ONLY GOD

1402-2023

mansoori@shirazu.ac.ir

Neural Network & Deep Learning

Convolutional NN

CSE & IT Department

School of ECE

Shiraz University

Convolutional NN (CNN)(ConvNet)

mansoori@shirazu.ac.ir

- A type of feed-forward neural network
- Connectivity pattern between its neurons is inspired by the organization of the animal visual cortex
- Individual cortical neurons respond to stimuli in a restricted region of space known as receptive field
- Receptive fields of different neurons partially overlap such that they tile visual field

mansoori@shirazu.ac.ir

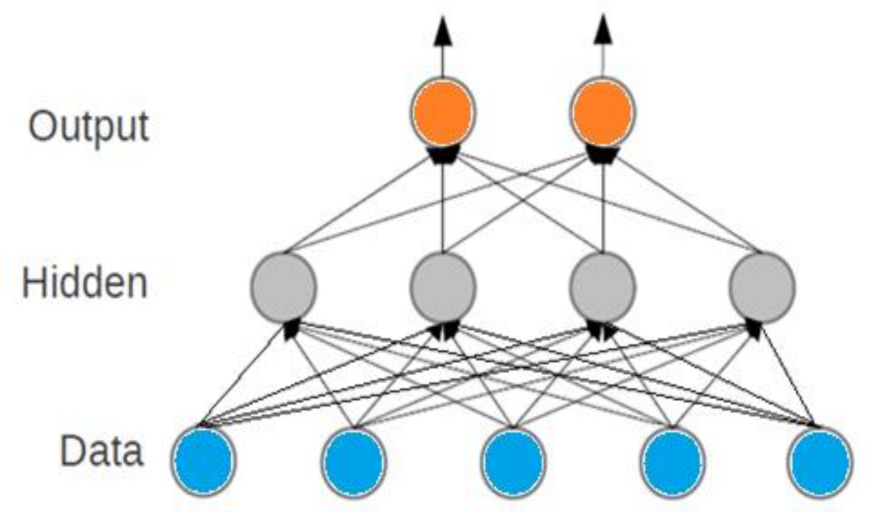
ConvNet

- Response of an individual neuron to stimuli within its receptive field can be approximated mathematically by a convolution operation
- Convolutional networks were inspired by biological processes and are variations of MLPs, designed to use minimal amounts of preprocessing
- Wide applications in image and video recognition, recommender systems and natural language processing

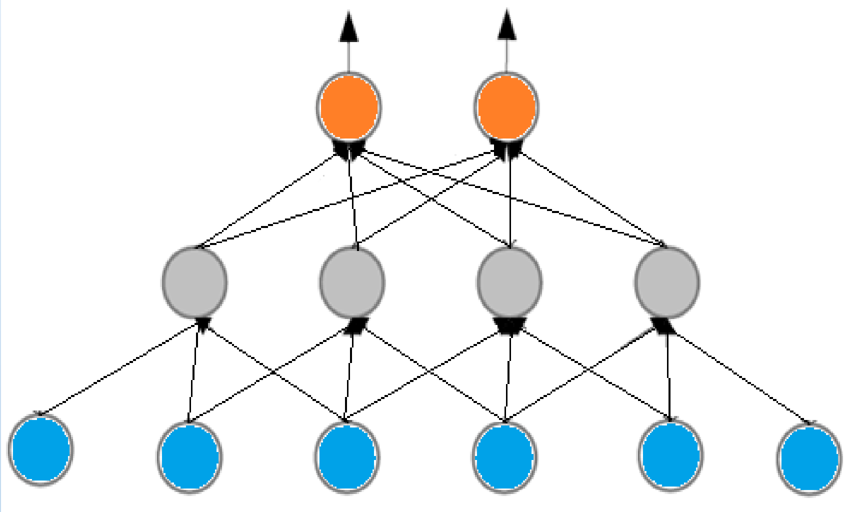
mansoori@shirazu.ac.ir

ConvNet

common NN



convolutional NN



- Each hidden neuron applies the same localized, nonlinear filter to input
- Like most NNs, ConvNets are trained with a version of back-propagation algorithm

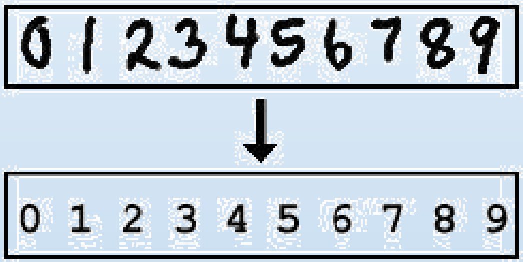
mansoori@shirazu.ac.ir

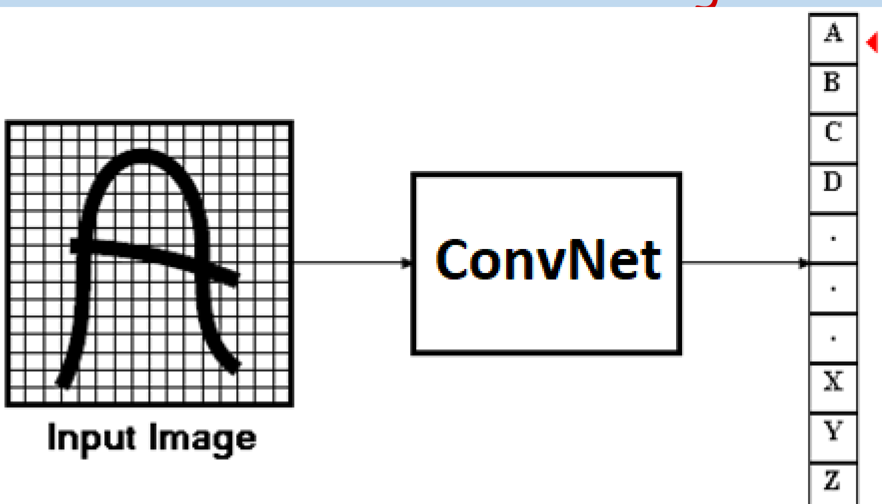
ConvNet History

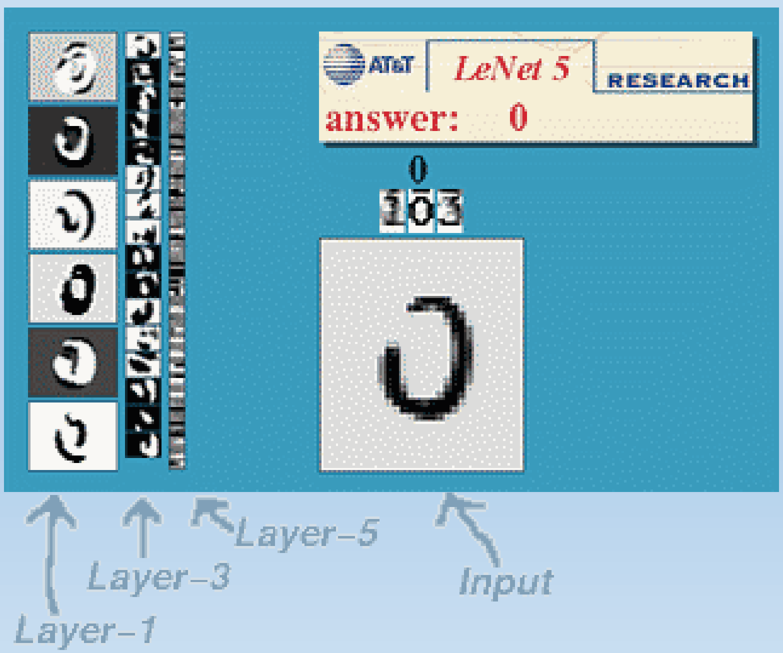
- Concept of ConvNet introduced in 1995

Yann LeCun

Yoshua Bengio







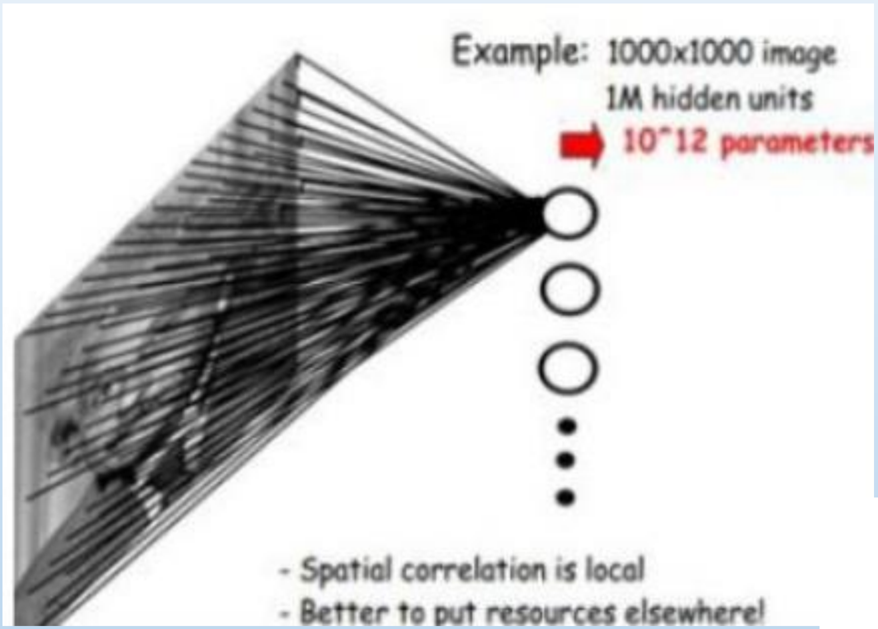
mansoori@shirazu.ac.ir

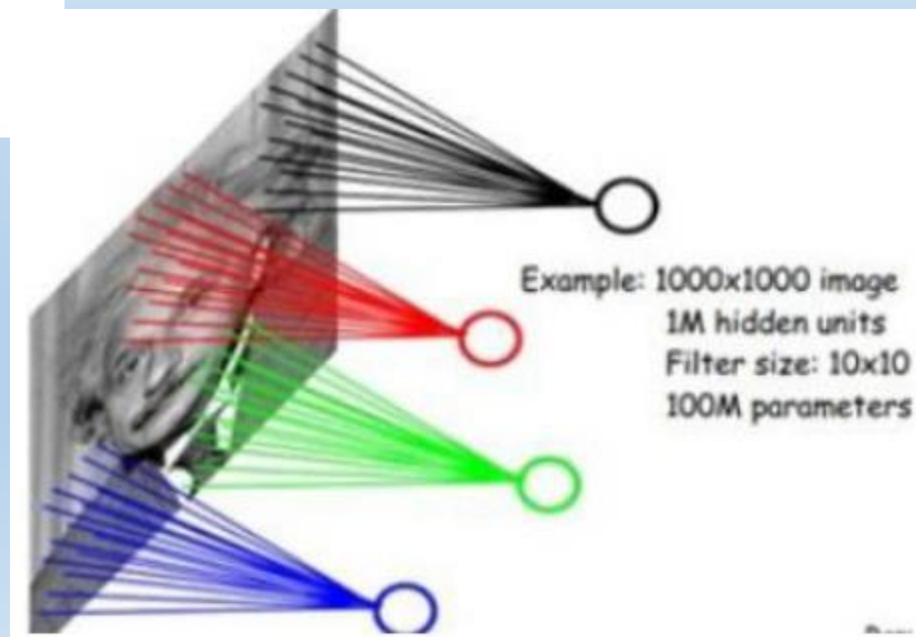
ConvNet Motivation

- Natural images are stationary
  - Statistics of one part of an image is the same as any other part
  - Features which are learned at one part of an image can also be applied to its other parts
  - The same features can also be used at all locations
- These models have revolutionized speech and object recognition

mansoori@shirazu.ac.ir

ConvNet Motivation





mansoori@shirazu.ac.ir

ConvNet Features

- Convolution leverages three important ideas that can help improve a machine learning system:
  - Sparse interactions (sparse connectivity/weights)
  - Parameter sharing
  - Equivariant representations
- ConvNet can implicitly extract relevant features
- It can extract topological properties from an image

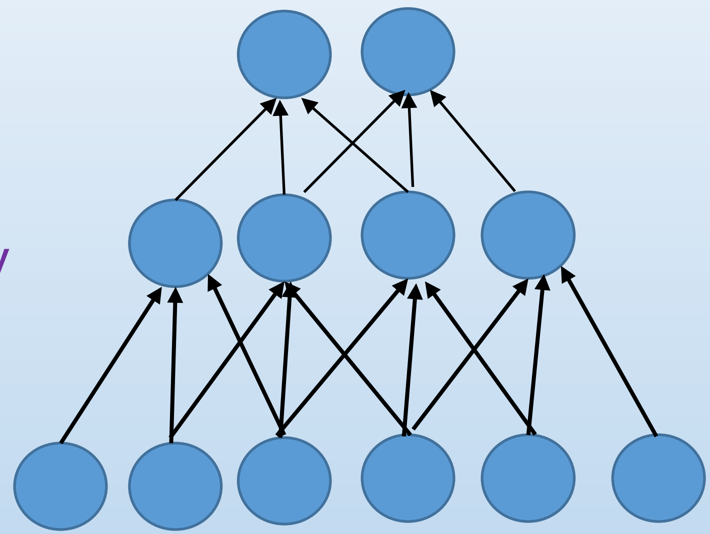
mansoori@shirazu.ac.ir

Recap of ConvNet

pooling

non-linearity

convolution



Input data

↓

Convolution

↓

Non-linearity

↓

Pooling

↓

Feature maps

- Feed-forward
  - Convolving input
  - Non-linearity in activation function
  - Pooling

- Supervisedly trained by back-propagating error

1D Convolution

mansoori@shirazu.ac.ir

Correlation (Similarity):

$\vec{x}$: x_1 x_2 x_3

$\vec{w}$: w_1 w_2

$$z\_in_1 = w_1x_1 + w_2x_2$$

$\vec{z}$: z_1 z_2

$\vec{x}$: x_1 x_2 x_3

$\vec{w}$: w_1 w_2

$$z\_in_2 = w_1x_2 + w_2x_3$$

$$z\_in_j = \sum_{i=1}^{F} w_i \, x_{j+i-1}$$

where $F = |\vec{w}|$: size of filter

Convolution:

$\vec{x}$: x_1 x_2 x_3

$\vec{\tilde{w}}$: w_2 w_1

$$z\_in_1 = w_2x_1 + w_1x_2$$

$\vec{z}$: z_1 z_2

$\vec{x}$: x_1 x_2 x_3

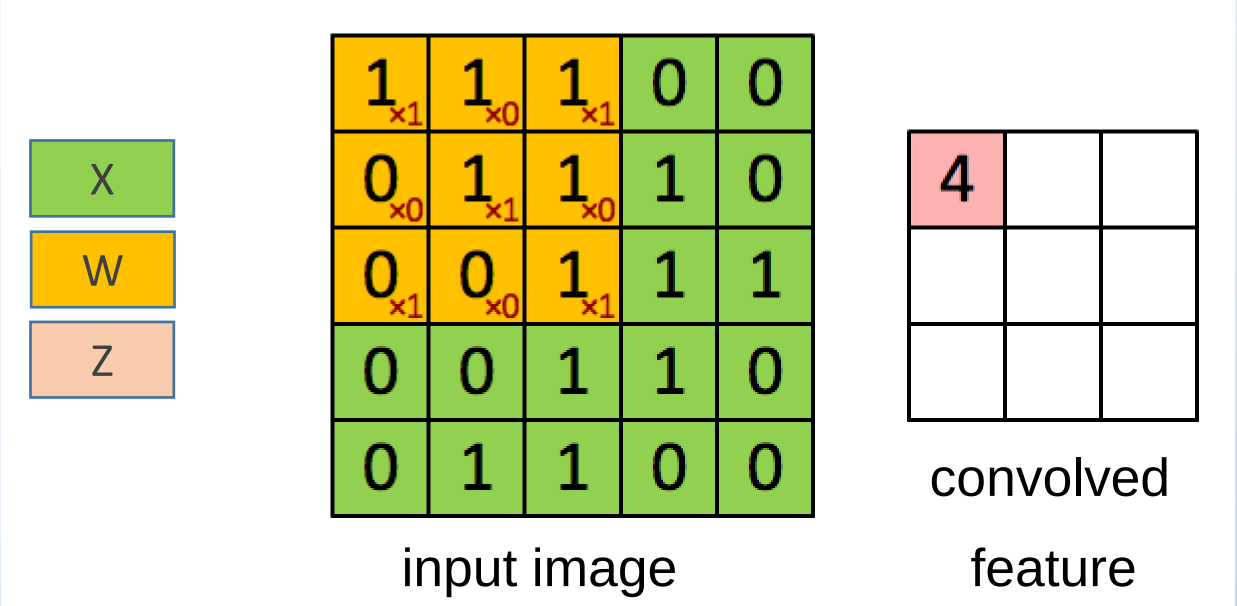
$\vec{\tilde{w}}$: w_2 w_1

$$z\_in_2 = w_2x_2 + w_1x_3$$

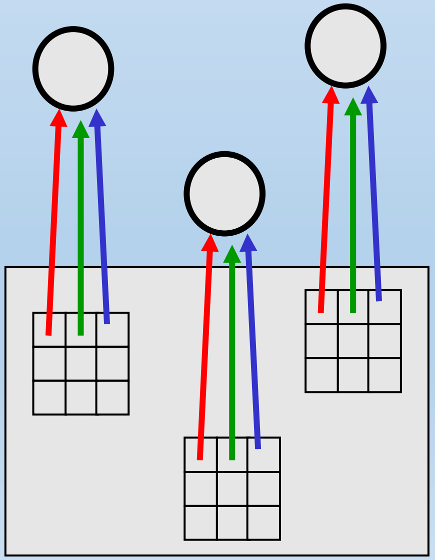
$$z\_in_j = \sum_{i=1}^{F} \tilde{w}_{F-i+1} \, x_{j+i-1}$$

mansoori@shirazu.ac.ir

2D Convolution



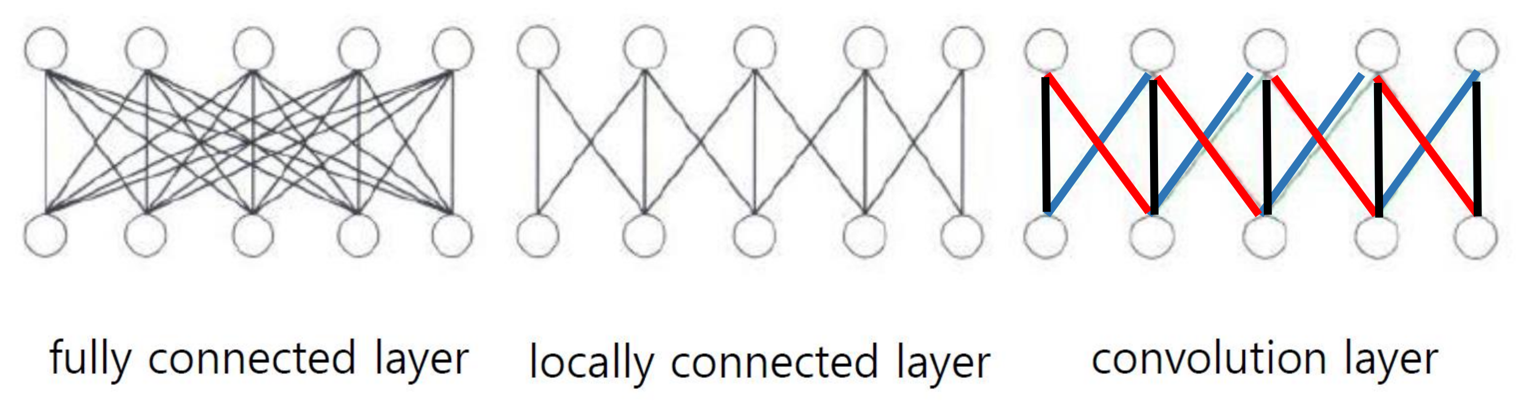
- The same colored connections all have the same weight



Convolution (Weight Sharing)

mansoori@shirazu.ac.ir

- Connectivity and weight sharing depends on layer



all different weights | all different weights | shared weights

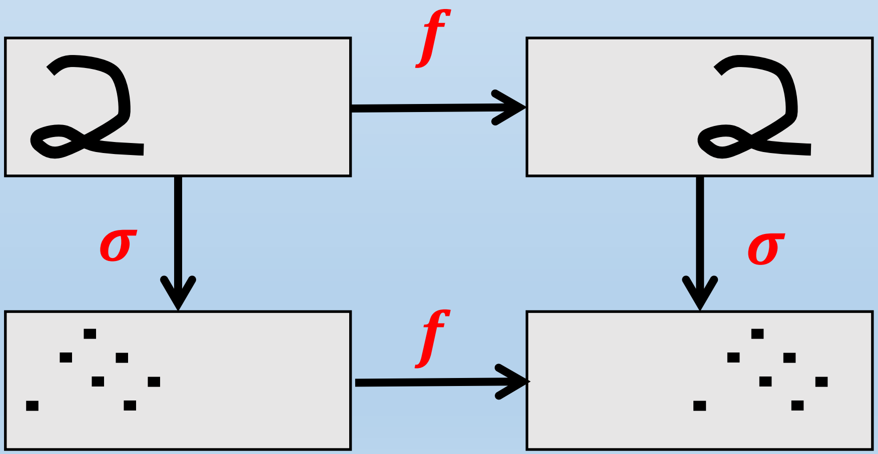
- Convolution layer has much smaller number of parameters by local connection and weight sharing

mansoori@shirazu.ac.ir

Convolution (Equivariance)

- Convolutional structure of ConvNets preserves symmetries of input data (outputs of ConvNets retain symmetries of inputs)
- For any symmetry operation $\sigma(.)$, applying σ to input data and then passing it through model $f(.)$ would be the same as applying σ to output of model f: $f(\sigma(x)) = \sigma(f(x))$

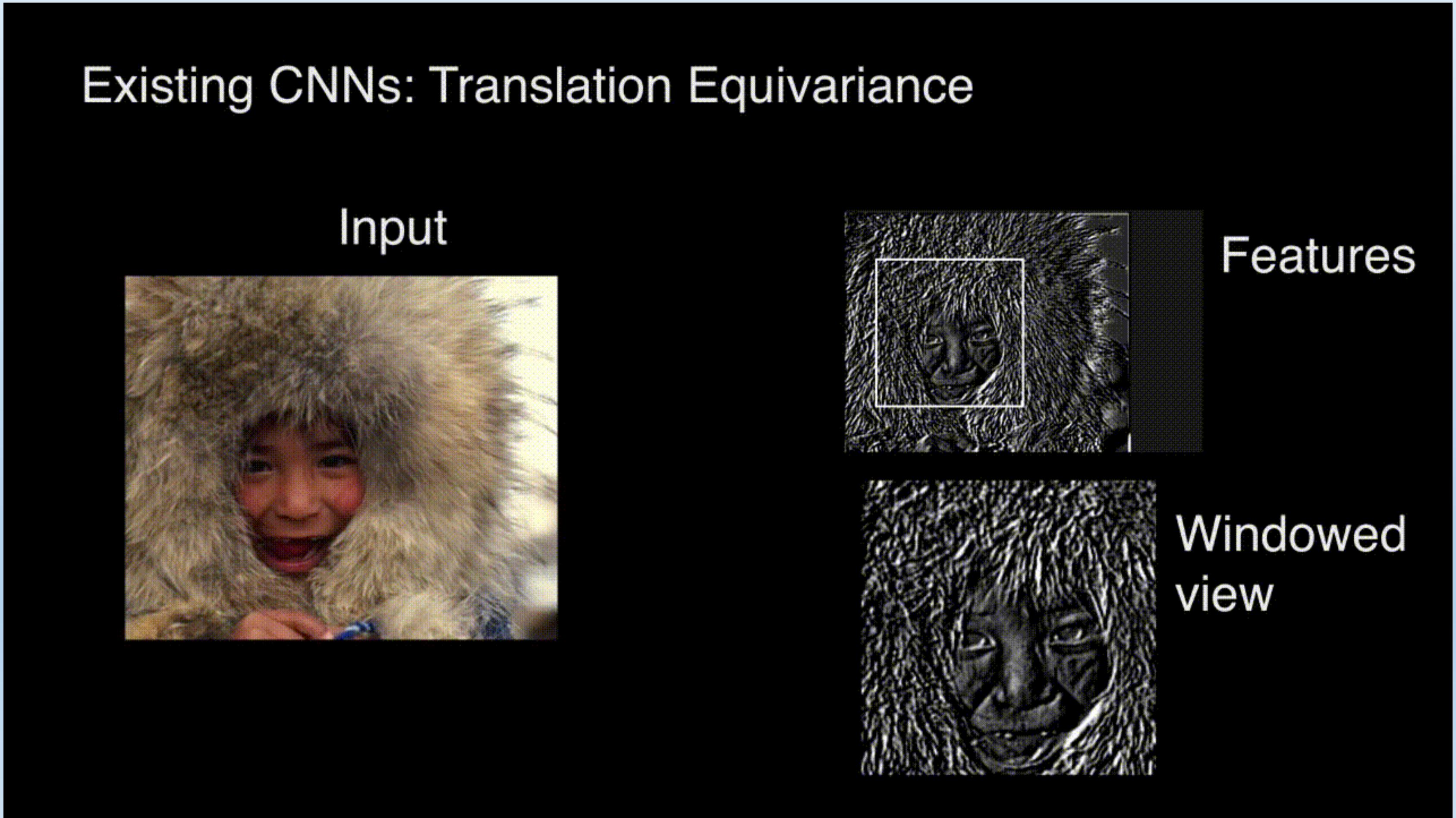
$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "\sigma"'] & X' \arrow[d, "\sigma"] \\
Y \arrow[r, "f"] & Y'
\end{tikzcd}$$



- Model f is equivariant with respect to symmetry operation σ

Convolution (Equivariance)

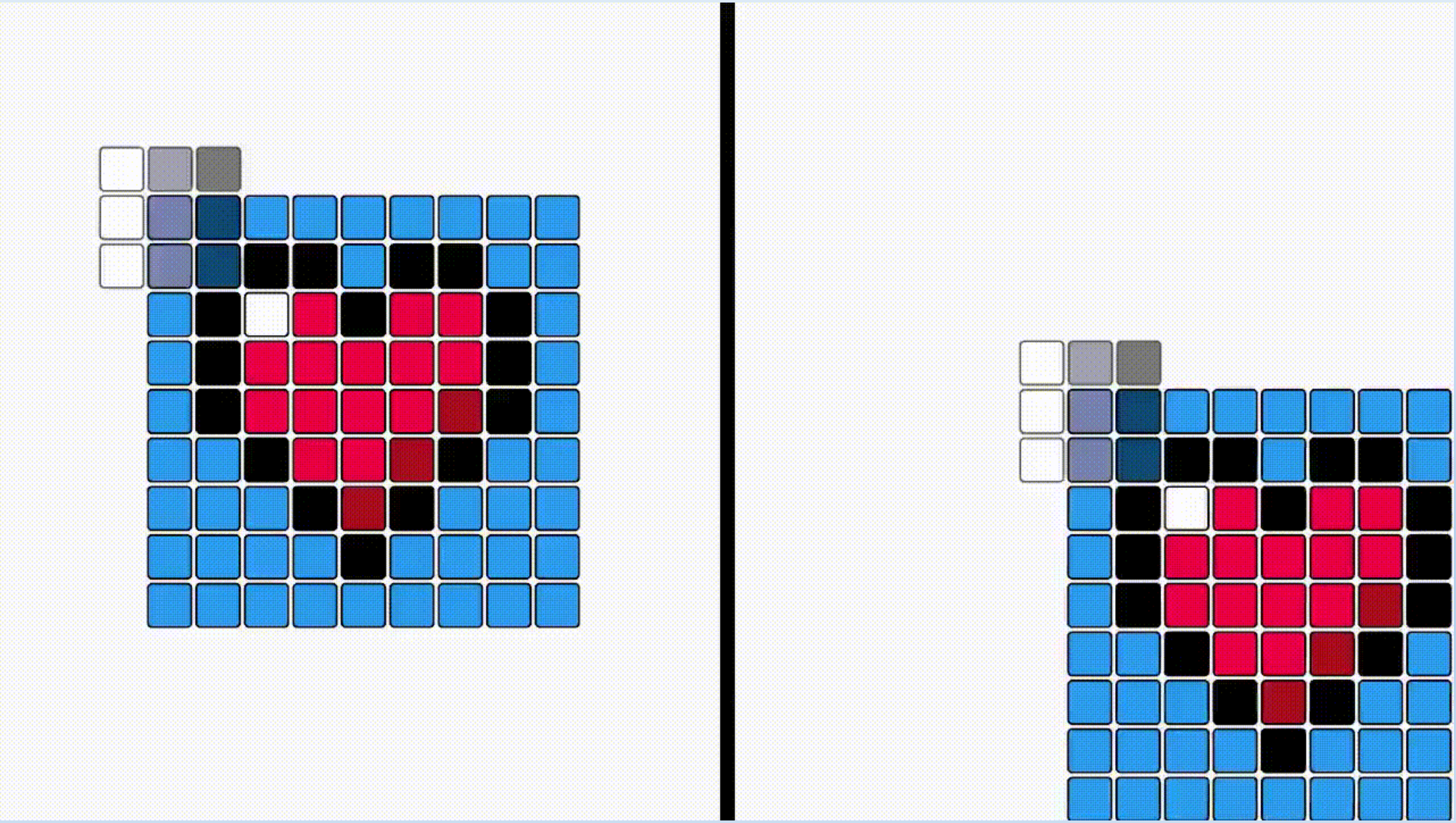
- Equivariance of ConvNets with respect to translation
- Shift to input image corresponds to shift of output features



Convolution (Equivariance)

mansoori@shirazu.ac.ir

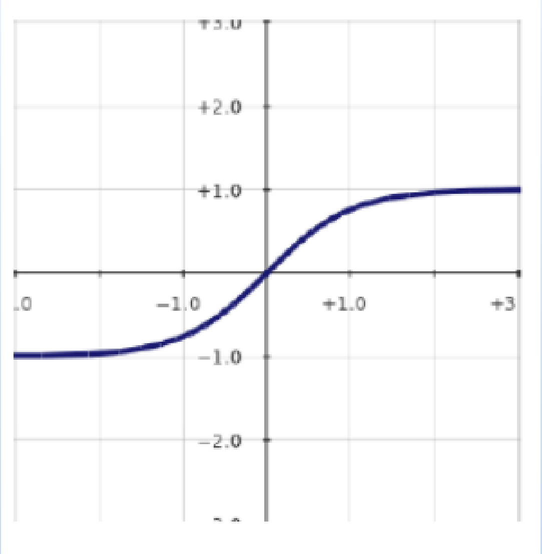
- Equivariance of ConvNets with respect to translation



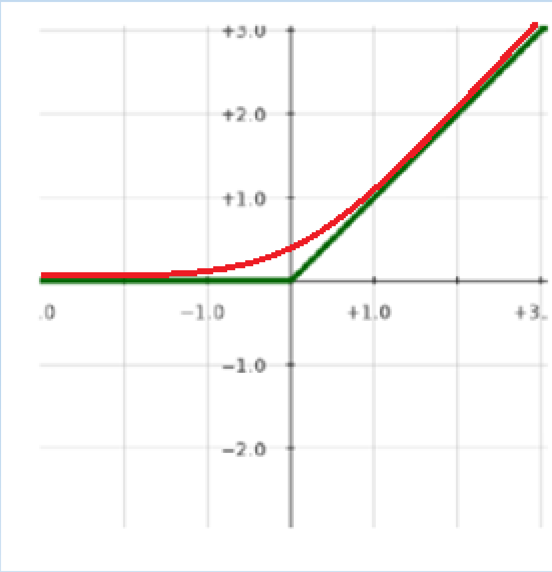
Non-linearity in ConvNet

mansoori@shirazu.ac.ir

- Bipolar sigmoid
  - $f(z\_in) = \tanh(z\_in) = \frac{1-e^{-z\_in}}{1+e^{-z\_in}}$
  - Slow to train
  - Has vanishing gradient problem
  - Commonly used



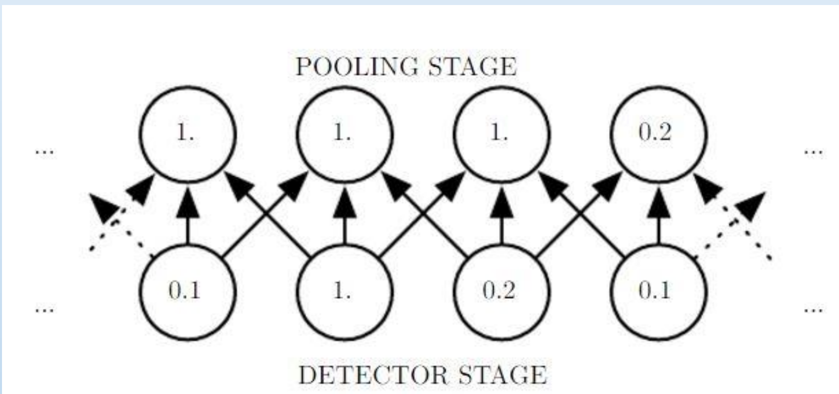
- Rectifier linear unit (reLU)
  - $f(z\_in) = \text{reLU}(z\_in) = \max(z\_in, 0)$
  - $f(z\_in) = \ln(1 + e^{x})$
  - Quick to train
  - Less vanishes gradient
  - Recently used

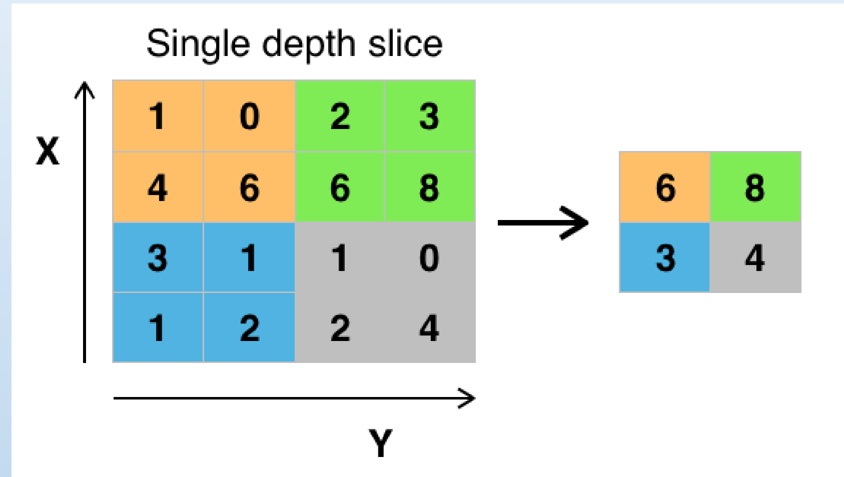


mansoori@shirazu.ac.ir

Popular Pooling Functions

- Maximum of a rectangular neighborhood (max pooling)





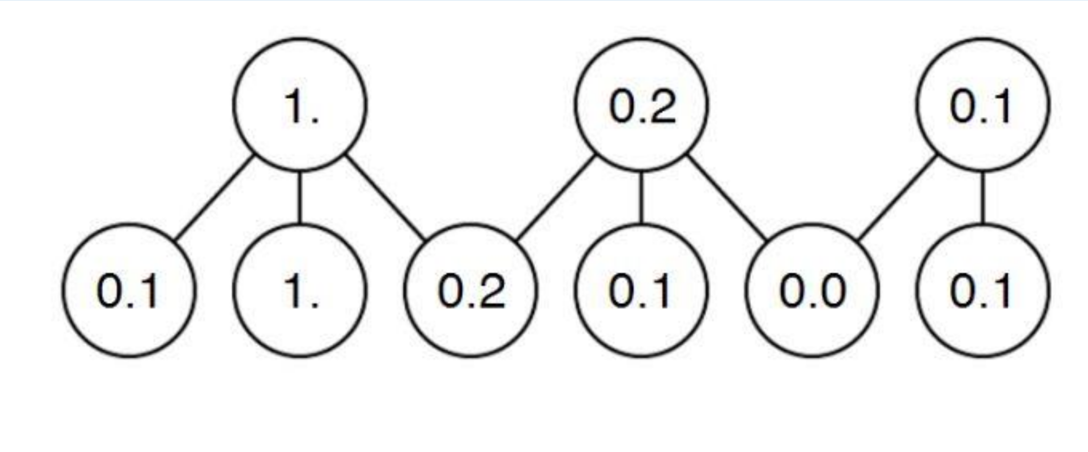
- Weighting of a rectangular neighborhood
- $l2$ norm of a rectangular neighborhood
- Average of a rectangular neighborhood
- Weighted average based on distance from central pixel

Pooling with Down-sampling

mansoori@shirazu.ac.ir

Down-sampling (subsampling) during pooling

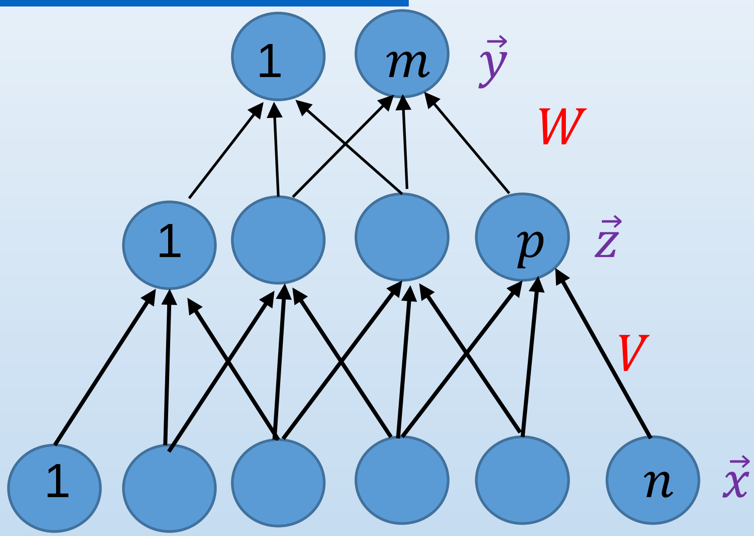
- Max-pooling with a pool width of 3 and a stride between pools of 2



- This reduces the representation size by a factor of 2,which reduces the computational and statistical burden

mansoori@shirazu.ac.ir

Back-propagation in ConvNet



- Output layer:

  for $k = 1..m$

  $$\delta_k^{\mathrm{O}} = -(t_k - y_k)\, f^{\mathrm{O}\prime}(y\_in_k)$$

- Pooling weights W:
  - To propagate the error through the pooling layer by calculating the error w.r.t to each unit incoming to the pooling layer

  for $k = 1..m$

  $$J_k = \mathrm{upsample}(k, 1..p)$$

  $$\underset{j \in J_k}{\text{All}}\ \Delta w_{jk} = -\alpha\, \delta_k^{\mathrm{O}}\, z_j$$

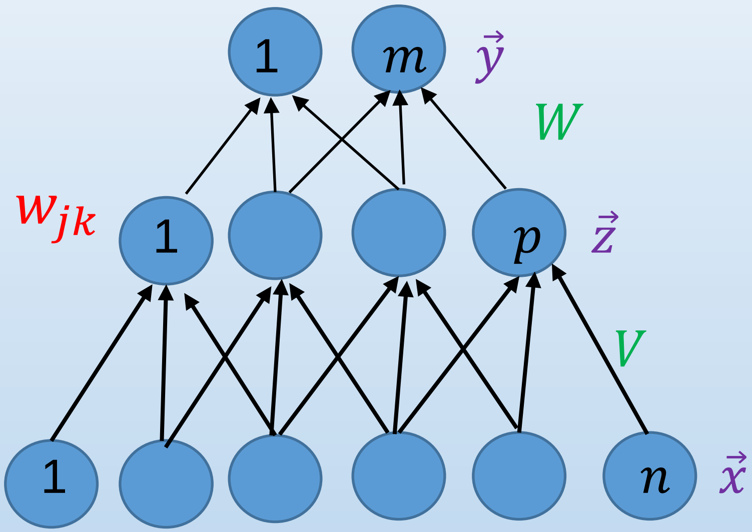
  - $\mathrm{upsample}(k, 1..p)$: neurons in $Z_1..Z_p$ connected to Y_k

Back-propagation in ConvNet

- Hidden layer:

  for $j = 1..p$

  $$\delta_j^{\mathrm{H}} = f^{\mathrm{H}\prime}(z\_in_j) \sum_{k=1..m|j \in J_k} \delta_k^{\mathrm{O}} \, w_{jk}$$



- Convolutional weights V:

  for $j = 1..p$

  $$I_j = \mathrm{upsample}(j, 1..n)$$

  $$\underset{i \in I_j}{\mathrm{All}} \; \Delta v_{ij} = -\alpha \, \delta_j^{\mathrm{H}} \, x_i$$

ConvNet in MATLAB

mansoori@shirazu.ac.ir

```
layers = [ imageInputLayer([28 28 1])
           convolution2dLayer(5, 1)
           reluLayer
           maxPooling2dLayer(2, 'Stride', 2)];

options = trainingOptions('sgdm', 'MaxEpochs',20,
'InitialLearnRate',1e-4);

net = trainNetwork(img, layers, options);

y = classify(net, img);
```