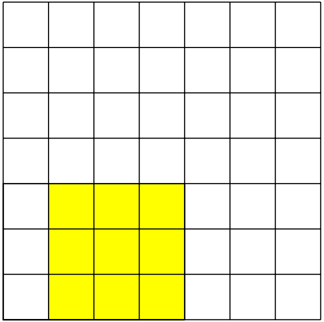


Grid size: 7×7

Filter size: 3×3

Stride: 1

2D Convolution

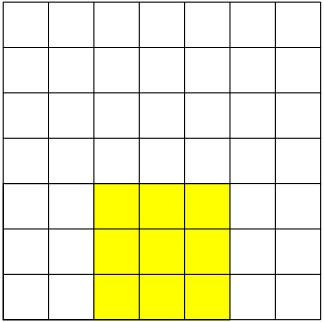


Grid size: 7×7

Filter size: 3×3

Stride: 1

2D Convolution

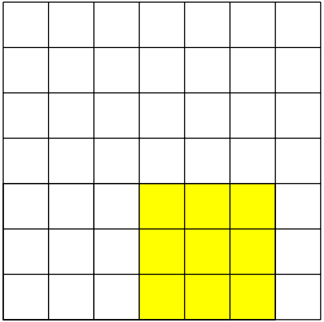


Grid size: 7×7

Filter size: 3×3

Stride: 1

2D Convolution

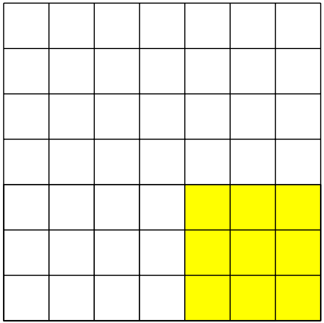


Grid size: 7×7

Filter size: 3×3

Stride: 1

2D Convolution

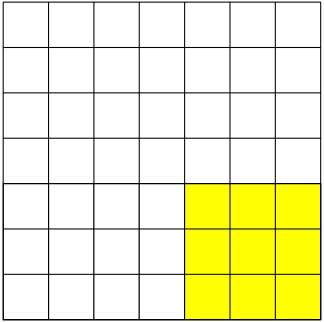


Grid size: 7×7

Filter size: 3×3

Stride: 1

2D Convolution



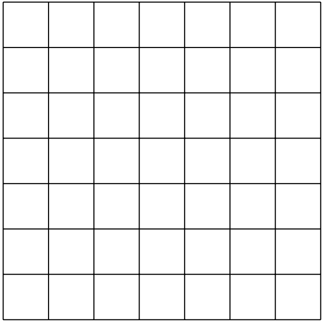
Grid size: 7×7

Filter size: 3×3

Stride: 1

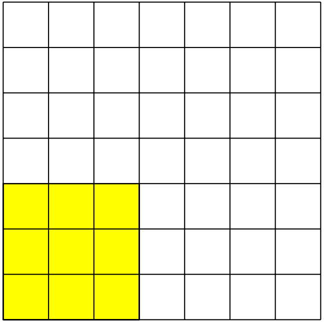
Output size: 5×5

2D convolution with stride



Grid size: 7×7

2D convolution with stride

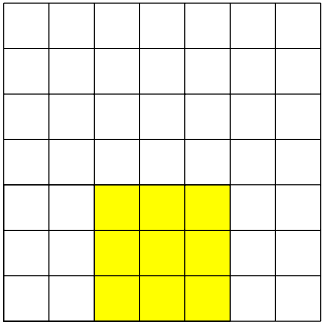


Grid size: 7×7

Filter size: 3×3

Stride: 2

2D convolution with stride

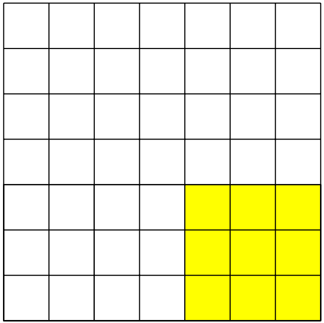


Grid size: 7×7

Filter size: 3×3

Stride: 2

2D convolution with stride

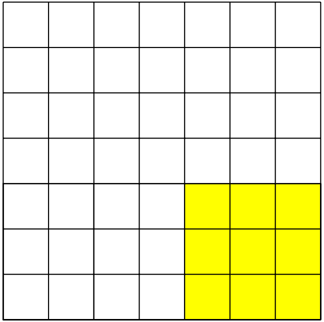


Grid size: 7×7

Filter size: 3×3

Stride: 2

2D convolution with stride



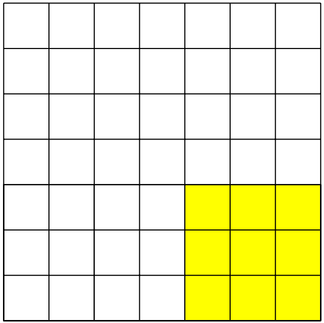
Grid size: 7×7

Filter size: 3×3

Stride: 2

Output size: 3×3

2D convolution with stride



Grid size: 7×7

Filter size: $\boxed{3 \times 3}$

Stride: 2

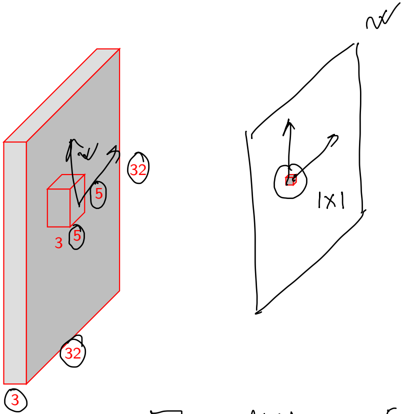
Output size: $\boxed{3 \times 3} \rightarrow \frac{7 \times 7}{7}$

Output size: $(N - F)/S + 1$

N - input size, F - Filter size,

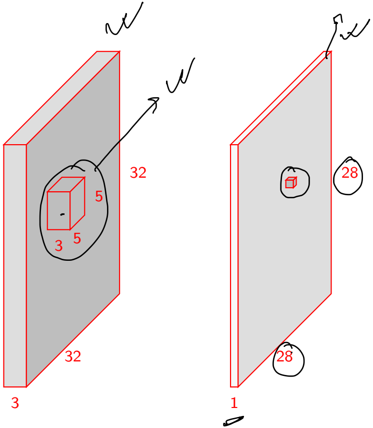
S - Stride

Convolution operation

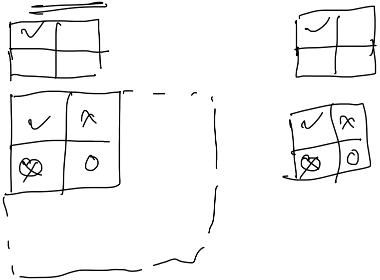


$32 \times 32 \times \boxed{3}$, filter - $\underline{5 \times 5} \times 3$

Convolution operation



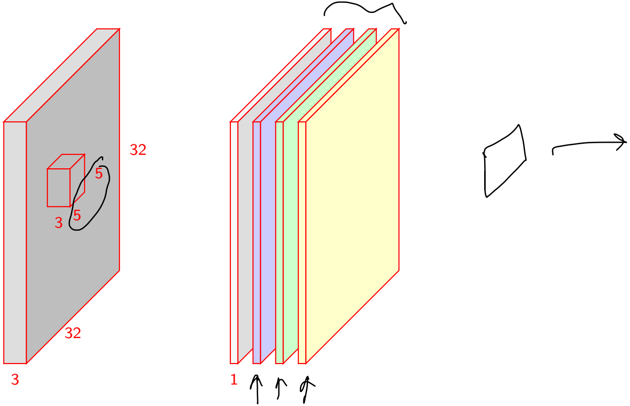
5x5x3



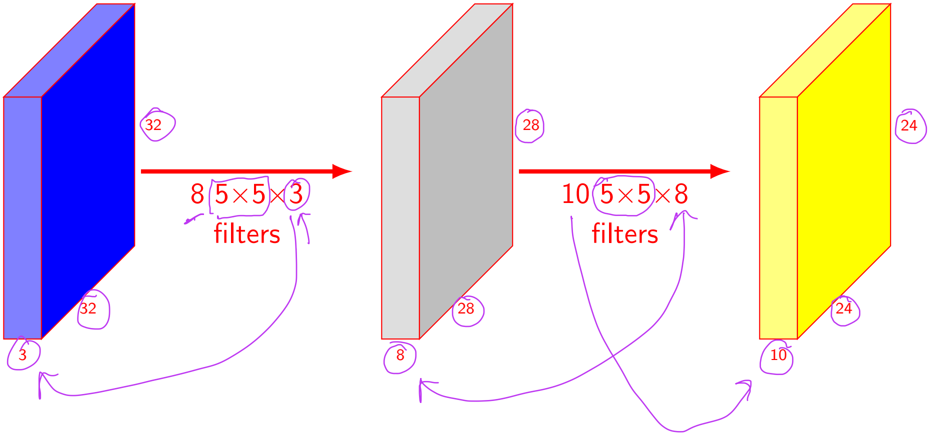
75 + bias = 76

5x5x3 x 28x28

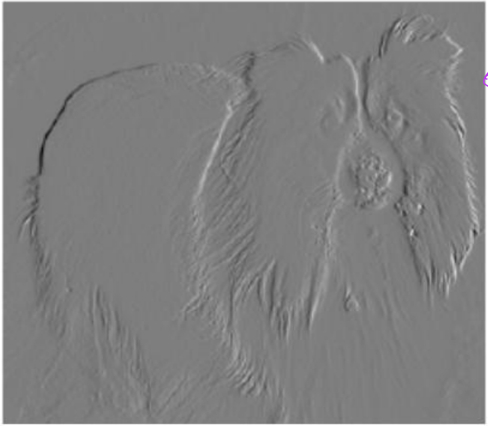
Convolution operation



Convolution example



Edge detection

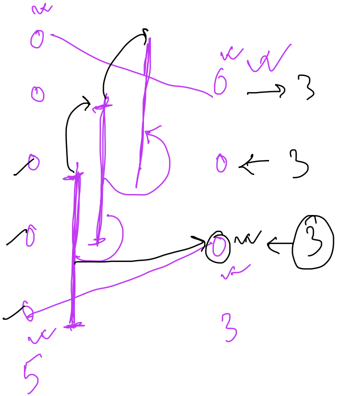


$W = [-1 \quad 1]$

Image source: Deep Learning Book

Advantages

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



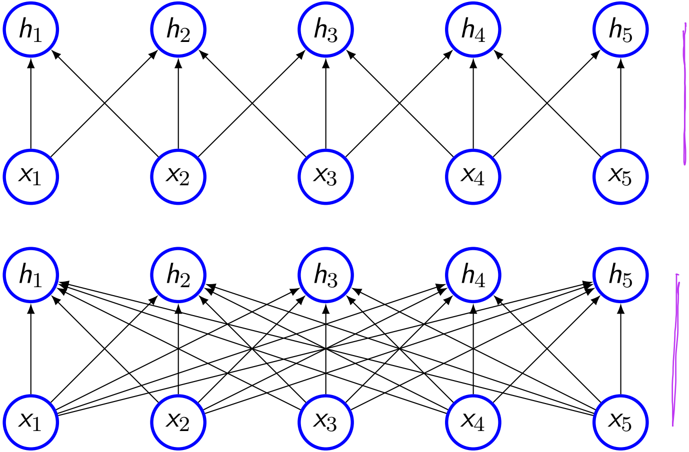
$\boxed{5 \times 3}$

9

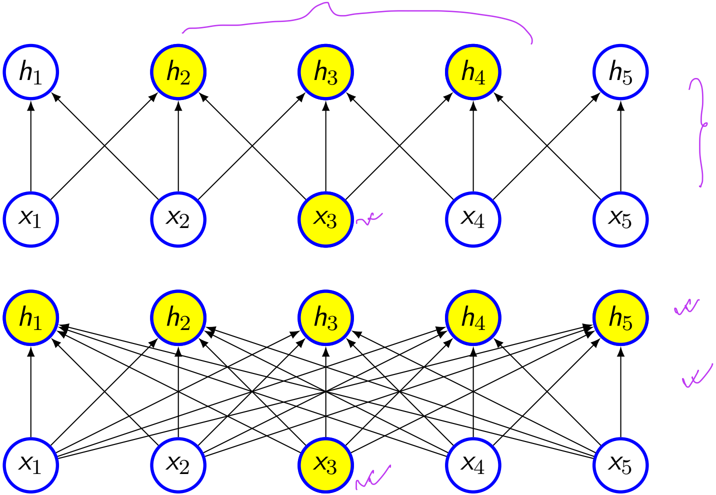
Sparse interaction

- 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 $\boxed{m \times n}$ parameters
- If each of the output is connected to at most k units, the number of parameters will be $\boxed{k \times n}$

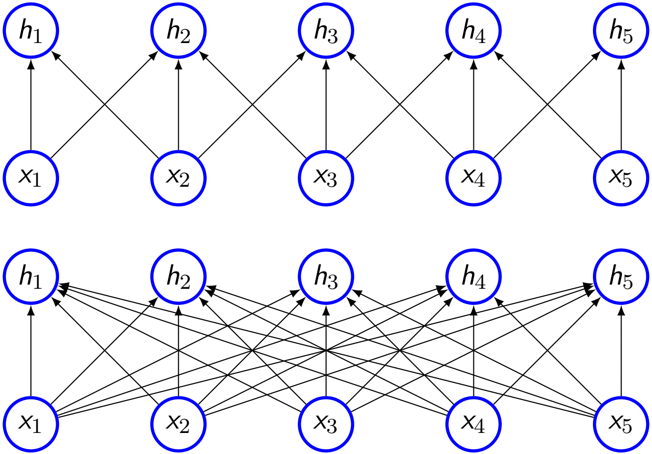
Sparse connectivity



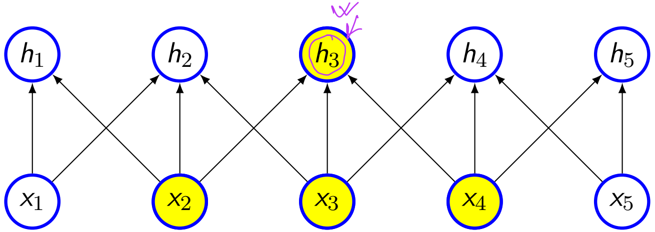
Sparse connectivity

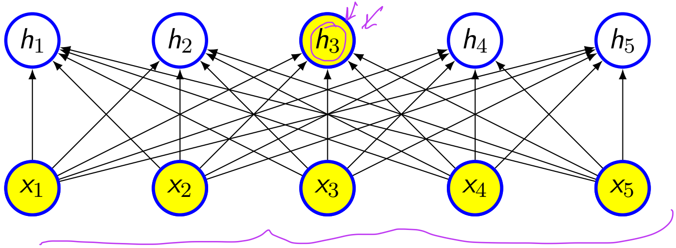


Sparse connectivity

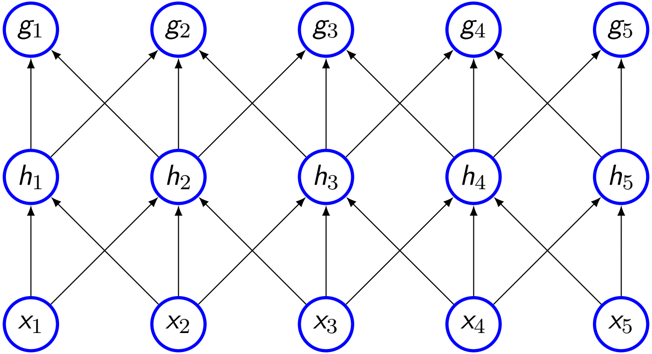


Sparse connectivity

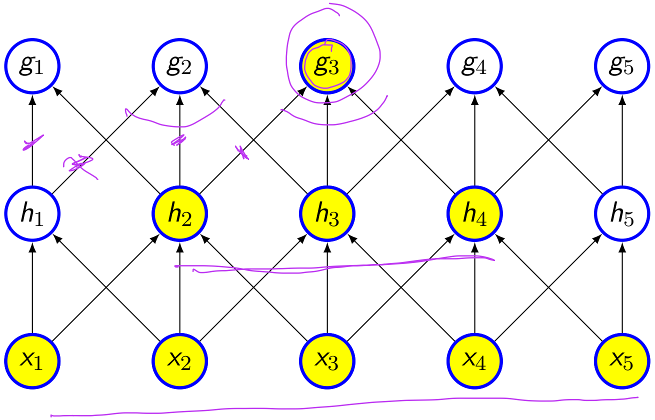




Receptive field



Receptive field

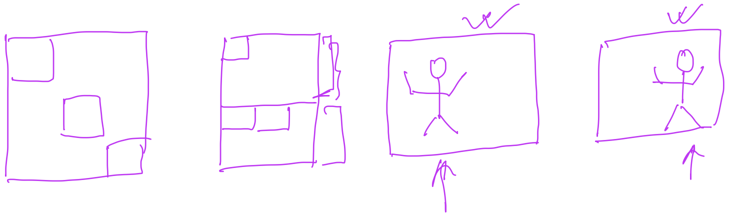


Parameter sharing

- 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

Equivariance

- If the input changes, the output changes in the same way
- 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



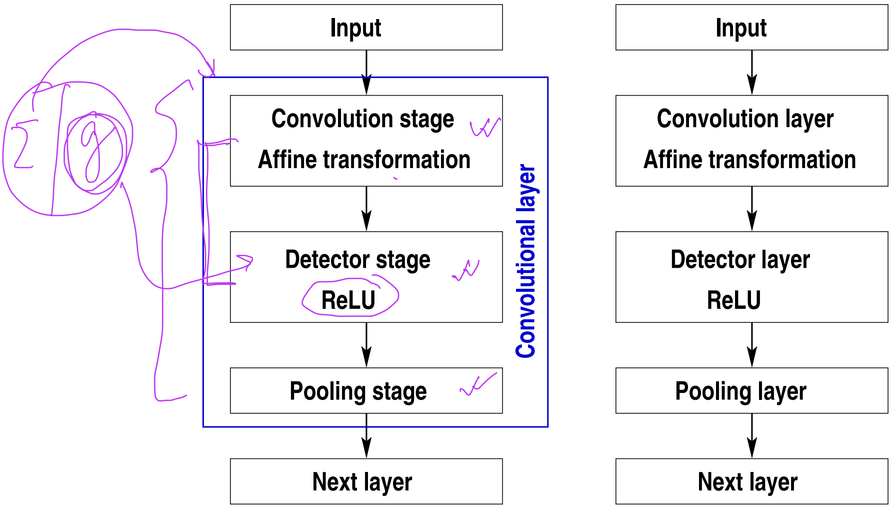
Pooling

- 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

$\begin{bmatrix}1 & 8\\ 4 & 6\end{bmatrix} \rightarrow 8$

$\rightarrow$ Max

Typical CNN



CDP

CDCDP

$ax+b$

$y = ax_1 + bx_2 + \ldots$

Max Pool

$k = \underline{2 \times 2}$, stride = 2

0	4	7	8
9	2	4	5
6	7	3	4
8	2	1	5

9 7

9	8
8	5

8 7

Max Pool

0	4	7	8
9	2	4	5
6	7	3	4
8	2	1	5

8	

Max Pool

0	4	7	8
9	2	4	5
6	7	3	4
8	2	1	5

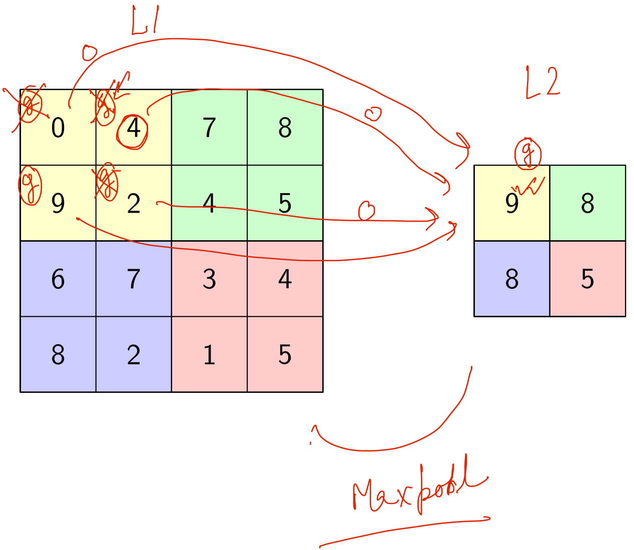
8	5

Max Pool

0	4	7	8
9	2	4	5
6	7	3	4
8	2	1	5

9	
8	5

Max Pool



Weight parameters ?

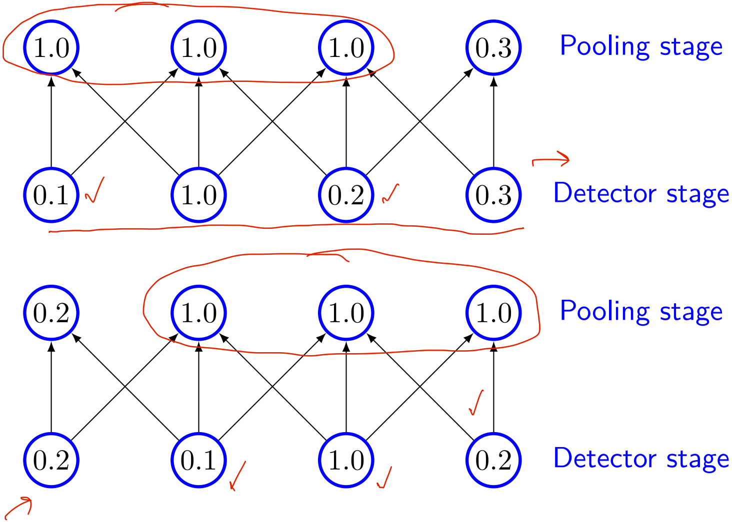
L3 0

L

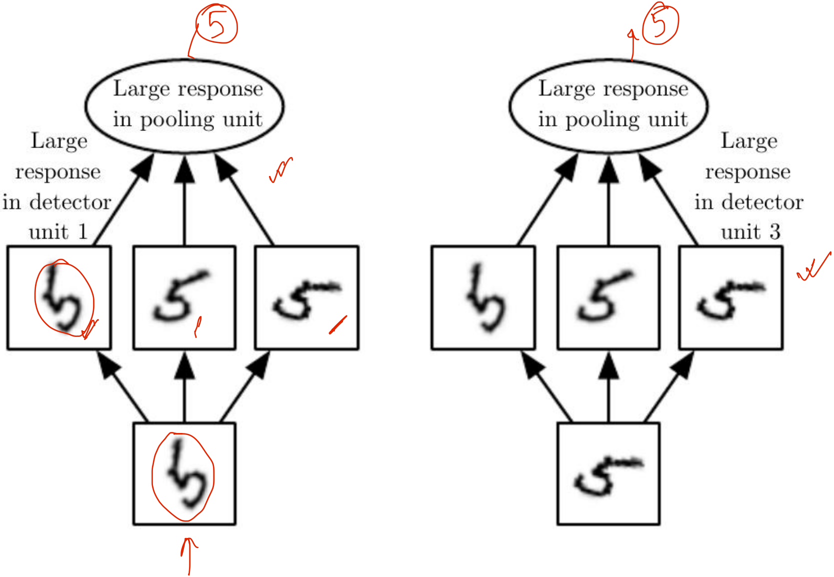
g

Minibatch

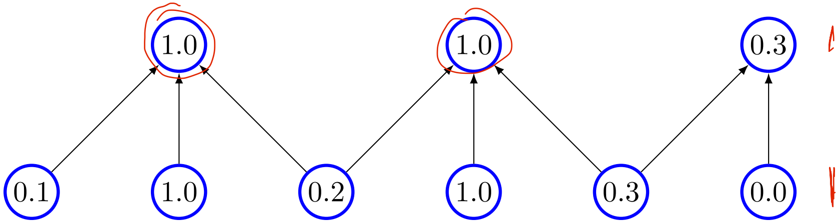
Invariance of maxpooling



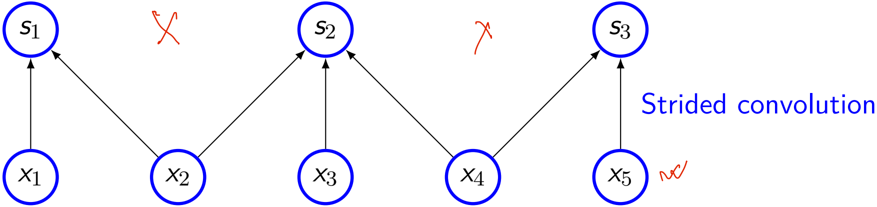
Learned invariances



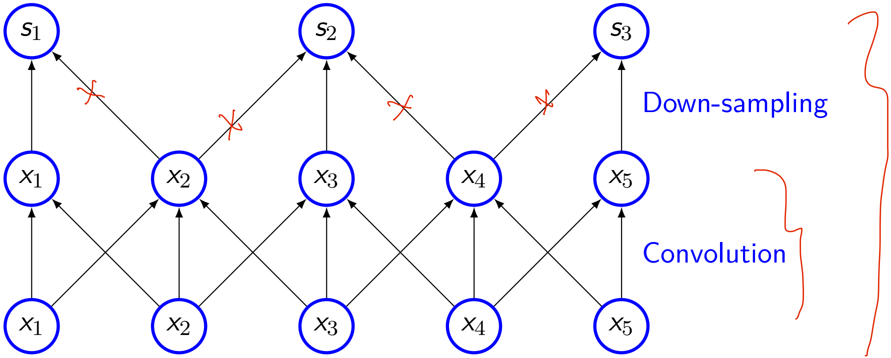
Pooling with downsampling



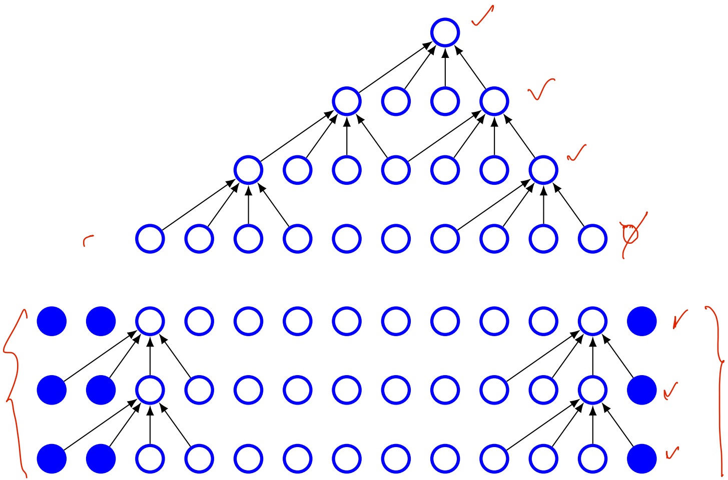
Strided convolution



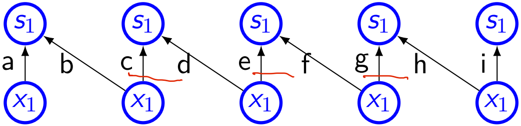
Strided convolution (contd)



Zero padding

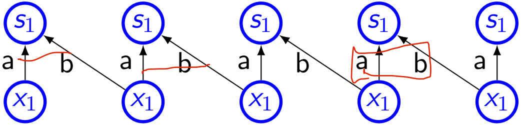


Connections

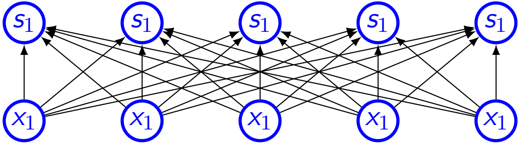


c = a, d = b

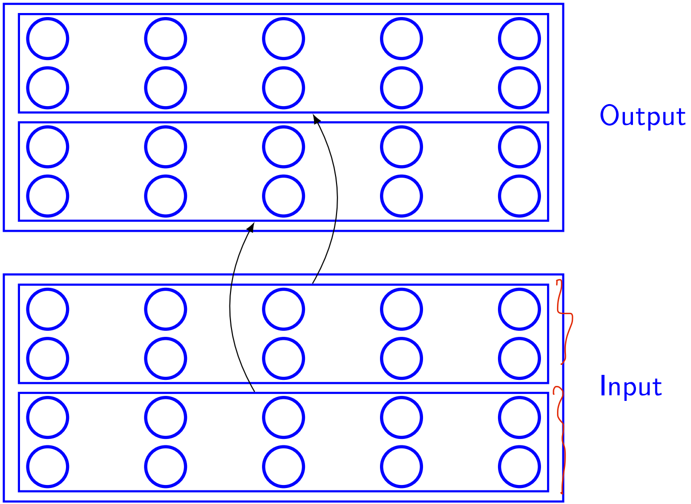
CNN



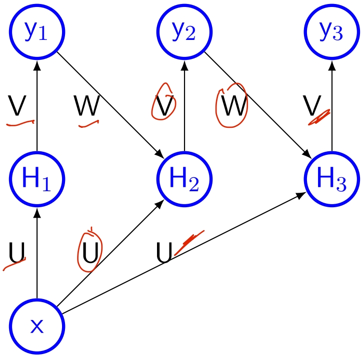
FC



Local convolution



Recurrent convolution network



AlexNet

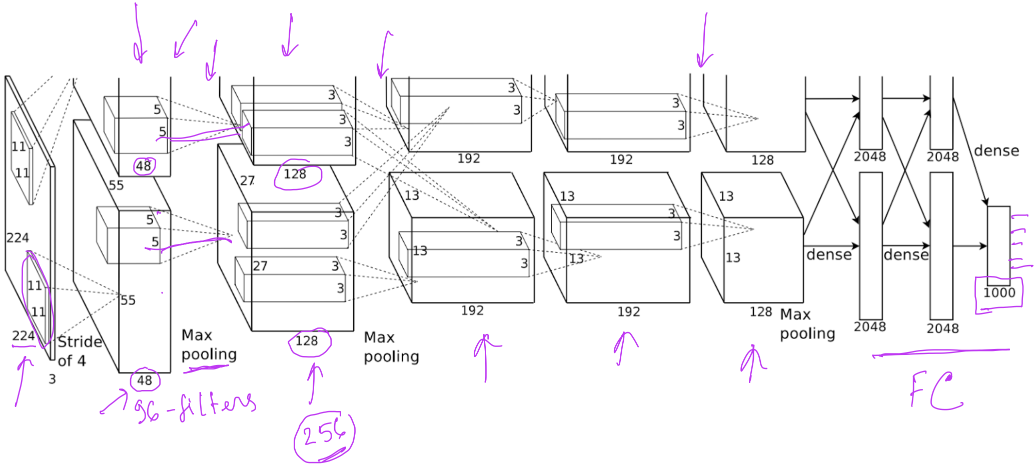


Image source: https://worksheets.codalab.org

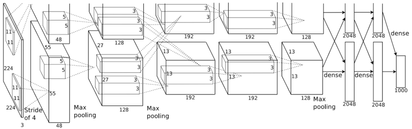
AlexNet

$96 \times 11 \times 11 \times 3$

$\rightarrow 58 \times 10^6$

$\cancel{16 \times 10^9}$

3.7×10^6



- Architecture

- INPUT - $227 \times 227 \times 3$
- CONV1 - 96 11×11 filters at stride 4, pad 0, Output: $55 \times 55 \times 96$
- MAX POOL1 - 3×3 filter, stride 2 Output: $27 \times 27 \times 96$
- NORM1 - Output: $27 \times 27 \times 96$
- CONV2 - 256 5×5 filters at stride 1, pad 2, Output: $27 \times 27 \times 256$
- MAX POOL2 - 3×3 filter, stride 2 Output: $13 \times 13 \times 256$
- NORM2 - O $13 \times 13 \times 256$
- CONV3 - 384 3×3 filter, stride 1, pad 1, Output: $13 \times 13 \times 384$
- CONV4 - 384 3×3 filter, stride 1, pad 1, Output: $13 \times 13 \times 384$
- CONV5 - 256 3×3 filter, stride 1, pad 1, Output: O $13 \times 13 \times 256$
- MAX POOL3 - 3×3 filter, stride 2, Output: $6 \times 6 \times 256$
- FC6 - 4096 Neurons
- FC7 - 4096 Neurons
- FC8 - 1000 Neurons

$\rightarrow 6 \times 6 \times 256 \times 4096$

4096×4096

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

VggNet

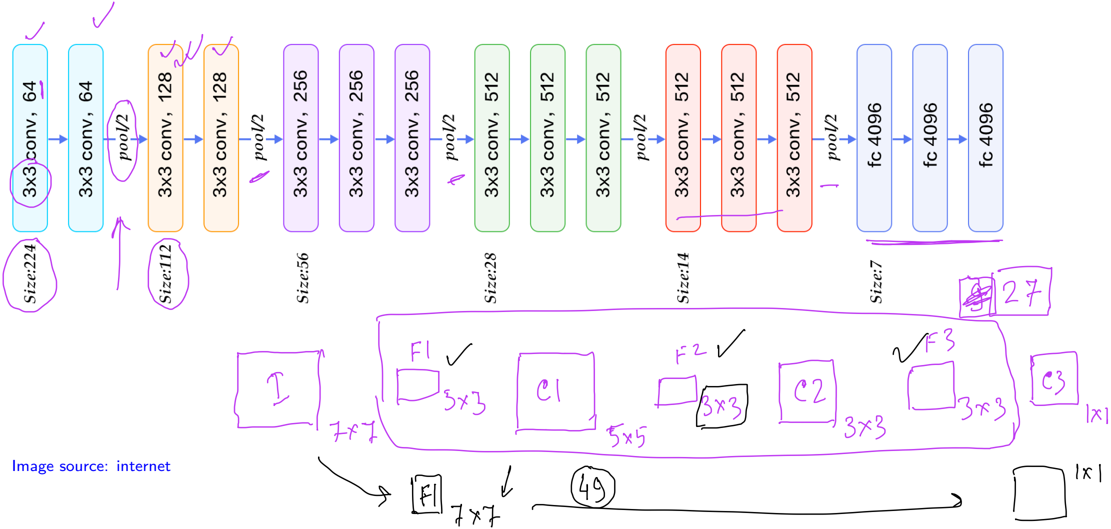


Image source: internet

GoogleNet

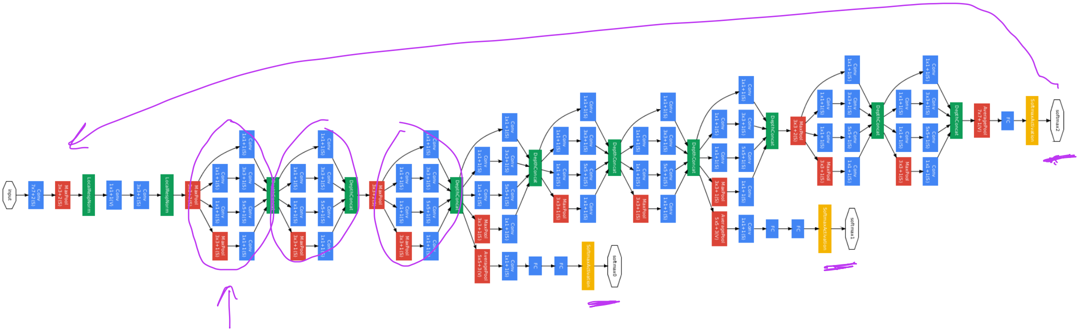
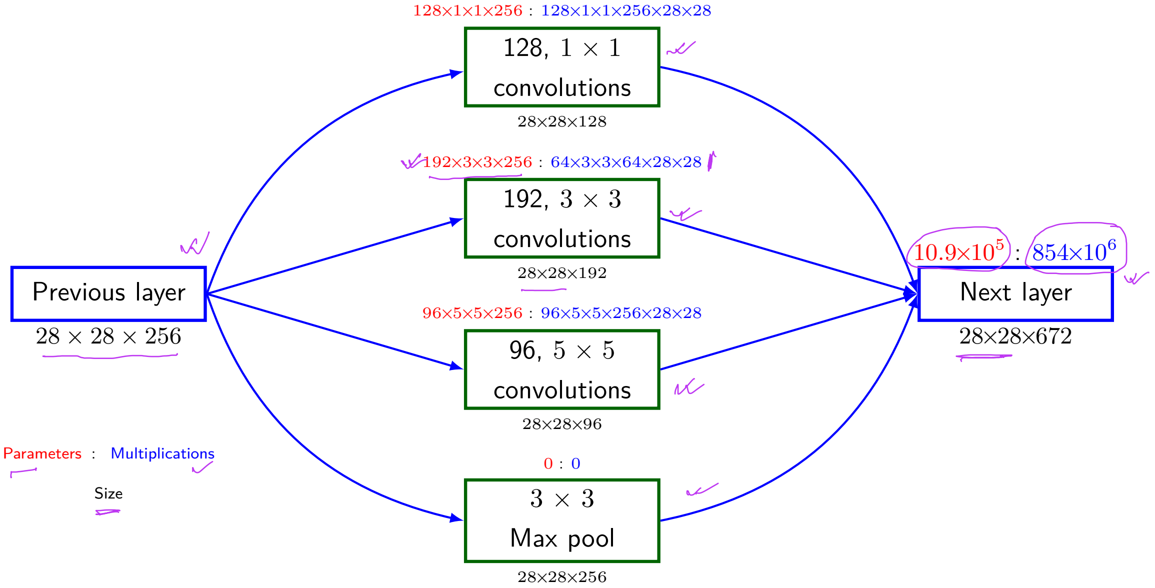
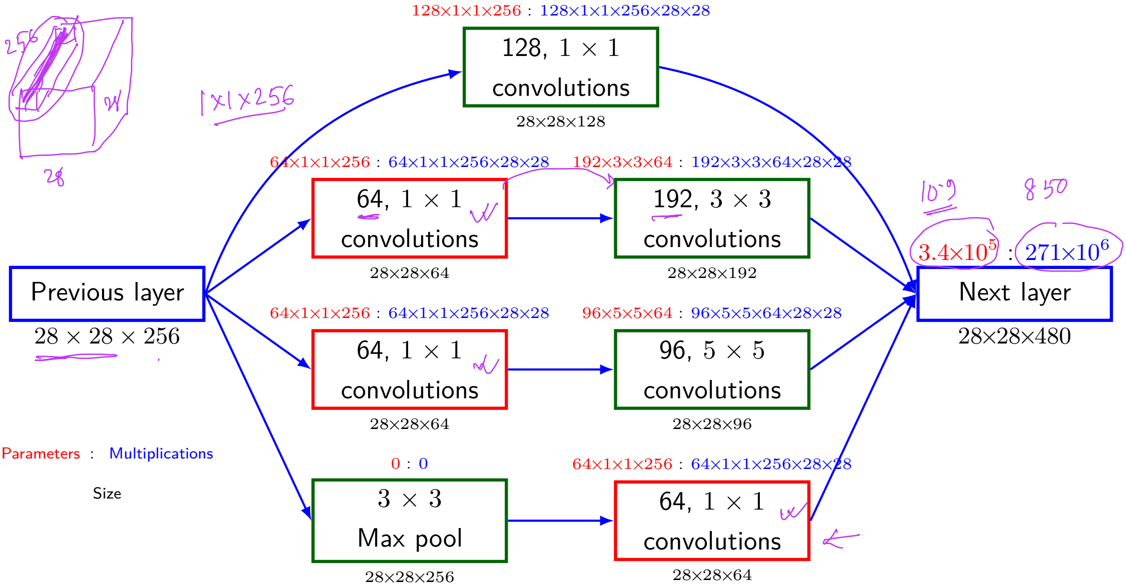


Image source: http://joelouismarino.github.io

Naive inception



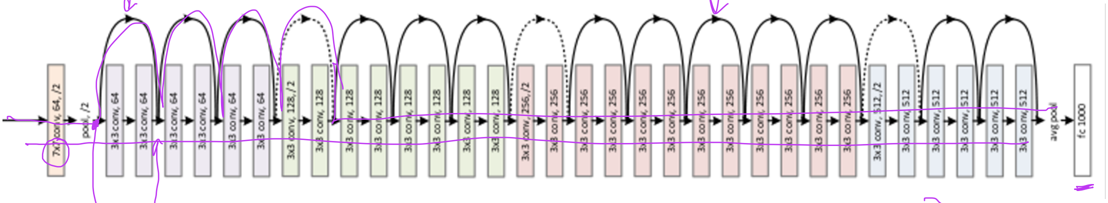
Inception



256
28
28
1×1×256
128×1×1×256 : 128×1×1×256×28×28
128, 1 × 1 convolutions
28×28×128
64×1×1×256 : 64×1×1×256×28×28
64, 1 × 1 convolutions
28×28×64
192×3×3×64 : 192×3×3×64×28×28
192, 3 × 3 convolutions
28×28×192
Previous layer
28 × 28 × 256
64×1×1×256 : 64×1×1×256×28×28
64, 1 × 1 convolutions
28×28×64
96×5×5×64 : 96×5×5×64×28×28
96, 5 × 5 convolutions
28×28×96
0 : 0
3 × 3 Max pool
28×28×256
64×1×1×256 : 64×1×1×256×28×28
64, 1 × 1 convolutions
28×28×64
10^9
850
3.4×10^5 : 271×10^6
Next layer
28×28×480
Parameters : Multiplications
Size

ResNet

skip



$n \rightarrow n'$

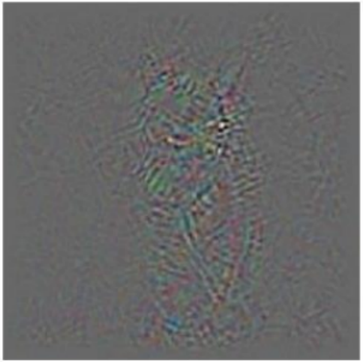
Image source: internet

Comparison of CNN architecture

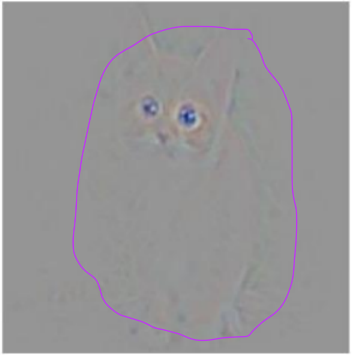
Model	Size (M)	Top-1/top-5 error (%)	# layers	Model description
AlexNet	238	41.00/18.00	8	5 conv + 3 fc layers
VGG-16	540	28.07/9.33	16	13 conv + 3 fc layers
VGG-19	560	27.30/9.00	19	16 conv + 3 fc layers
GoogleNet	40	29.81/10.04	22	21 conv + 1 fc layers
ResNet-50	100	22.85/6.71	50	49 conv + 1 fc layers
ResNet-152	235	21.43/3.57	152	151 conv + 1 fc layers

Image source: internet

Guided backpropagation



Backprop

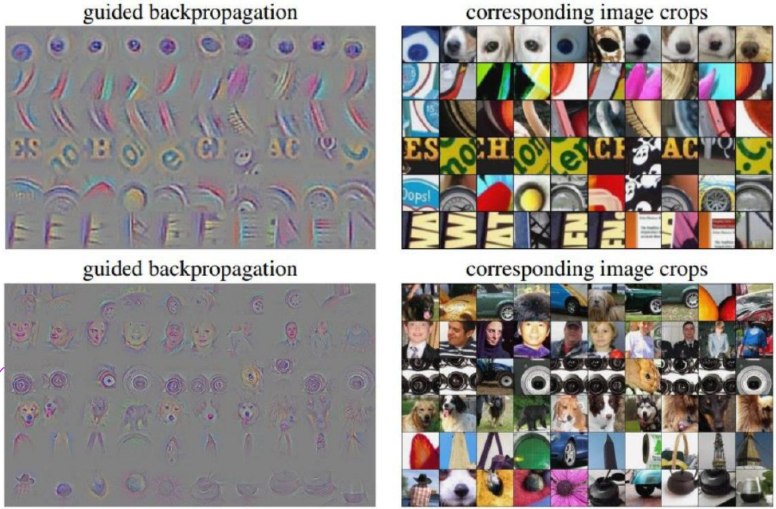


Guided Backprop

$g > 0$

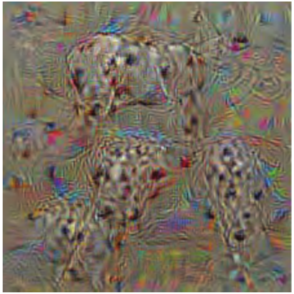
Image source: internet

Guided backpropagation



Fantasy image


cup


dalmatian


goose

$\frac{\partial J}{\partial x_i}$ $\frac{\partial J}{\partial w}$

Deep Dream

softmax

Image source: internet