# CS551: Introduction to Deep Learning

## Convolutional Neural Network

Arijit Mondal

Dept. of Computer Science & Engineering

Indian Institute of Technology Patna

`arijit@iitp.ac.in`

## Introduction

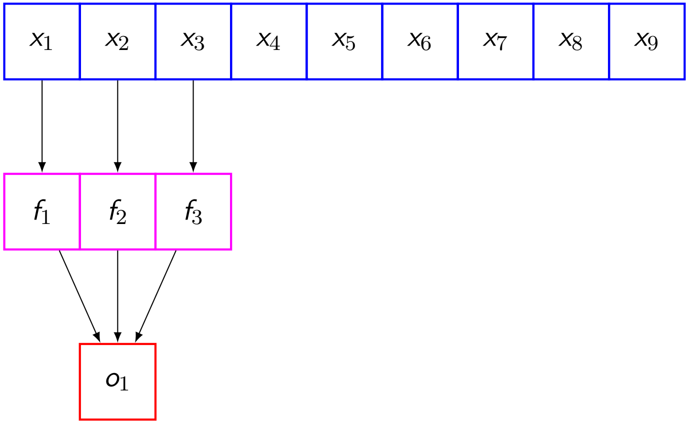
- Specialized neural network for processing data that has grid like topology
  - Time series data (one dimensional)
  - Image (two dimensional)
- Found to be reasonably suitable for certain class of problems eg. computer vision
- Instead of matrix multiplication, it uses convolution in at least one of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

## Convolution operation

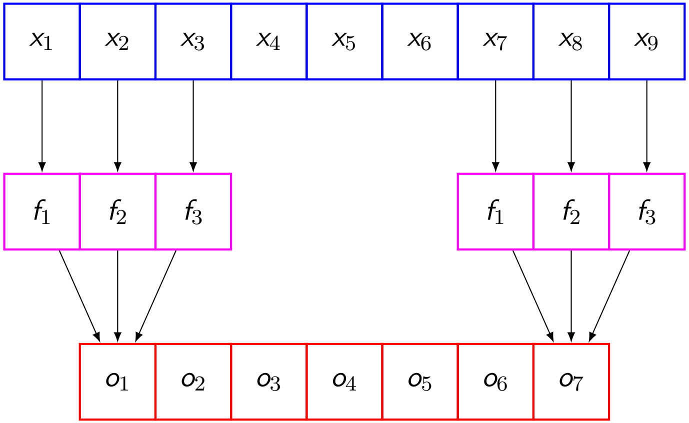
- Consider the scenario of locating a spaceship with a laser sensor
- Suppose, the sensor is noisy
  - Accurate estimation is not possible
- Weighted average of location can provide a good estimate $s(t) = \int x(a)w(t-a)da$
  - $x(a)$ — Location at age $a$ by the sensor, $t$ — current time, $w$ — weight
  - 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
- Discrete convolution can be represented as

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

# Convolution in 1D



# Convolution in 1D



## Convolution in 2D

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

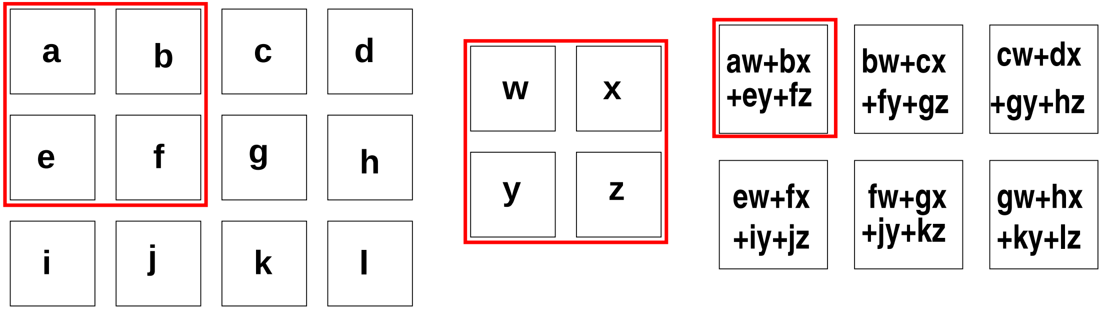
$$s(i,j) = (I * K)(i,j) = \sum_{m,n} I(m,n)k(i-m,j-n) = \sum_{m,n} I(i-m,j-n)k(m,n)$$

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

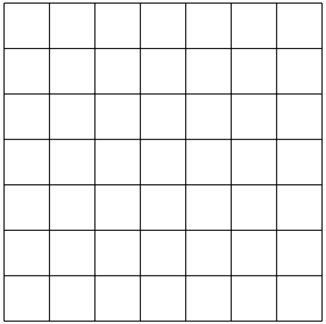
$$s(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I(i+m,j+n)k(m,n)$$

  - No kernel flip is possible

## 2D convolution

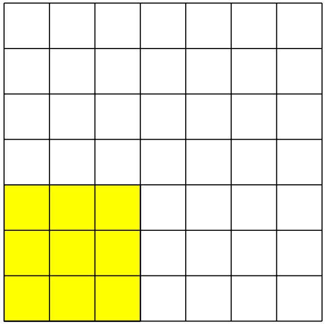


# 2D Convolution



Grid size: $7 \times 7$

# 2D Convolution

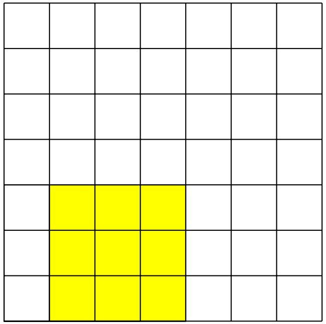


Grid size: $7 \times 7$

Filter size: $3 \times 3$

Stride: $1$

# 2D Convolution

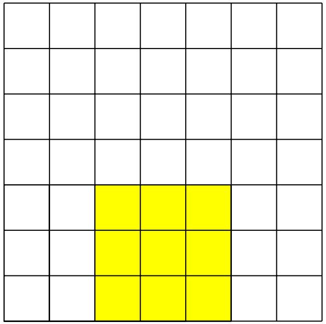


Grid size: $7 \times 7$

Filter size: $3 \times 3$

Stride: $1$

# 2D Convolution

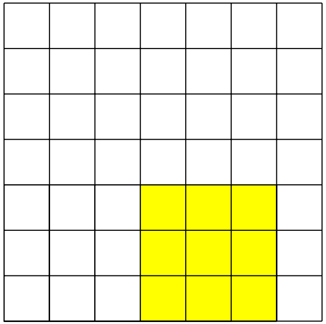


Grid size: $7 \times 7$

Filter size: $3 \times 3$

Stride: $1$

# 2D Convolution

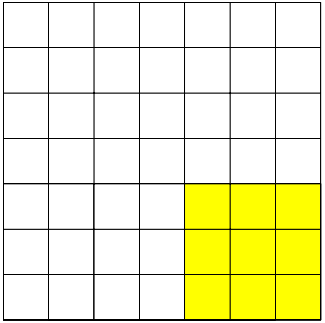


Grid size: $7 \times 7$

Filter size: $3 \times 3$

Stride: $1$

# 2D Convolution

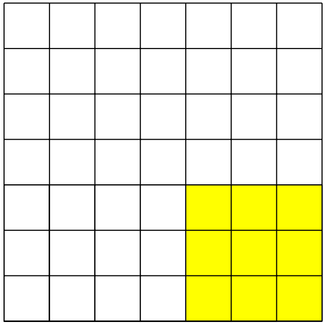


Grid size: $7 \times 7$

Filter size: $3 \times 3$

Stride: $1$

# 2D Convolution



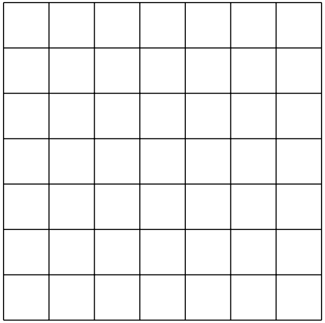
Grid size: $7 \times 7$

Filter size: $3 \times 3$

Stride: $1$

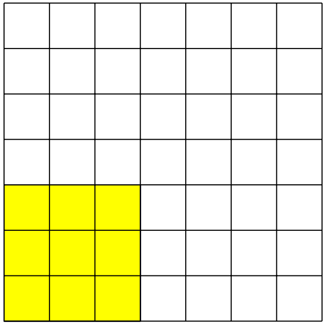
Output size: $5 \times 5$

# 2D convolution with stride



Grid size: $7 \times 7$

# 2D convolution with stride

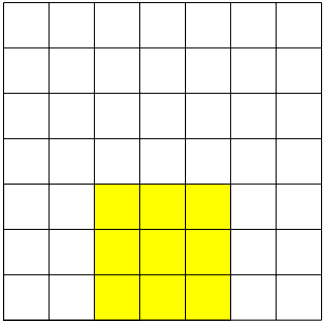


Grid size: $7 \times 7$

Filter size: $3 \times 3$

Stride: $2$

# 2D convolution with stride

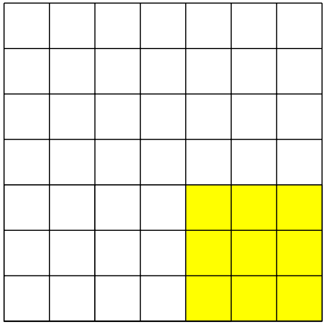


Grid size: $7 \times 7$

Filter size: $3 \times 3$

Stride: $2$

# 2D convolution with stride

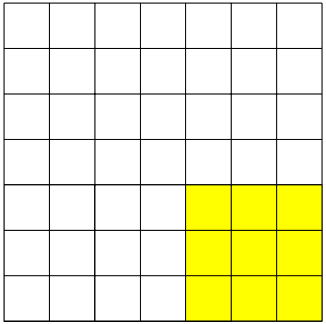


Grid size: $7 \times 7$

Filter size: $3 \times 3$

Stride: $2$

# 2D convolution with stride



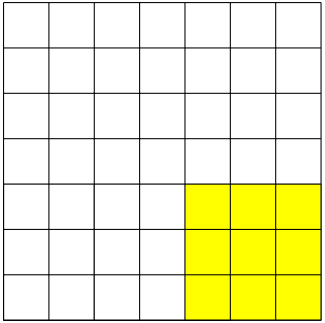
Grid size: $7 \times 7$

Filter size: $3 \times 3$

Stride: $2$

Output size: $3 \times 3$

# 2D convolution with stride



Grid size: $7 \times 7$

Filter size: $3 \times 3$

Stride: 2

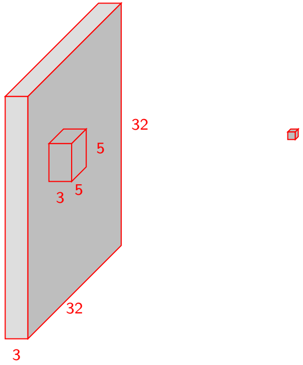
Output size: $3 \times 3$

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

N - input size, F - Filter size,

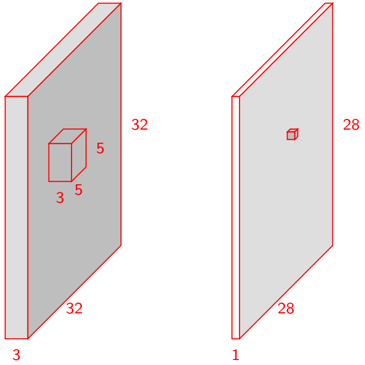
S - Stride

# Convolution operation



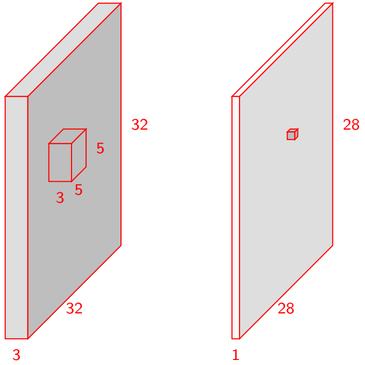
Filters are specified as $5 \times 5$. Channel depth is implicit.

# Convolution operation



Filters are specified as $5 \times 5$. Channel depth is implicit.

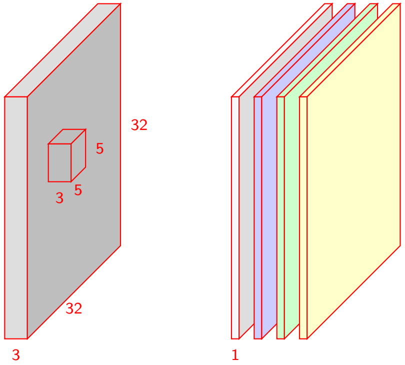
## Convolution operation



Filters are specified as $5 \times 5$. Channel depth is implicit.

No. of paramters 75 excluding bias. Computation (multiplication) - $28 \times 28 \times 5 \times 5 \times 3$

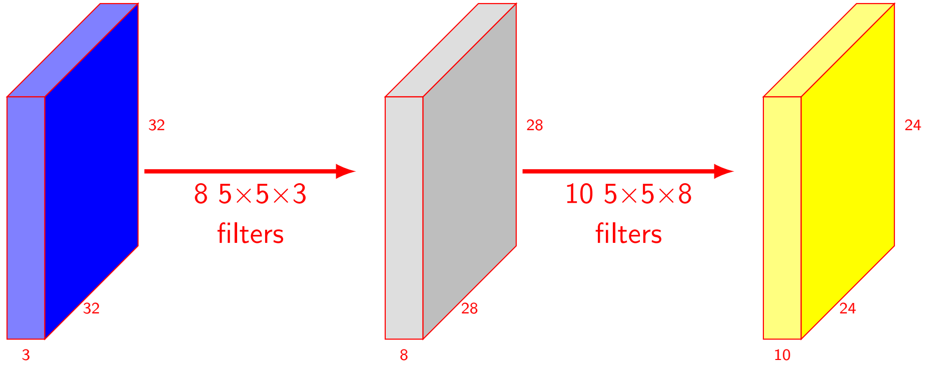
## Convolution operation



Filters are specified as $5 \times 5$. Channel depth is implicit.

No. of paramters 75 excluding bias. Computation (multiplication) - $28 \times 28 \times 5 \times 5 \times 3$

# Convolution example



# Edge detection

- Applied filter: $[1 \quad -1]$, Original image is on the left

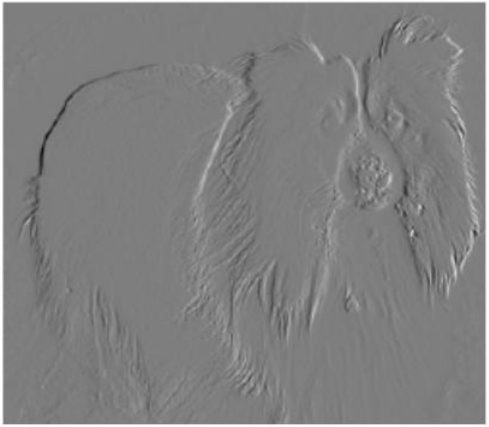


Image source: Deep Learning Book

# Convolution Filter-1

| | | |
|---|---|---|
| $1$ | $0$ | $-1$ |
| $1$ | $0$ | $-1$ |
| $1$ | $0$ | $-1$ |

# Convolution Filter-1

$$\begin{array}{|c|c|c|c|c|c|}\hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline \end{array} \otimes \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} =$$

# Convolution Filter-1

$$\begin{array}{|c|c|c|c|c|c|}\hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline \end{array} \otimes \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & 3 & 3 & 0 \\ \hline 0 & 3 & 3 & 0 \\ \hline 0 & 3 & 3 & 0 \\ \hline 0 & 3 & 3 & 0 \\ \hline \end{array}$$

# Convolution Filter-2

|  |  |  |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| $-1$ | $-1$ | $-1$ |

## Convolution Filter-2

$$\begin{array}{|c|c|c|c|c|c|}\hline 1&1&1&1&1&1\\\hline 1&1&1&1&1&1\\\hline 1&1&1&1&1&1\\\hline 0&0&0&0&0&0\\\hline 0&0&0&0&0&0\\\hline 0&0&0&0&0&0\\\hline\end{array} \otimes \begin{array}{|c|c|c|}\hline 1&1&1\\\hline 0&0&0\\\hline -1&-1&-1\\\hline\end{array} =$$

# Convolution Filter-2

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

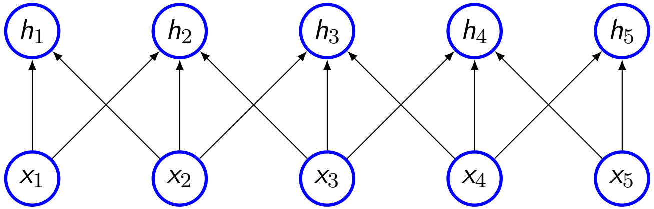
## Advantages

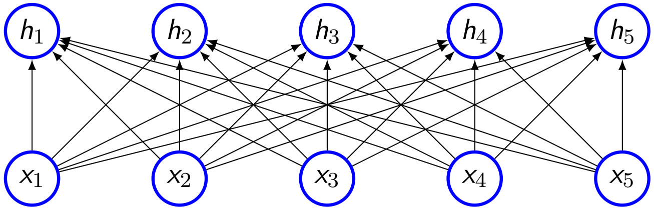
- Convolution can exploit the following properties
  - Sparse interaction (Also known as sparse connectivity or sparse weights)
  - Parameter sharing
  - Equivariant representation

## Sparse interaction

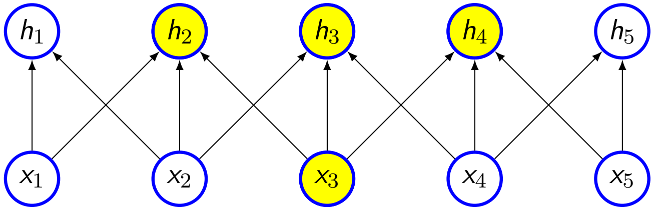
- Traditional neural network layers use matrix multiplication to describe how outputs and inputs are related
- Convolution uses a smaller kernel
  - Significant reduction in number of parameters
  - Computing output require few comparison
- For example, if there is $m$ inputs and $n$ outputs, traditional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

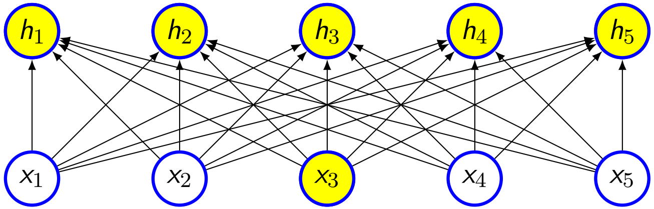
# Sparse connectivity



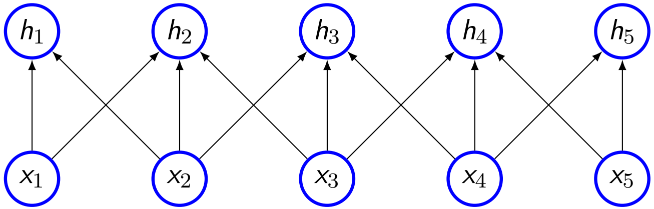


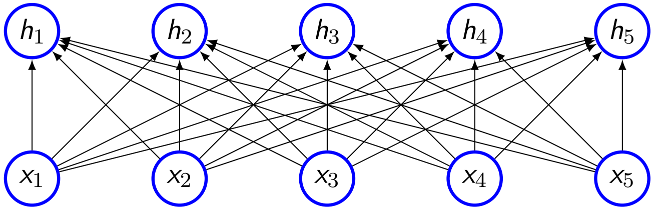
# Sparse connectivity



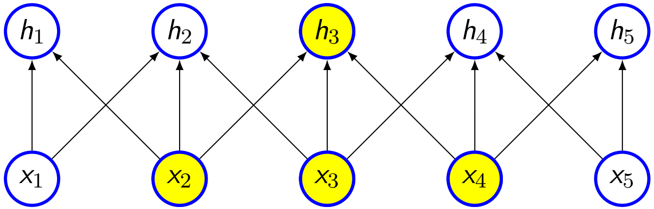


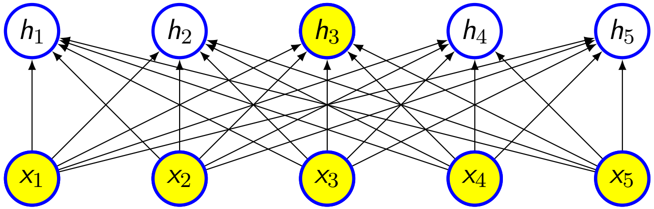
# Sparse connectivity



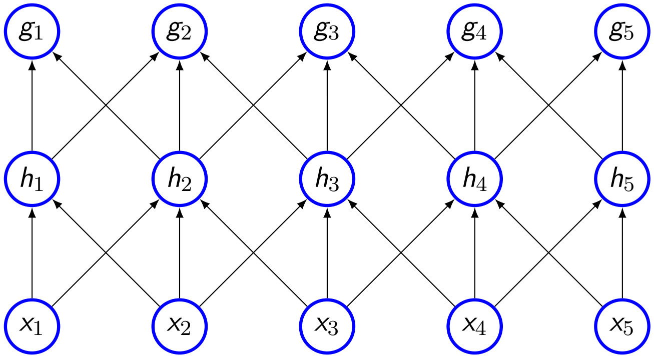


## Sparse connectivity

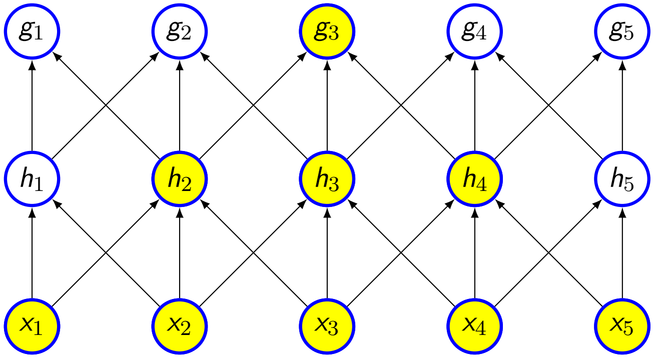




## Receptive field



# 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

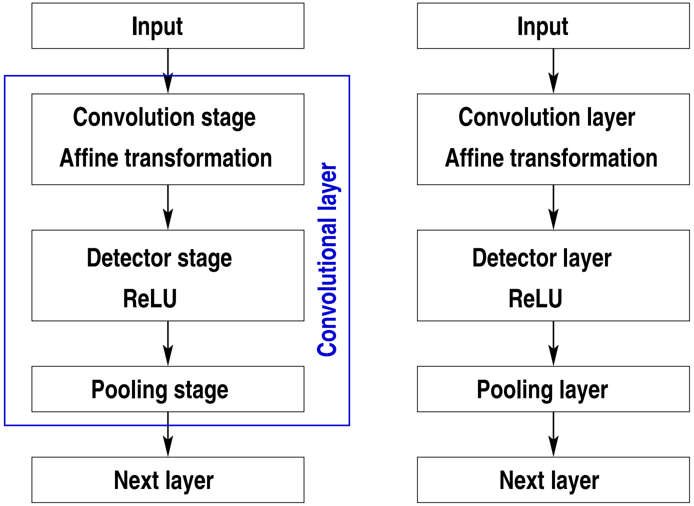
## Equivariance

- 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, combination of a linear translation and brightness at image coordinates

## 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

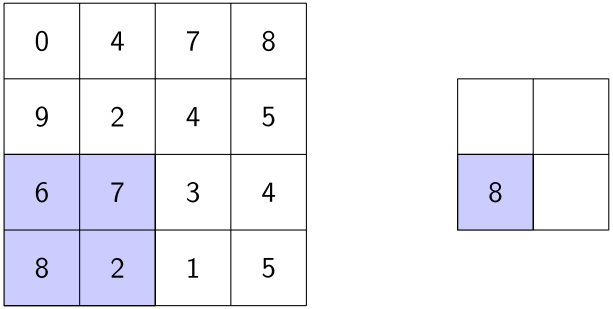

Input
Convolution stage
Affine transformation
Detector stage
ReLU
Pooling stage
Convolutional layer
Next layer
Input
Convolution layer
Affine transformation
Detector layer
ReLU
Pooling layer
Next layer

# Max Pool

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

| | |
|---|---|
| | |
| | |

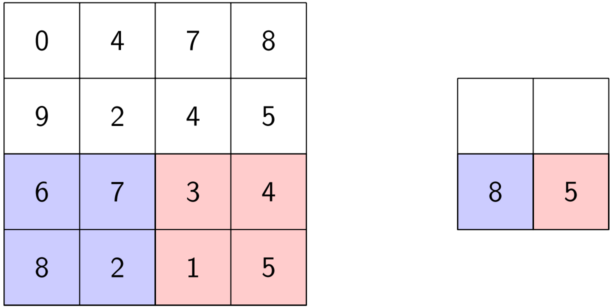
# Max Pool



| | | | |
|---|---|---|---|
| 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 |

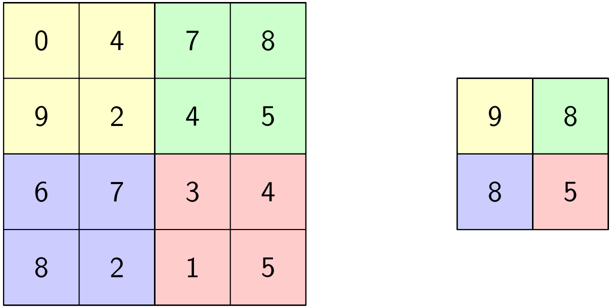
| | |
|---|---|
| 9 | |
| 8 | 5 |

# Max Pool

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

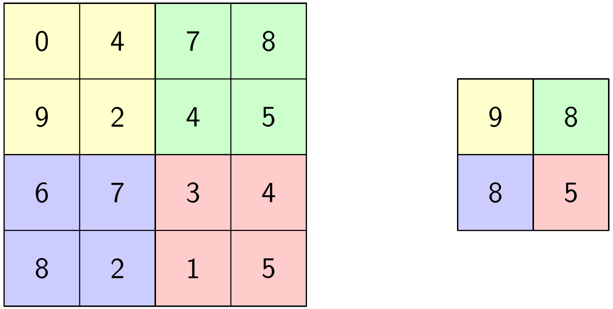
| | |
|---|---|
| 9 | 8 |
| 8 | 5 |

# Max Pool



How many parameters are there?

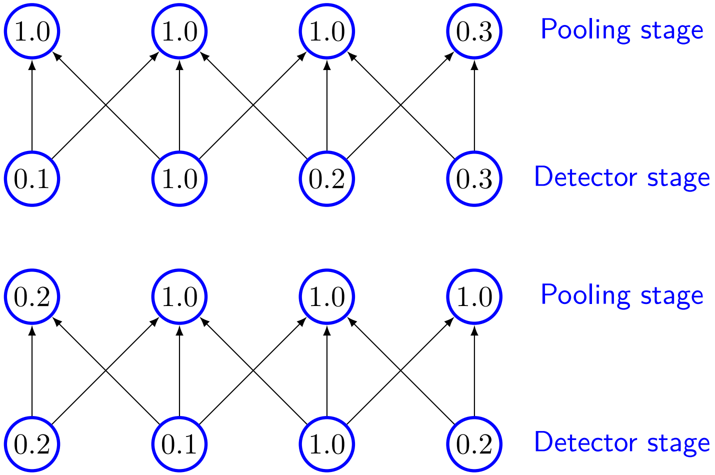
## Max Pool



How many parameters are there?

How to compute gradients?

# Invariance of maxpooling



# Learned invariances

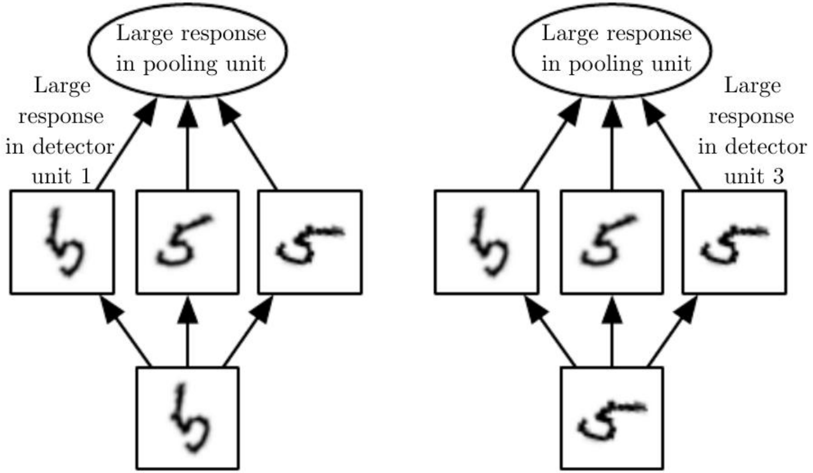
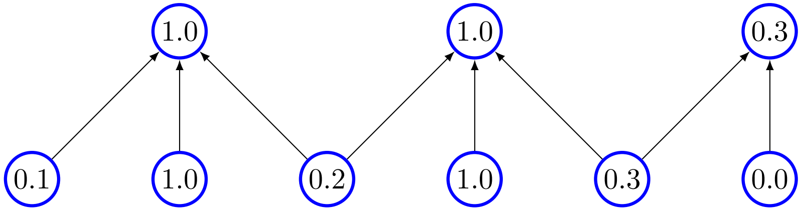
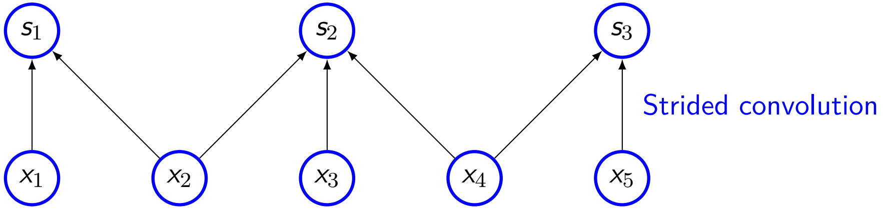

Large response in pooling unit
Large response in detector unit 1
Large response in pooling unit
Large response in detector unit 3

Image source: Deep Learning Book

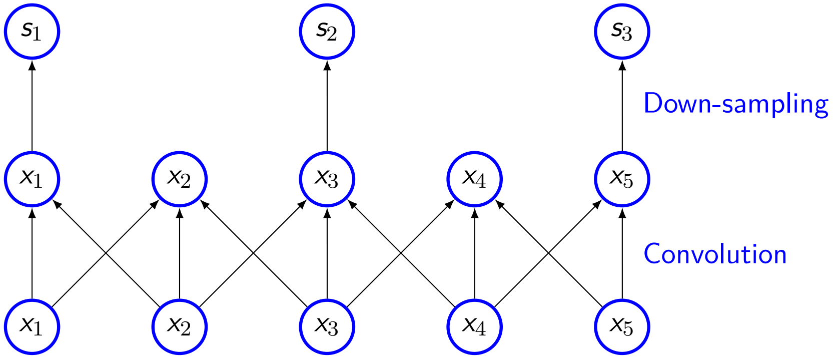
# Pooling with downsampling



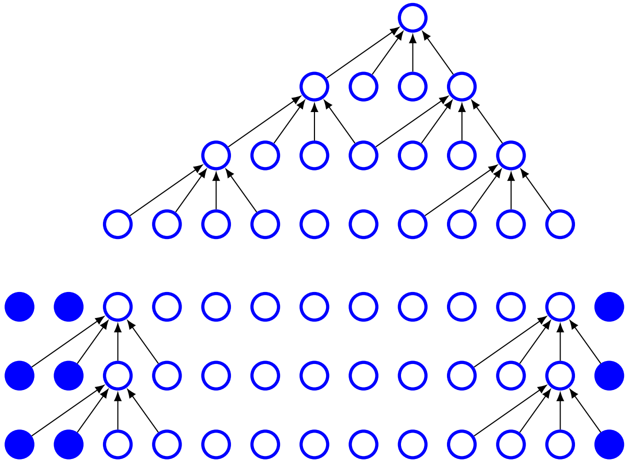
# Strided convolution



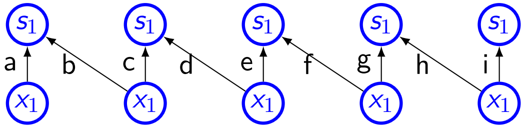
# Strided convolution (contd)

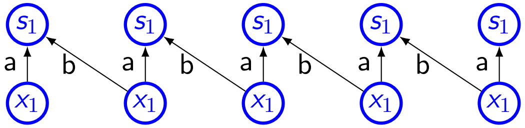


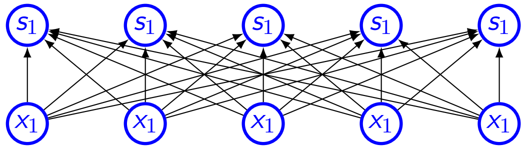
# Zero padding



# Connections





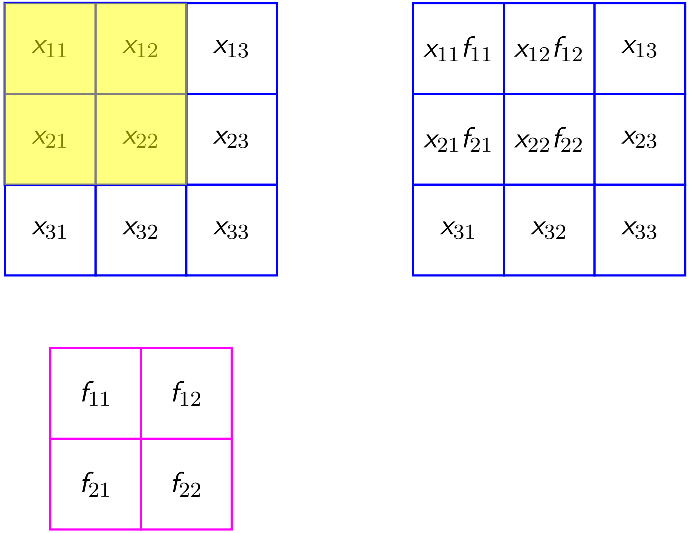


# CNN & Backpropagation-1

|  |  |  |
|---|---|---|
| $x_{11}$ | $x_{12}$ | $x_{13}$ |
| $x_{21}$ | $x_{22}$ | $x_{23}$ |
| $x_{31}$ | $x_{32}$ | $x_{33}$ |

|  |  |
|---|---|
| $f_{11}$ | $f_{12}$ |
| $f_{21}$ | $f_{22}$ |

## CNN & Backpropagation-1


$x_{11}$ $x_{12}$ $x_{13}$
$x_{21}$ $x_{22}$ $x_{23}$
$x_{31}$ $x_{32}$ $x_{33}$
$x_{11} f_{11}$ $x_{12} f_{12}$ $x_{13}$
$x_{21} f_{21}$ $x_{22} f_{22}$ $x_{23}$
$x_{31}$ $x_{32}$ $x_{33}$
$f_{11}$ $f_{12}$
$f_{21}$ $f_{22}$

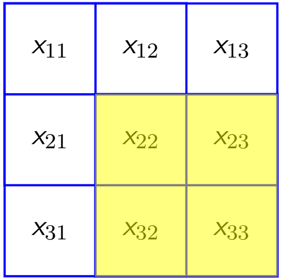
# CNN & Backpropagation-1

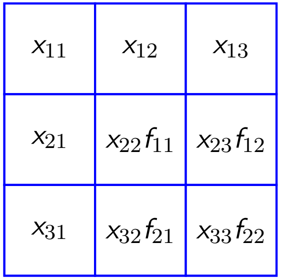
| | | |
|---|---|---|
| $x_{11}$ | $x_{12}$ | $x_{13}$ |
| $x_{21}$ | $x_{22}$ | $x_{23}$ |
| $x_{31}$ | $x_{32}$ | $x_{33}$ |

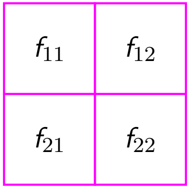
| | | |
|---|---|---|
| $x_{11}$ | $x_{12}$ | $x_{13}$ |
| $x_{21}$ | $x_{22}f_{11}$ | $x_{23}f_{12}$ |
| $x_{31}$ | $x_{32}f_{21}$ | $x_{33}f_{22}$ |

| | |
|---|---|
| $f_{11}$ | $f_{12}$ |
| $f_{21}$ | $f_{22}$ |

# CNN & Backpropagation-1


$x_{11}$ $x_{12}$ $x_{13}$
$x_{21}$ $x_{22}$ $x_{23}$
$x_{31}$ $x_{32}$ $x_{33}$


$x_{11}$ $x_{12}$ $x_{13}$
$x_{21}$ $x_{22}f_{11}$ $x_{23}f_{12}$
$x_{31}$ $x_{32}f_{21}$ $x_{33}f_{22}$



$$\begin{aligned}
o_{11} &= x_{11}f_{11} + x_{12}f_{12} + x_{21}f_{21} + x_{22}f_{22} \\
o_{12} &= x_{12}f_{11} + x_{13}f_{12} + x_{22}f_{21} + x_{23}f_{22} \\
o_{21} &= x_{21}f_{11} + x_{22}f_{12} + x_{31}f_{21} + x_{32}f_{22} \\
o_{22} &= x_{22}f_{11} + x_{23}f_{12} + x_{32}f_{21} + x_{33}f_{22}
\end{aligned}$$

## CNN & Backpropagation-2

- Gradient with respect to filter: $\dfrac{\partial L}{\partial F} = \dfrac{\partial L}{\partial o} \times \dfrac{\partial o}{\partial F}$
- Gradient with respect to every element: $\dfrac{\partial L}{\partial F_i} = \sum_{k=1}^{m} \dfrac{\partial L}{\partial o_k} \times \dfrac{\partial o_k}{\partial F_i}$

# CNN & Backpropagation-2

- Gradient with respect to filter: $\dfrac{\partial L}{\partial F} = \dfrac{\partial L}{\partial o} \times \dfrac{\partial o}{\partial F}$
- Gradient with respect to every element: $\dfrac{\partial L}{\partial F_i} = \sum_{k=1}^{m} \dfrac{\partial L}{\partial o_k} \times \dfrac{\partial o_k}{\partial F_i}$
- Hence,

$$\frac{\partial L}{\partial f_{11}} =$$

# CNN & Backpropagation-2

- Gradient with respect to filter: $\frac{\partial L}{\partial F} = \frac{\partial L}{\partial o} \times \frac{\partial o}{\partial F}$
- Gradient with respect to every element: $\frac{\partial L}{\partial F_i} = \sum_{k=1}^{m} \frac{\partial L}{\partial o_k} \times \frac{\partial o_k}{\partial F_i}$
- Hence,

$$\frac{\partial L}{\partial f_{11}} = \frac{\partial L}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial f_{11}} + \frac{\partial L}{\partial o_{12}} \times \frac{\partial o_{12}}{\partial f_{11}} + \frac{\partial L}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial f_{11}} + \frac{\partial L}{\partial o_{22}} \times \frac{\partial o_{22}}{\partial f_{11}}$$

## CNN & Backpropagation-2

- Gradient with respect to filter: $\dfrac{\partial L}{\partial F} = \dfrac{\partial L}{\partial o} \times \dfrac{\partial o}{\partial F}$
- Gradient with respect to every element: $\dfrac{\partial L}{\partial F_i} = \sum_{k=1}^{m} \dfrac{\partial L}{\partial o_k} \times \dfrac{\partial o_k}{\partial F_i}$
- Hence,

$$\begin{aligned}
\frac{\partial L}{\partial f_{11}} &= \frac{\partial L}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial f_{11}} + \frac{\partial L}{\partial o_{12}} \times \frac{\partial o_{12}}{\partial f_{11}} + \frac{\partial L}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial f_{11}} + \frac{\partial L}{\partial o_{22}} \times \frac{\partial o_{22}}{\partial f_{11}} \\
\frac{\partial L}{\partial f_{12}} &= \frac{\partial L}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial f_{12}} + \frac{\partial L}{\partial o_{12}} \times \frac{\partial o_{12}}{\partial f_{12}} + \frac{\partial L}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial f_{12}} + \frac{\partial L}{\partial o_{22}} \times \frac{\partial o_{22}}{\partial f_{12}} \\
\frac{\partial L}{\partial f_{21}} &= \frac{\partial L}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial f_{21}} + \frac{\partial L}{\partial o_{12}} \times \frac{\partial o_{12}}{\partial f_{21}} + \frac{\partial L}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial f_{21}} + \frac{\partial L}{\partial o_{22}} \times \frac{\partial o_{22}}{\partial f_{21}} \\
\frac{\partial L}{\partial f_{22}} &= \frac{\partial L}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial f_{22}} + \frac{\partial L}{\partial o_{12}} \times \frac{\partial o_{12}}{\partial f_{22}} + \frac{\partial L}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial f_{22}} + \frac{\partial L}{\partial o_{22}} \times \frac{\partial o_{22}}{\partial f_{22}}
\end{aligned}$$

## CNN & Backpropagation-3

- After simplification, we get

$$
\begin{aligned}
\frac{\partial L}{\partial f_{11}} &= \frac{\partial L}{\partial o_{11}} \times x_{11} + \frac{\partial L}{\partial o_{12}} \times x_{12} + \frac{\partial L}{\partial o_{21}} \times x_{21} + \frac{\partial L}{\partial o_{22}} \times x_{22} \\
\frac{\partial L}{\partial f_{12}} &= \frac{\partial L}{\partial o_{11}} \times x_{12} + \frac{\partial L}{\partial o_{12}} \times x_{13} + \frac{\partial L}{\partial o_{21}} \times x_{22} + \frac{\partial L}{\partial o_{22}} \times x_{23} \\
\frac{\partial L}{\partial f_{21}} &= \frac{\partial L}{\partial o_{11}} \times x_{21} + \frac{\partial L}{\partial o_{12}} \times x_{22} + \frac{\partial L}{\partial o_{21}} \times x_{31} + \frac{\partial L}{\partial o_{22}} \times x_{32} \\
\frac{\partial L}{\partial f_{22}} &= \frac{\partial L}{\partial o_{11}} \times x_{22} + \frac{\partial L}{\partial o_{12}} \times x_{23} + \frac{\partial L}{\partial o_{21}} \times x_{32} + \frac{\partial L}{\partial o_{22}} \times x_{33}
\end{aligned}
$$

# CNN & Backpropagation-3

- After simplification, we get

$$\frac{\partial L}{\partial f_{11}} = \frac{\partial L}{\partial o_{11}} \times x_{11} + \frac{\partial L}{\partial o_{12}} \times x_{12} + \frac{\partial L}{\partial o_{21}} \times x_{21} + \frac{\partial L}{\partial o_{22}} \times x_{22}$$
$$\frac{\partial L}{\partial f_{12}} = \frac{\partial L}{\partial o_{11}} \times x_{12} + \frac{\partial L}{\partial o_{12}} \times x_{13} + \frac{\partial L}{\partial o_{21}} \times x_{22} + \frac{\partial L}{\partial o_{22}} \times x_{23}$$
$$\frac{\partial L}{\partial f_{21}} = \frac{\partial L}{\partial o_{11}} \times x_{21} + \frac{\partial L}{\partial o_{12}} \times x_{22} + \frac{\partial L}{\partial o_{21}} \times x_{31} + \frac{\partial L}{\partial o_{22}} \times x_{32}$$
$$\frac{\partial L}{\partial f_{22}} = \frac{\partial L}{\partial o_{11}} \times x_{22} + \frac{\partial L}{\partial o_{12}} \times x_{23} + \frac{\partial L}{\partial o_{21}} \times x_{32} + \frac{\partial L}{\partial o_{22}} \times x_{33}$$

$$\begin{array}{|c|c|c|} \hline x_{11} & x_{12} & x_{13} \\ \hline x_{21} & x_{22} & x_{23} \\ \hline x_{31} & x_{32} & x_{33} \\ \hline \end{array} \otimes \begin{array}{|c|c|} \hline \frac{\partial L}{\partial o_{11}} & \frac{\partial L}{\partial o_{12}} \\ \hline \frac{\partial L}{\partial o_{21}} & \frac{\partial L}{\partial o_{22}} \\ \hline \end{array} = \begin{array}{|c|c|} \hline \frac{\partial L}{\partial f_{11}} & \frac{\partial L}{\partial f_{12}} \\ \hline \frac{\partial L}{\partial f_{21}} & \frac{\partial L}{\partial f_{22}} \\ \hline \end{array}$$

## CNN & Backpropagation-4

- Gradient with respect to filter: $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial o} \times \frac{\partial o}{\partial x}$
- Gradient with respect to every element: $\frac{\partial L}{\partial x_i} = \sum_{k=1}^{m} \frac{\partial L}{\partial o_k} \times \frac{\partial o_k}{\partial x_i}$

## CNN & Backpropagation-4

- Gradient with respect to filter: $\dfrac{\partial L}{\partial x} = \dfrac{\partial L}{\partial o} \times \dfrac{\partial o}{\partial x}$
- Gradient with respect to every element: $\dfrac{\partial L}{\partial x_i} = \displaystyle\sum_{k=1}^{m} \dfrac{\partial L}{\partial o_k} \times \dfrac{\partial o_k}{\partial x_i}$
- Hence,

  $\frac{\partial L}{\partial x_{11}} =$

## CNN & Backpropagation-4

- Gradient with respect to filter: $\dfrac{\partial L}{\partial x} = \dfrac{\partial L}{\partial o} \times \dfrac{\partial o}{\partial x}$
- Gradient with respect to every element: $\dfrac{\partial L}{\partial x_i} = \displaystyle\sum_{k=1}^{m} \dfrac{\partial L}{\partial o_k} \times \dfrac{\partial o_k}{\partial x_i}$
- Hence,

  $\frac{\partial L}{\partial x_{11}} = \frac{\partial L}{\partial o_{11}} \times f_{11}$ $\qquad$ $\frac{\partial L}{\partial x_{12}} =$

## CNN & Backpropagation-4

- Gradient with respect to filter: $\dfrac{\partial L}{\partial x} = \dfrac{\partial L}{\partial o} \times \dfrac{\partial o}{\partial x}$
- Gradient with respect to every element: $\dfrac{\partial L}{\partial x_i} = \sum_{k=1}^{m} \dfrac{\partial L}{\partial o_k} \times \dfrac{\partial o_k}{\partial x_i}$
- Hence,

$\frac{\partial L}{\partial x_{11}} = \frac{\partial L}{\partial o_{11}} \times f_{11}$ $\qquad$ $\frac{\partial L}{\partial x_{12}} = \frac{\partial L}{\partial o_{11}} \times f_{12} + \frac{\partial L}{\partial o_{12}} \times f_{11}$

# CNN & Backpropagation-4

- Gradient with respect to filter: $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial o} \times \frac{\partial o}{\partial x}$
- Gradient with respect to every element: $\frac{\partial L}{\partial x_i} = \sum_{k=1}^{m} \frac{\partial L}{\partial o_k} \times \frac{\partial o_k}{\partial x_i}$
- Hence,

$$\frac{\partial L}{\partial x_{11}} = \frac{\partial L}{\partial o_{11}} \times f_{11} \qquad \frac{\partial L}{\partial x_{12}} = \frac{\partial L}{\partial o_{11}} \times f_{12} + \frac{\partial L}{\partial o_{12}} \times f_{11}$$

$$\frac{\partial L}{\partial x_{13}} = \frac{\partial L}{\partial o_{12}} \times f_{12} \qquad \frac{\partial L}{\partial x_{21}} = \frac{\partial L}{\partial o_{11}} \times f_{21} + \frac{\partial L}{\partial o_{21}} \times f_{11}$$

$$\frac{\partial L}{\partial x_{22}} =$$

## CNN & Backpropagation-4

- Gradient with respect to filter: $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial o} \times \frac{\partial o}{\partial x}$
- Gradient with respect to every element: $\frac{\partial L}{\partial x_i} = \sum_{k=1}^{m} \frac{\partial L}{\partial o_k} \times \frac{\partial o_k}{\partial x_i}$
- Hence,

$$\frac{\partial L}{\partial x_{11}} = \frac{\partial L}{\partial o_{11}} \times f_{11} \qquad \frac{\partial L}{\partial x_{12}} = \frac{\partial L}{\partial o_{11}} \times f_{12} + \frac{\partial L}{\partial o_{12}} \times f_{11}$$

$$\frac{\partial L}{\partial x_{13}} = \frac{\partial L}{\partial o_{12}} \times f_{12} \qquad \frac{\partial L}{\partial x_{21}} = \frac{\partial L}{\partial o_{11}} \times f_{21} + \frac{\partial L}{\partial o_{21}} \times f_{11}$$

$$\frac{\partial L}{\partial x_{22}} = \frac{\partial L}{\partial o_{12}} \times f_{22} + \frac{\partial L}{\partial x_{12}} \times f_{21} + \frac{\partial L}{\partial o_{21}} \times f_{12} + \frac{\partial L}{\partial o_{22}} \times f_{11}$$

# CNN & Backpropagation-4

- Gradient with respect to filter: $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial o} \times \frac{\partial o}{\partial x}$
- Gradient with respect to every element: $\frac{\partial L}{\partial x_i} = \sum_{k=1}^{m} \frac{\partial L}{\partial o_k} \times \frac{\partial o_k}{\partial x_i}$
- Hence,

$$\frac{\partial L}{\partial x_{11}} = \frac{\partial L}{\partial o_{11}} \times f_{11} \qquad \frac{\partial L}{\partial x_{12}} = \frac{\partial L}{\partial o_{11}} \times f_{12} + \frac{\partial L}{\partial o_{12}} \times f_{11}$$

$$\frac{\partial L}{\partial x_{13}} = \frac{\partial L}{\partial o_{12}} \times f_{12} \qquad \frac{\partial L}{\partial x_{21}} = \frac{\partial L}{\partial o_{11}} \times f_{21} + \frac{\partial L}{\partial o_{21}} \times f_{11}$$

$$\frac{\partial L}{\partial x_{22}} = \frac{\partial L}{\partial o_{12}} \times f_{22} + \frac{\partial L}{\partial x_{12}} \times f_{21} + \frac{\partial L}{\partial o_{21}} \times f_{12} + \frac{\partial L}{\partial o_{22}} \times f_{11}$$

$$\frac{\partial L}{\partial x_{23}} = \frac{\partial L}{\partial o_{12}} \times f_{22} + \frac{\partial L}{\partial o_{22}} \times f_{12} \qquad \frac{\partial L}{\partial x_{31}} = \frac{\partial L}{\partial o_{21}} \times f_{21}$$

$$\frac{\partial L}{\partial x_{32}} = \frac{\partial L}{\partial o_{21}} \times f_{22} + \frac{\partial L}{\partial o_{22}} \times f_{21} \qquad \frac{\partial L}{\partial x_{33}} = \frac{\partial L}{\partial o_{22}} \times f_{22}$$

# CNN & Backpropagation-5

$$\begin{bmatrix} \frac{\partial L}{\partial x_{11}} & \frac{\partial L}{\partial x_{12}} & \frac{\partial L}{\partial x_{13}} \\ \frac{\partial L}{\partial x_{21}} & \frac{\partial L}{\partial x_{22}} & \frac{\partial L}{\partial x_{23}} \\ \frac{\partial L}{\partial x_{31}} & \frac{\partial L}{\partial x_{32}} & \frac{\partial L}{\partial x_{33}} \end{bmatrix} = \begin{bmatrix} f_{22} & f_{21} \\ f_{12} & f_{11} \end{bmatrix} \boxed{\otimes} \begin{bmatrix} \frac{\partial L}{\partial o_{11}} & \frac{\partial L}{\partial o_{12}} \\ \frac{\partial L}{\partial o_{21}} & \frac{\partial L}{\partial o_{22}} \end{bmatrix}$$

## CNN Architectures

- AlexNet
- VggNet
- GoogleNet
- ResNet
- SENet
- Wide ResNet
- ResNeXT
- DenseNet
- MobileNets
- NASNet
- EfficientNet
- ... and many more

# ImageNet Challenge

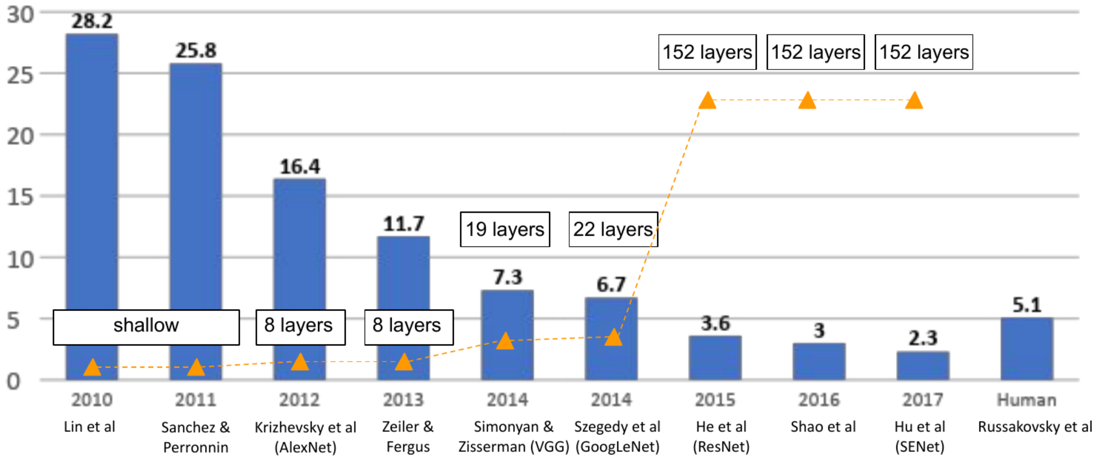


Image source: Jiajun et al. CS230 slides

# AlexNet

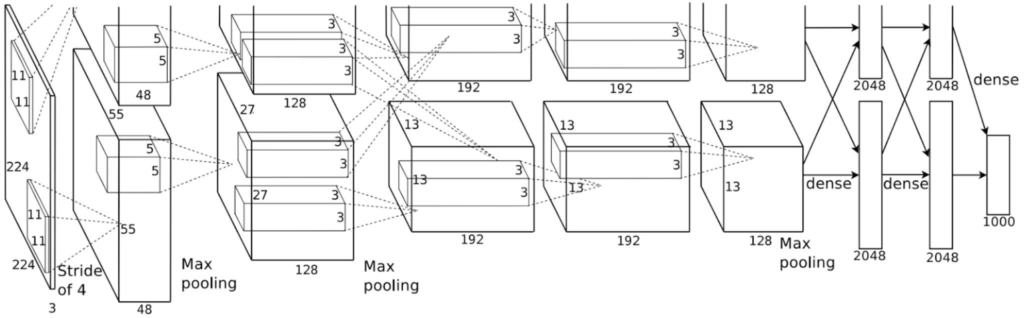
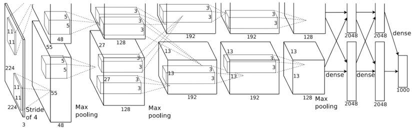


Image source: https://worksheets.codalab.org

## AlexNet



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

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

# ZFNet

- Almost similar to AlexNet
- CONV1: changes from (11x11 stride 4) to (7x7 stride 2)
- CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512
- Error reduces to 11.7% from 16.4% (top 5)

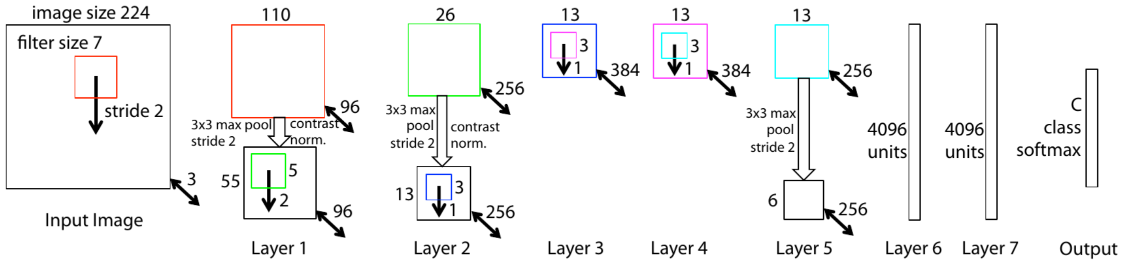


Image source: Visualizing and Understanding Convolutional Networks

# VggNet: VGG16

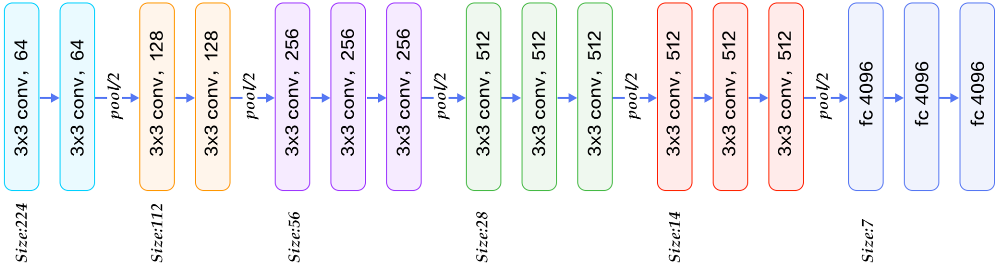


Image source: internet

# VggNet: Vgg16 vs Vgg19

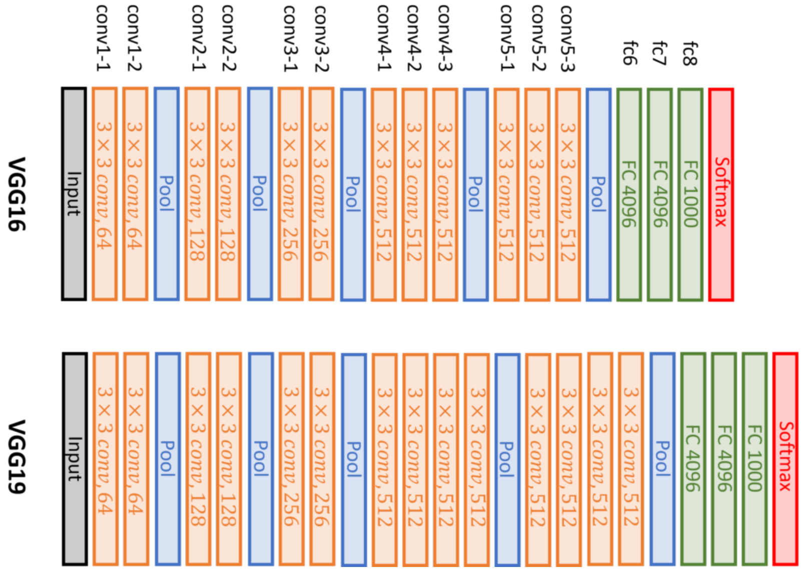


Image source: internet

# Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |

# Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | | | |

# Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | 224x224x64 | 3.2M | 1728 |
| C3-64 | | | |

## Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | 224x224x64 | 3.2M | 1728 |
| C3-64 | 224x224x64 | 3.2M | 36864 |
| Pool2 | | | |

## Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | 224x224x64 | 3.2M | 1728 |
| C3-64 | 224x224x64 | 3.2M | 36864 |
| Pool2 | 112x112x64 | 800k | 0 |
| C3-128 | | | |

# Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | 224x224x64 | 3.2M | 1728 |
| C3-64 | 224x224x64 | 3.2M | 36864 |
| Pool2 | 112x112x64 | 800k | 0 |
| C3-128 | 112x112x128 | 1.6M | 73728 |
| C3-128 | | | |

## Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | 224x224x64 | 3.2M | 1728 |
| C3-64 | 224x224x64 | 3.2M | 36864 |
| Pool2 | 112x112x64 | 800k | 0 |
| C3-128 | 112x112x128 | 1.6M | 73728 |
| C3-128 | 112x112x128 | 1.6M | 147456 |
| Pool2 | | | |

# Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | 224x224x64 | 3.2M | 1728 |
| C3-64 | 224x224x64 | 3.2M | 36864 |
| Pool2 | 112x112x64 | 800k | 0 |
| C3-128 | 112x112x128 | 1.6M | 73728 |
| C3-128 | 112x112x128 | 1.6M | 147456 |
| Pool2 | 56x56x128 | 400k | 0 |
| C3-256 | | | |

# Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | 224x224x64 | 3.2M | 1728 |
| C3-64 | 224x224x64 | 3.2M | 36864 |
| Pool2 | 112x112x64 | 800k | 0 |
| C3-128 | 112x112x128 | 1.6M | 73728 |
| C3-128 | 112x112x128 | 1.6M | 147456 |
| Pool2 | 56x56x128 | 400k | 0 |
| C3-256 | 56x56x256 | 800k | 294912 |
| C3-256 | | | |

## Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | 224x224x64 | 3.2M | 1728 |
| C3-64 | 224x224x64 | 3.2M | 36864 |
| Pool2 | 112x112x64 | 800k | 0 |
| C3-128 | 112x112x128 | 1.6M | 73728 |
| C3-128 | 112x112x128 | 1.6M | 147456 |
| Pool2 | 56x56x128 | 400k | 0 |
| C3-256 | 56x56x256 | 800k | 294912 |
| C3-256 | 56x56x256 | 800k | 589824 |
| C3-256 | | | |

# Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | 224x224x64 | 3.2M | 1728 |
| C3-64 | 224x224x64 | 3.2M | 36864 |
| Pool2 | 112x112x64 | 800k | 0 |
| C3-128 | 112x112x128 | 1.6M | 73728 |
| C3-128 | 112x112x128 | 1.6M | 147456 |
| Pool2 | 56x56x128 | 400k | 0 |
| C3-256 | 56x56x256 | 800k | 294912 |
| C3-256 | 56x56x256 | 800k | 589824 |
| C3-256 | 56x56x256 | 800k | 589824 |
| Pool2 | | | |

# Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | 224x224x64 | 3.2M | 1728 |
| C3-64 | 224x224x64 | 3.2M | 36864 |
| Pool2 | 112x112x64 | 800k | 0 |
| C3-128 | 112x112x128 | 1.6M | 73728 |
| C3-128 | 112x112x128 | 1.6M | 147456 |
| Pool2 | 56x56x128 | 400k | 0 |
| C3-256 | 56x56x256 | 800k | 294912 |
| C3-256 | 56x56x256 | 800k | 589824 |
| C3-256 | 56x56x256 | 800k | 589824 |
| Pool2 | 28x28x256 | 200k | 0 |

| Layer | Size | Memory | Params |
|---|---|---|---|
| C3-512 | 28x28x512 | 400k | 1179648 |
| C3-512 | 28x28x512 | 400k | 2359296 |
| C3-512 | 28x28x512 | 400k | 2359296 |
| Pool2 | 14x14x512 | 100k | 0 |
| C3-512 | 14x14x512 | 100k | 2359296 |
| C3-512 | 14x14x512 | 100k | 2359296 |
| C3-512 | 14x14x512 | 100k | 2359296 |
| Pool2 | 7x7x512 | 25k | 0 |
| FC | 1x1x4096 | | |

## Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | 224x224x64 | 3.2M | 1728 |
| C3-64 | 224x224x64 | 3.2M | 36864 |
| Pool2 | 112x112x64 | 800k | 0 |
| C3-128 | 112x112x128 | 1.6M | 73728 |
| C3-128 | 112x112x128 | 1.6M | 147456 |
| Pool2 | 56x56x128 | 400k | 0 |
| C3-256 | 56x56x256 | 800k | 294912 |
| C3-256 | 56x56x256 | 800k | 589824 |
| C3-256 | 56x56x256 | 800k | 589824 |
| Pool2 | 28x28x256 | 200k | 0 |

| Layer | Size | Memory | Params |
|---|---|---|---|
| C3-512 | 28x28x512 | 400k | 1179648 |
| C3-512 | 28x28x512 | 400k | 2359296 |
| C3-512 | 28x28x512 | 400k | 2359296 |
| Pool2 | 14x14x512 | 100k | 0 |
| C3-512 | 14x14x512 | 100k | 2359296 |
| C3-512 | 14x14x512 | 100k | 2359296 |
| C3-512 | 14x14x512 | 100k | 2359296 |
| Pool2 | 7x7x512 | 25k | 0 |
| FC | 1x1x4096 | 4096 | 102760448 |
| FC | 1x1x4096 | 4096 | 16777216 |
| FC | 1x1x1000 | 1000 | 4096000 |

# Convolution Filter-2

| Layer | Size | Memory | Params |
|---|---|---|---|
| Input | 224x224x3 | 150k | 0 |
| C3-64 | 224x224x64 | 3.2M | 1728 |
| C3-64 | 224x224x64 | 3.2M | 36864 |
| Pool2 | 112x112x64 | 800k | 0 |
| C3-128 | 112x112x128 | 1.6M | 73728 |
| C3-128 | 112x112x128 | 1.6M | 147456 |
| Pool2 | 56x56x128 | 400k | 0 |
| C3-256 | 56x56x256 | 800k | 294912 |
| C3-256 | 56x56x256 | 800k | 589824 |
| C3-256 | 56x56x256 | 800k | 589824 |
| Pool2 | 28x28x256 | 200k | 0 |

| Layer | Size | Memory | Params |
|---|---|---|---|
| C3-512 | 28x28x512 | 400k | 1179648 |
| C3-512 | 28x28x512 | 400k | 2359296 |
| C3-512 | 28x28x512 | 400k | 2359296 |
| Pool2 | 14x14x512 | 100k | 0 |
| C3-512 | 14x14x512 | 100k | 2359296 |
| C3-512 | 14x14x512 | 100k | 2359296 |
| C3-512 | 14x14x512 | 100k | 2359296 |
| Pool2 | 7x7x512 | 25k | 0 |
| FC | 1x1x4096 | 4096 | 102760448 |
| FC | 1x1x4096 | 4096 | 16777216 |
| FC | 1x1x1000 | 1000 | 4096000 |

- Total memory: 24M * 4 bytes $\sim=$ 96MB
- Total params: 138M

# GoogleNet

- Winner for 2014, 6.7% error for top-5
- 22 Layers
- Only 5 million parameters
- 12X less than AlexNet, 27X less than VGG16
- Efficient inception module
- No FC layers

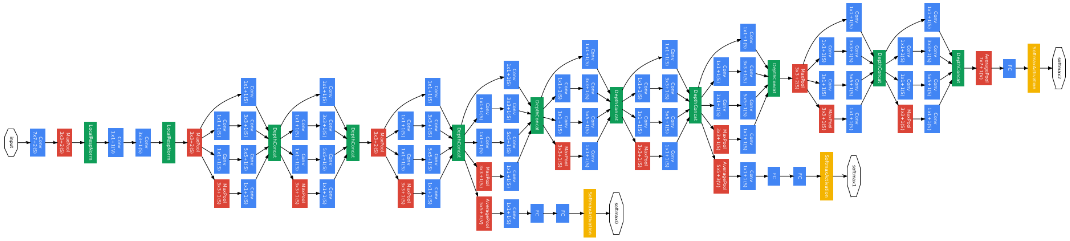
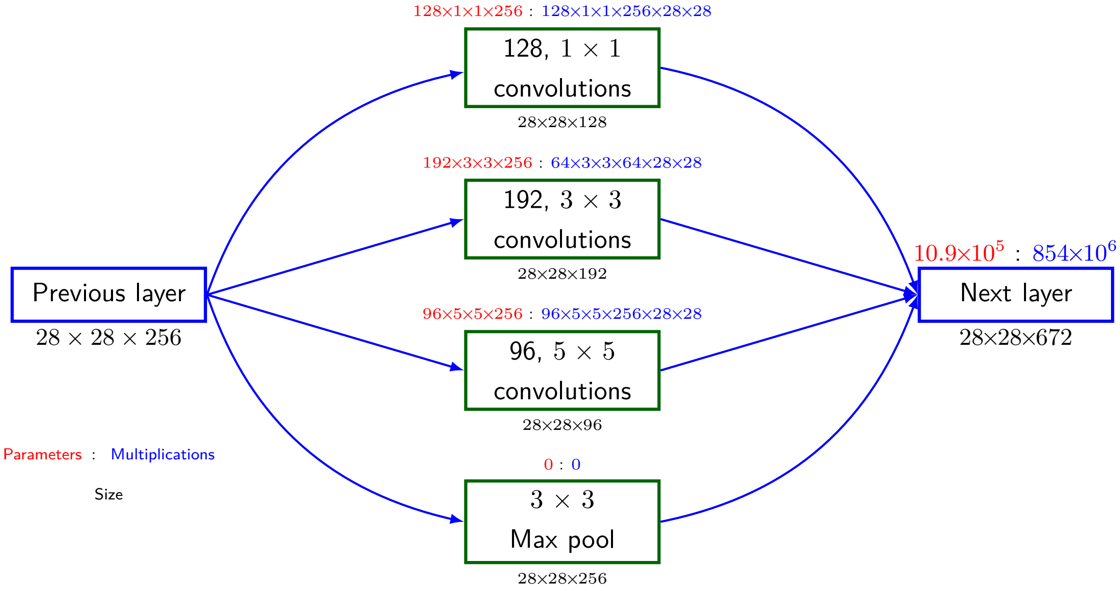
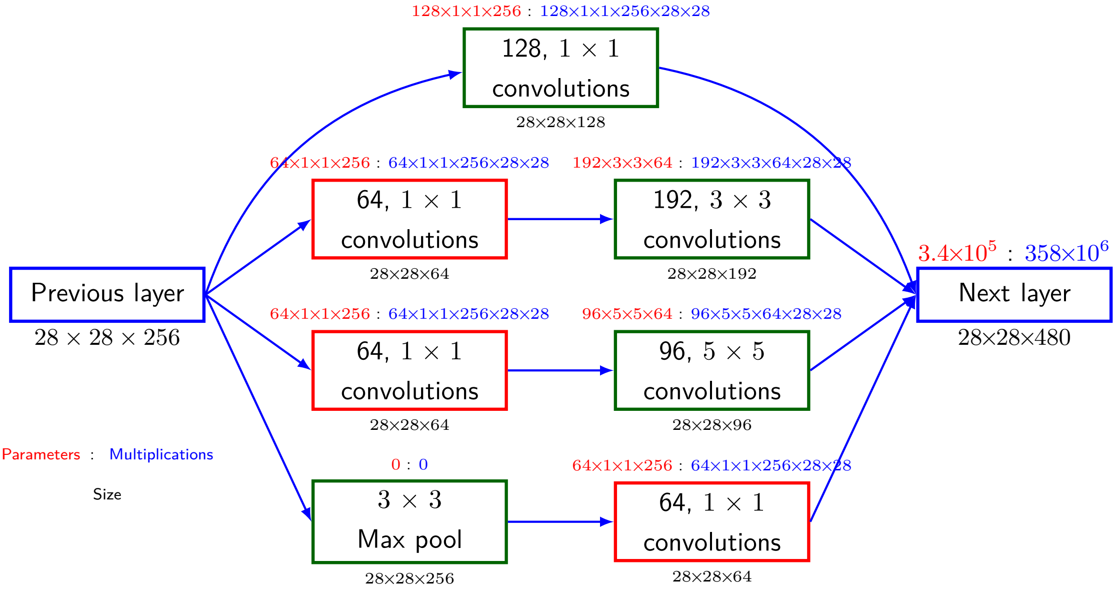


Image source: http://joelouismarino.github.io

# Naive inception



# Inception

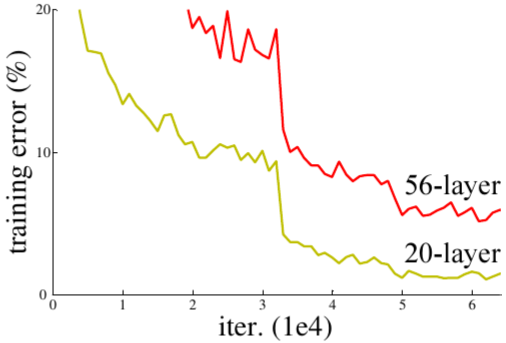


Parameters : Multiplications

Size

## ResNet: Observation

- Winner for for 2015, 3.57% error for top-5
- 152 Layers



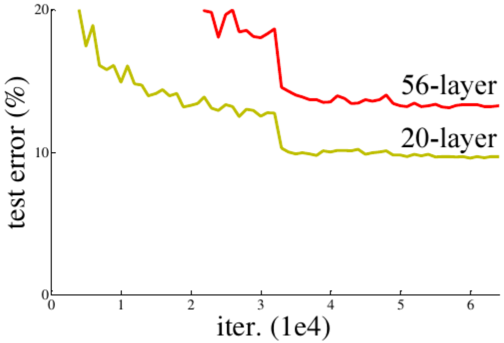


Image source: internet

# ResNet

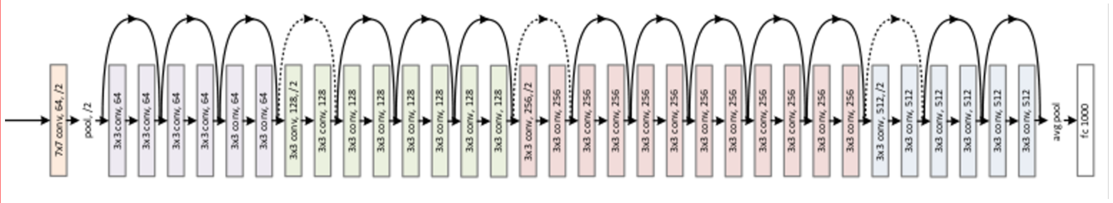
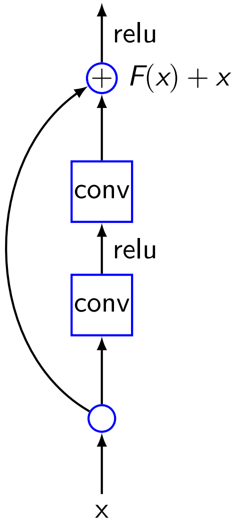
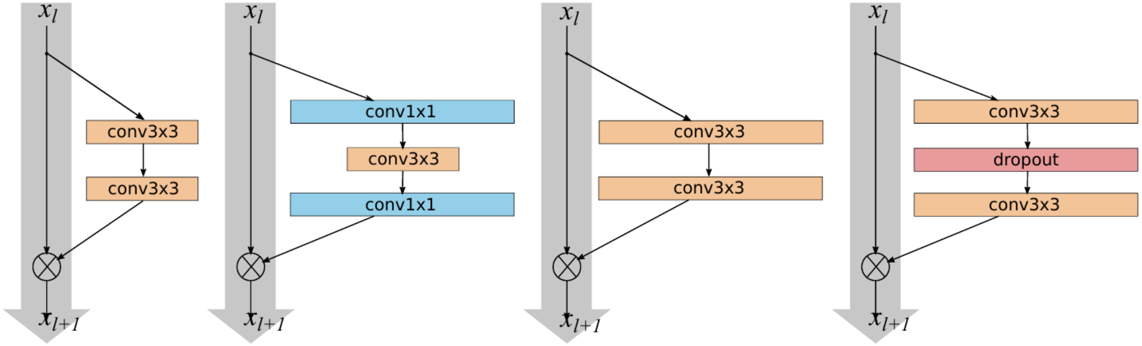


Image source: internet

# Resnet: Skip Connection



## Wide Resnet



(a) basic (b) bottleneck (c) basic-wide (d) wide-dropout

Image source: Wide Residual Networks

# DenseNet

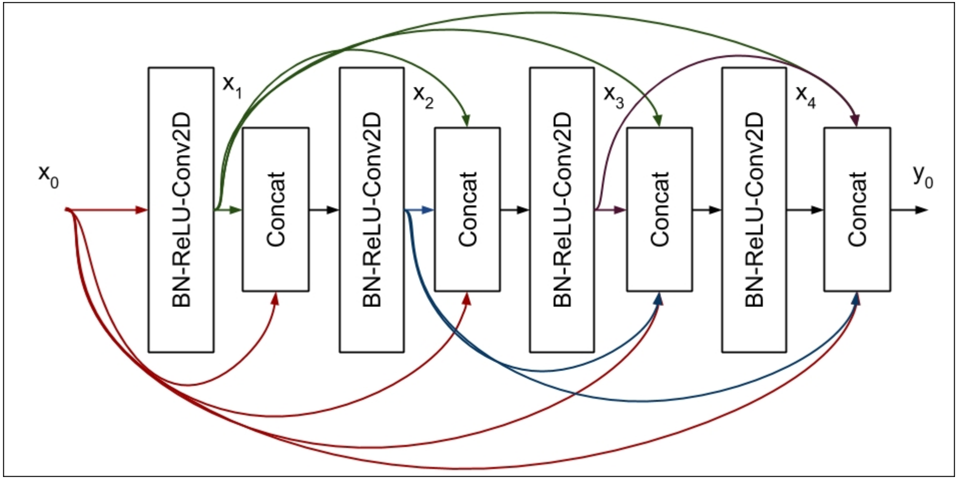
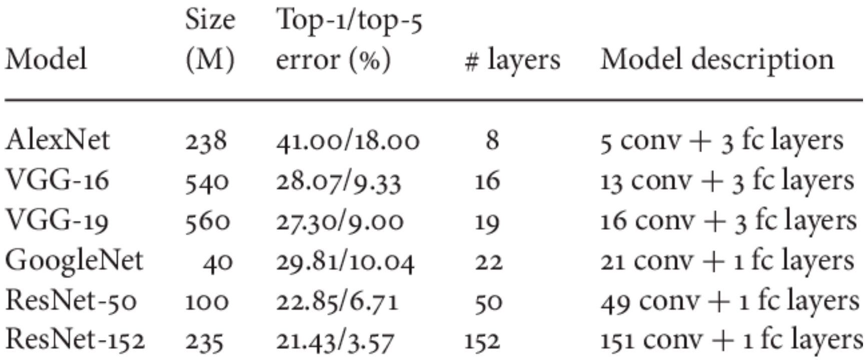


Image source: internet

# 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

## Computer Vision Tasks

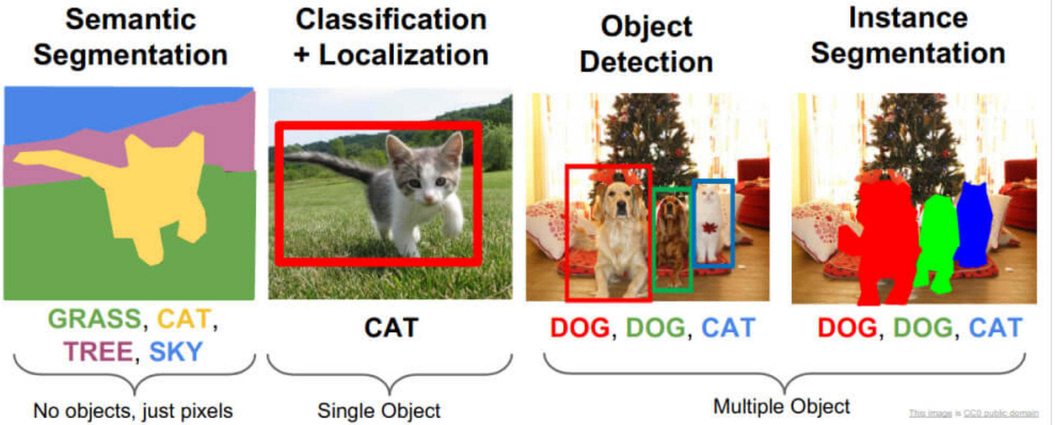
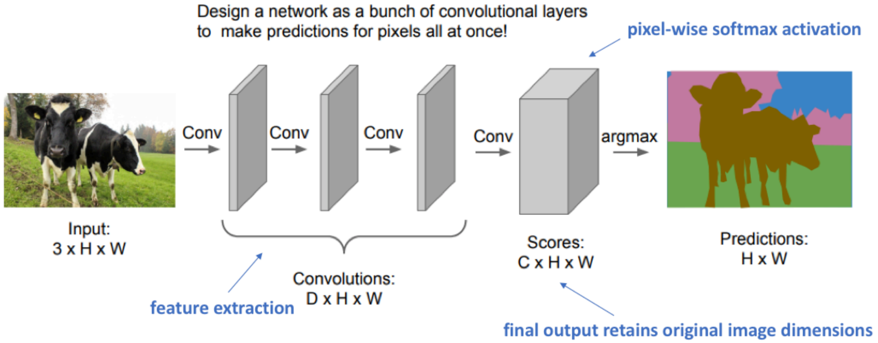


Image source: internet

# Semantic Segmentation-1

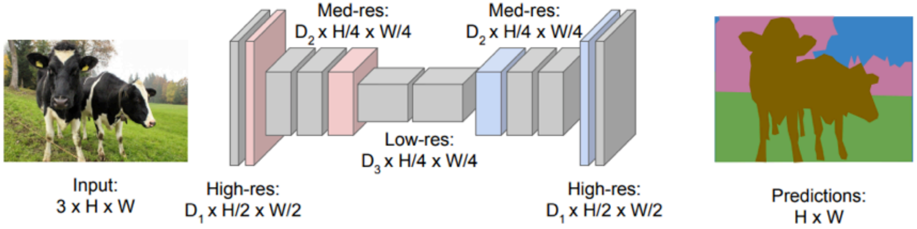

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!
pixel-wise softmax activation
Conv
Conv
Conv
Conv
argmax
Input:
3 x H x W
Convolutions:
D x H x W
feature extraction
Scores:
C x H x W
Predictions:
H x W
final output retains original image dimensions

**Downside:** Preserving image dimensions throughout entire network will be computationally expensive.

Image source: internet

# Semantic Segmentation-2

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



**Solution:** Make network deep and *work at a lower spatial resolution* for many of the layers.

Image source: internet

# Semantic Segmentation-3

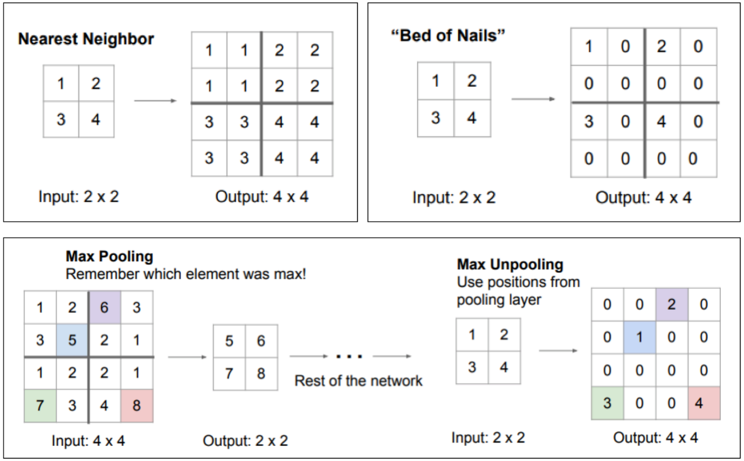


Image source: internet

## Object identification: R-CNN

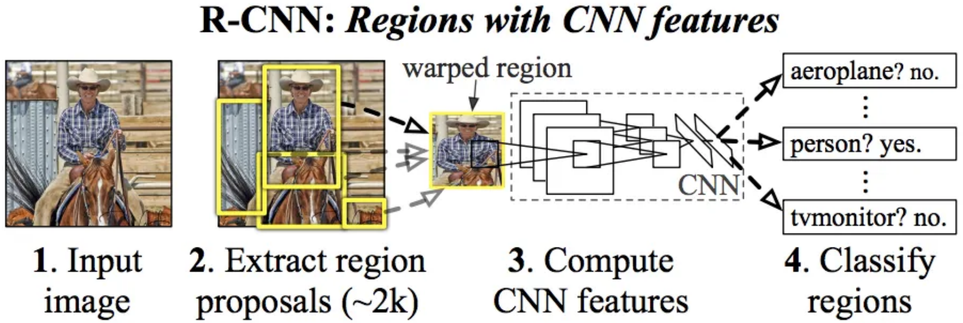


Image source: internet

# Object identification: R-CNN

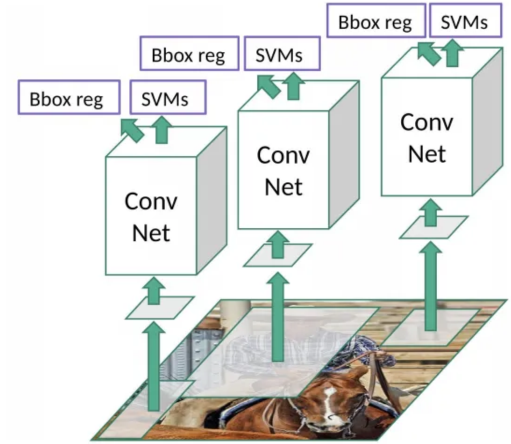


Image source: internet

# Object identification: Fast R-CNN

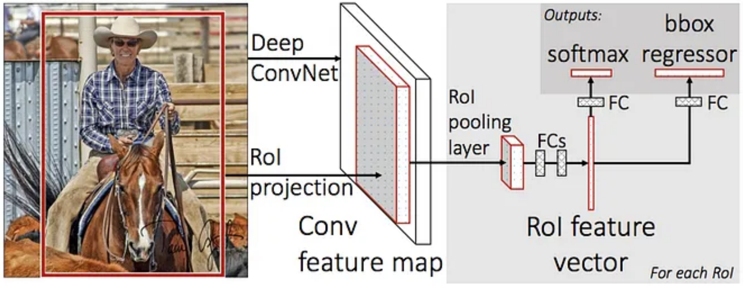


Image source: internet

# Object identification: Faster R-CNN

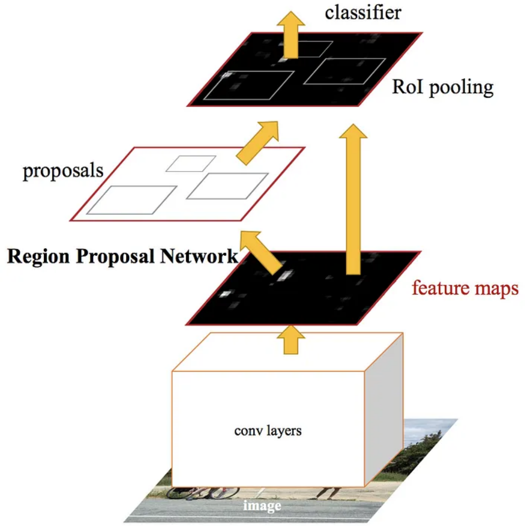
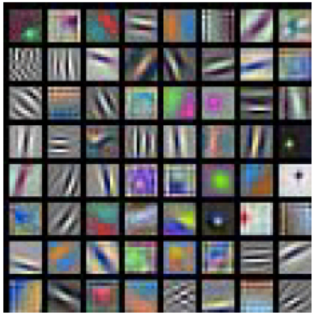


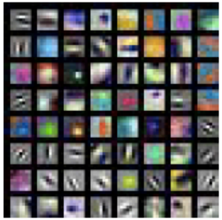
Image source: internet

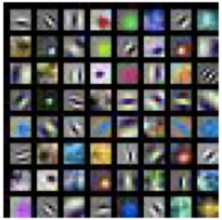
## CNN: Visualization

- Visualization of fliters
- Visualization of last layer features
- Visualization of activations
- Identifying important pixels
- Saliency map
- Image synthesis
- Style transfer

## CNN: Visualization of Filters


AlexNet:
64 x 3 x 11 x 11


ResNet-18:
64 x 3 x 7 x 7


ResNet-101:
64 x 3 x 7 x 7

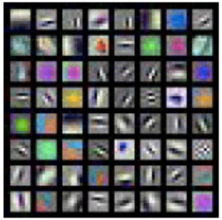
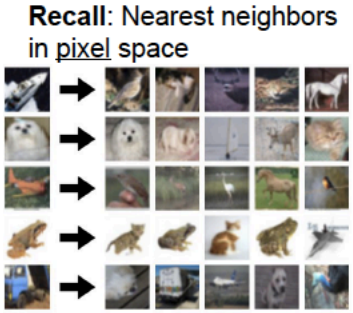
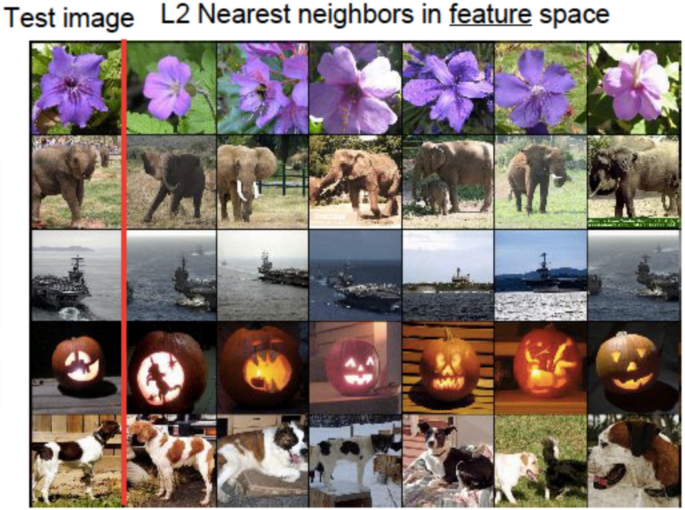

DenseNet-121:
64 x 3 x 7 x 7

Image source: https://srdas.github.io/DLBook/ConvNets.html

# CNN: Nearest neighbour





Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figures reproduced with permission.

Image source: https://srdas.github.io/DLBook/ConvNets.html

# t-SNE: Last layer feature

- t-distributed stochastic neighbor embedding

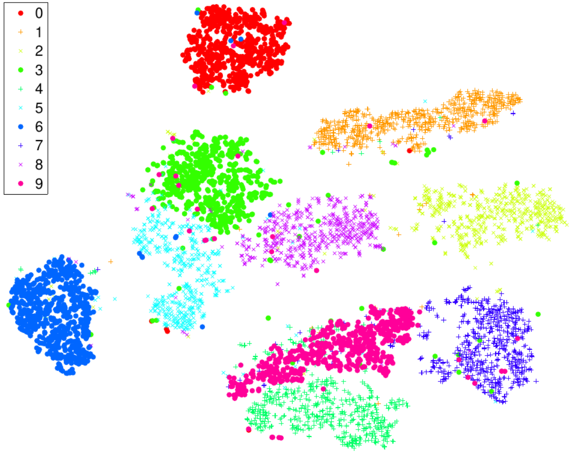
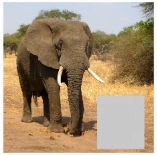
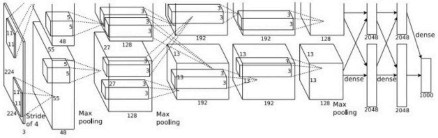


Image source: https://srdas.github.io/DLBook/ConvNets.html

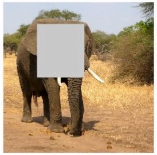
# Saliency via Occlusion

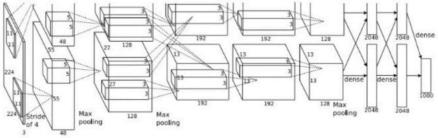
Mask part of the image before feeding to CNN,
check how much predicted probabilities change





P(elephant) = 0.95





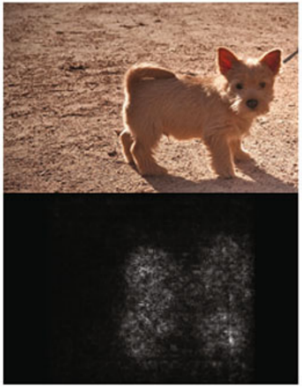
P(elephant) = 0.75

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Image source: https://aman.ai/cs231n/visualization/

## CNN: Saliency via Backprop

- Find $\frac{\partial o}{\partial x_i}$. Eg. for imagenet, we will have 224x224x3
- Take absolute value and max over RGB channel
- Image will reduced to 224x224x1



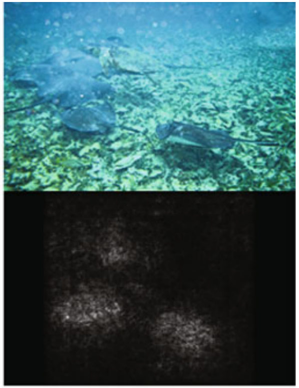


Image source: Neural Networks and Deep Learning

# CNN: Guided Backpropagation

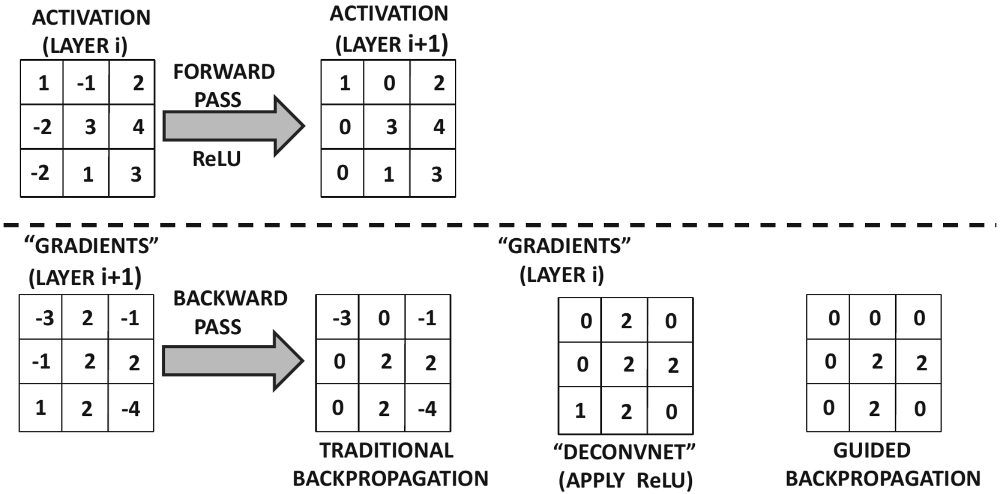
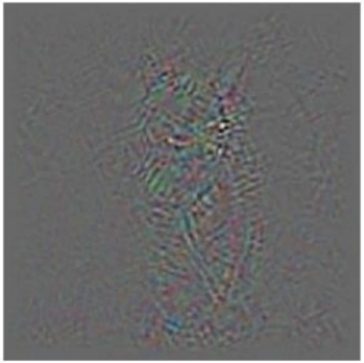


Image source: Neural Networks and Deep Learning

# Guided backpropagation


Backprop

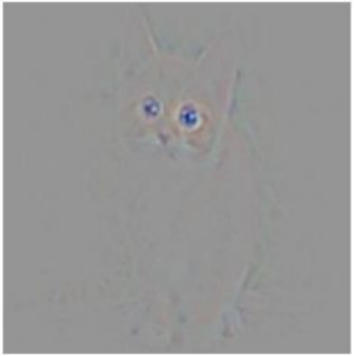

Guided Backprop

Image source: internet

# Guided backpropagation

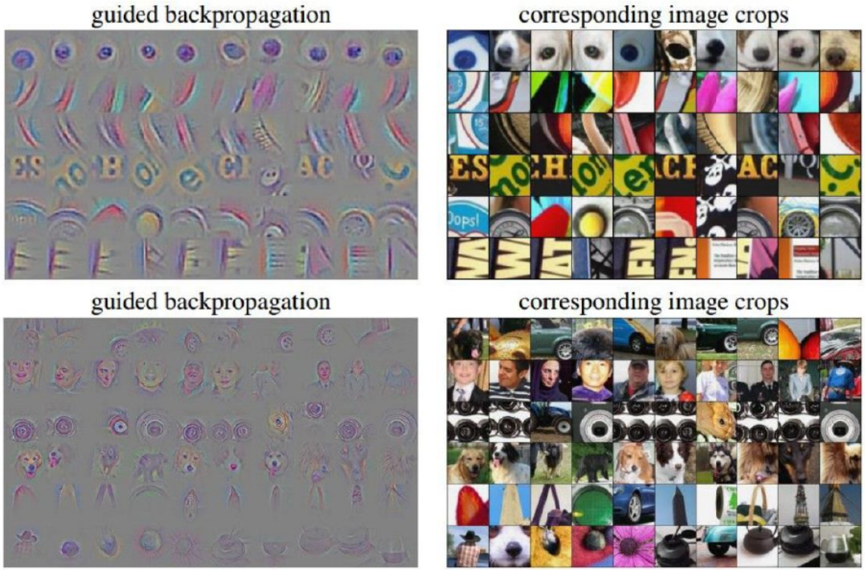


Image source: Neural Networks and Deep Learning

## Gradient Ascent

- What kind of image maximizes an activation in a neuron?
- What kind of an image achieves the maximum score for a specific category/class?
- With gradient ascent, we are trying to learn the image that maximizes the activations for a particular class: $I^* = \arg\max_I f(I) + R(I)$, $f$ - neuron value, $R$ - regularizer
- In summary, here are the steps of gradient ascent:
  - Initialize image to zeros.
  - Forward image to compute current scores.
  - Backprop to get gradient of the neuron activation with respect to the input image pixels.
  - Make a small update to the image.

# Synthetic images

- Example images having high confidence score on state of the art neural networks

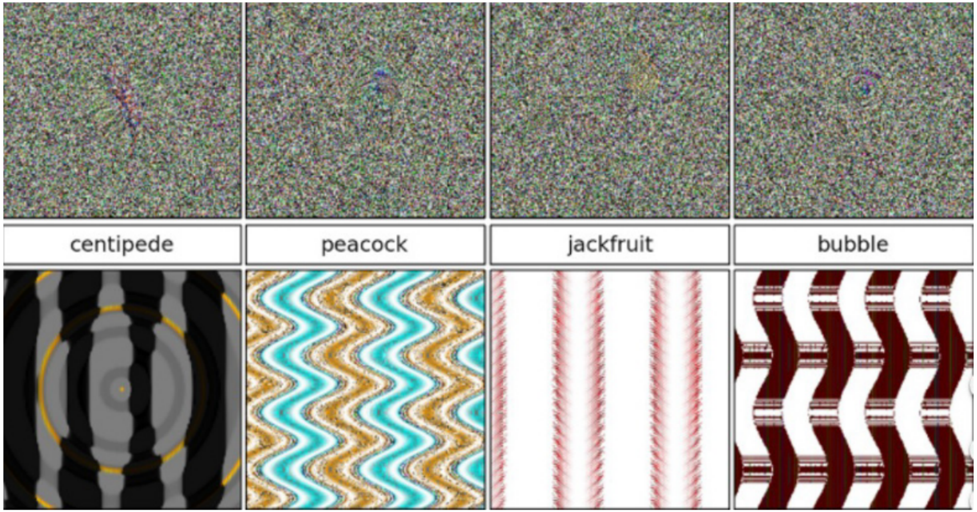


Image source: The Data Science Design Manual

# Fantasy image

- Find an input that maximizes a particular neuron at the output (before softmax)

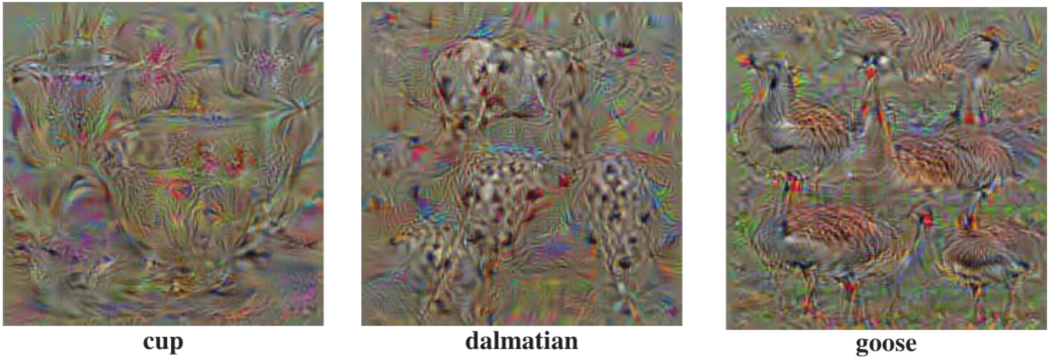


Image source: Neural Networks and Deep Learning

# Adversarial Perturbations

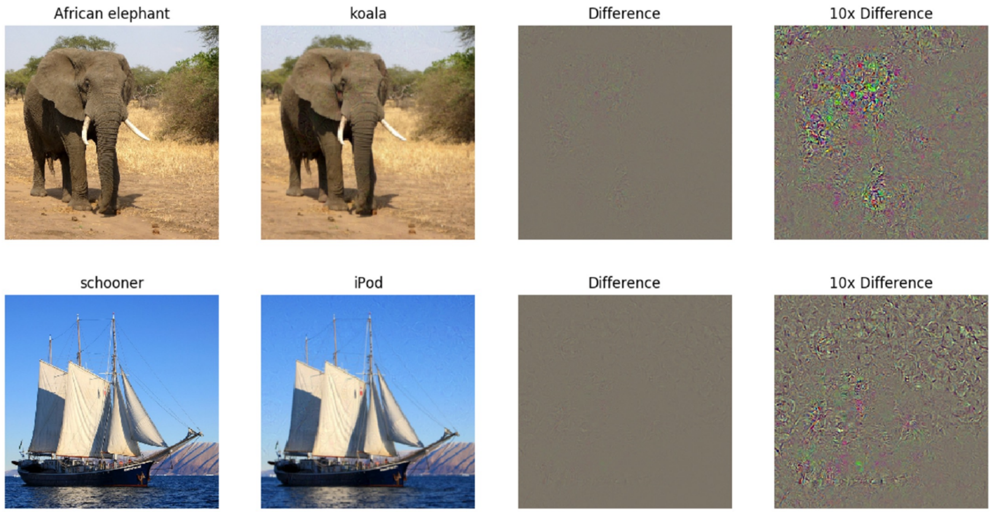


Image source: internet

# Universal Adversarial Perturbations

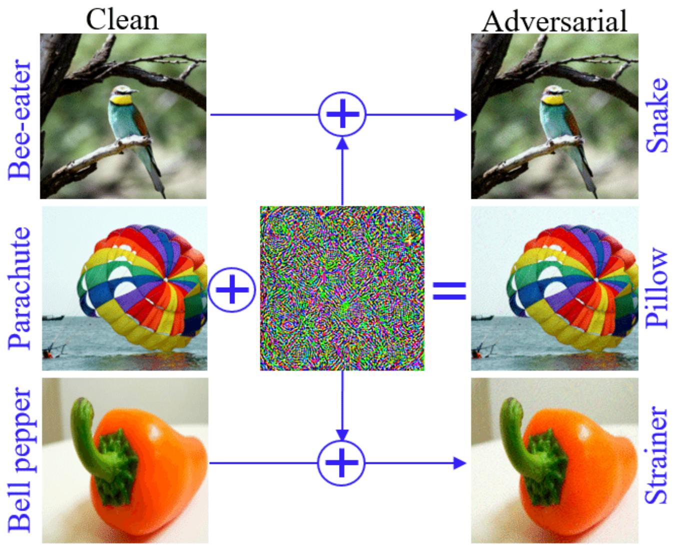


Image source: internet

## Deep Dream-1

- Start with a trained CNN. Weight values are fixed. Select some image from the test set.
- Do a forward pass through the network and compute the activations at all the nodes, until a chosen layer.
- Set the gradients at each node of the chosen layer equal to the activation at that node, i.e., $\frac{\partial L}{\partial h_i} = h_i$
- Using the Backprop algorithm compute the gradients $\frac{\partial L}{\partial x_{ijk}}$
- Change the pixel value to $x_{ijk} = x_{ijk} + \eta \frac{\partial L}{\partial x_{ijk}}$
- Add the training set mean image to the final pixel values to obtain the final image.

# Deep Dream-2

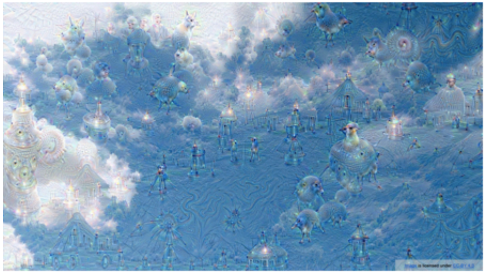


Image source: internet

# Deep Dream-3

Image source: internet

# Feature Inversion

- Generation of image whose feature vector is specified
- $\mathcal{L} = \|\phi(X) - \phi_0\| + \lambda R(X)$, $\phi_0$ is the feature vector

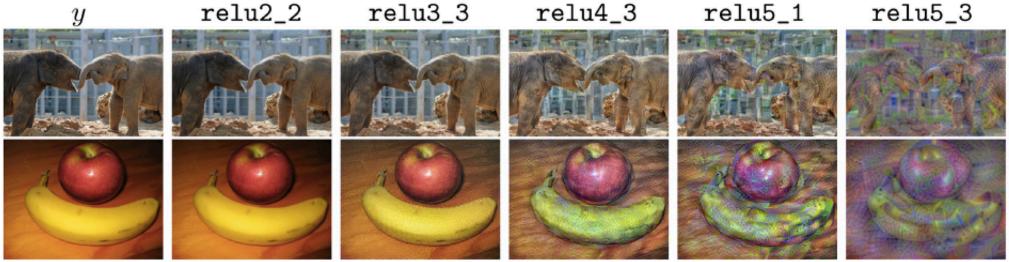


Image source: internet

## Resources

- URL: `https://cs.stanford.edu/people/karpathy/convnetjs/`
- URL: `https://playground.tensorflow.org`