Convolutional Neural Network

Vlado Menkovski

Eindhoven University of Technology

v.menkovski@tue.nl

March 29, 2018

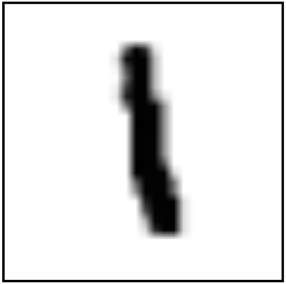
Convolutional Neural Networks

Overview

1. Data with spatial correlation
2. Convolutional neural network
3. Training
4. CNN Architecture

Image data

Image data



$$\simeq \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Speech data

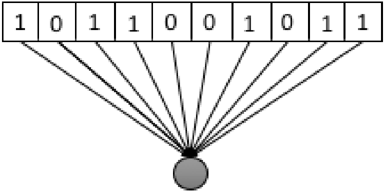
Spatial data

- High Dimensionality
- Local Correlations
- Convolutional Neural Networks (CNN) - utilize the local correlation property

Spatial correlation

1	0	1	1	0	0	1	0	1	1

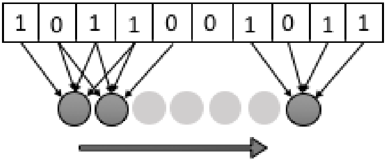
Spatial correlation



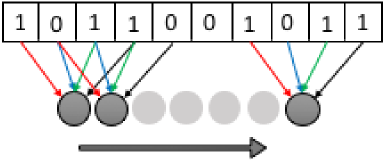
Spatial correlation

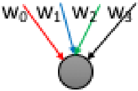
0	1	0	1	1	0	0	0	0	0

Spatial correlation

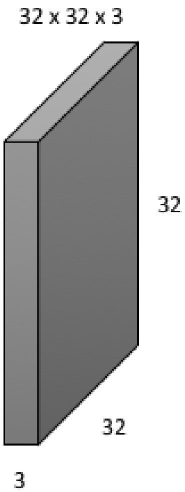


Parameter re-use





Input visualization



Layer architectures

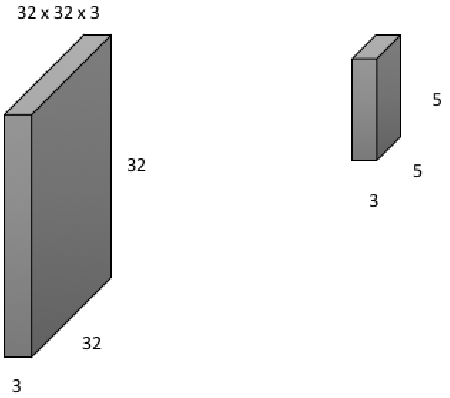
ConvNets operate on tensors

Take tensor of activations and produce tensor of activations

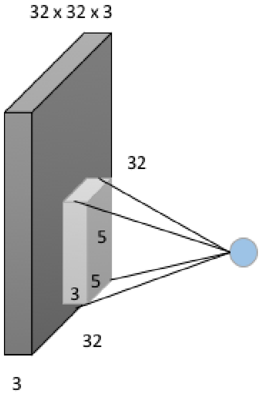
Convolution Dimensionality

- for 1D convolutions the volumes are 2D matrix
- for 2D convolution the volumes of activations are 3D tensor
- for 3D convolutions the volumes are 4D tensor

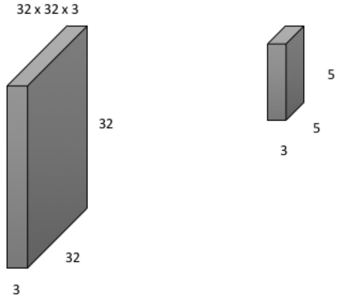
CNN activation and filter



CNN activation and filter



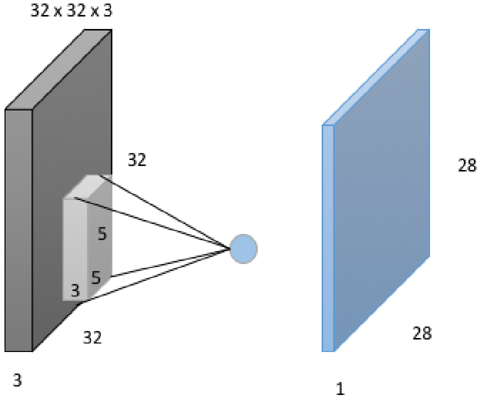
CNN activation and filter



The depth of the filter is equal to the depth of the image

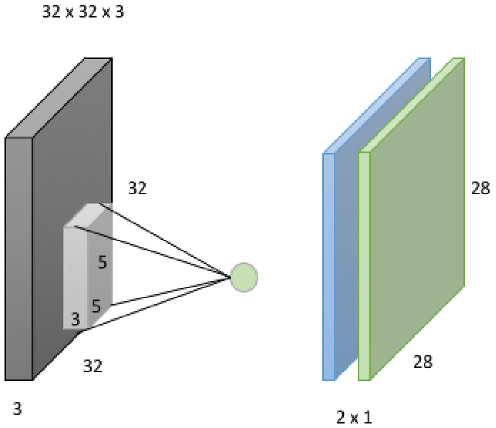
Neuron computes: Afine transformation + non-linearity

CNN activation and filter



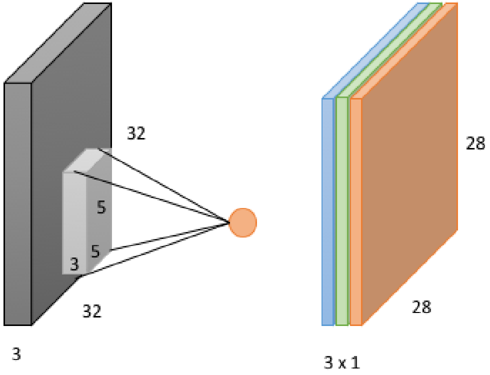
Fully convolving the input → produces an activation map

CNN activation and filter



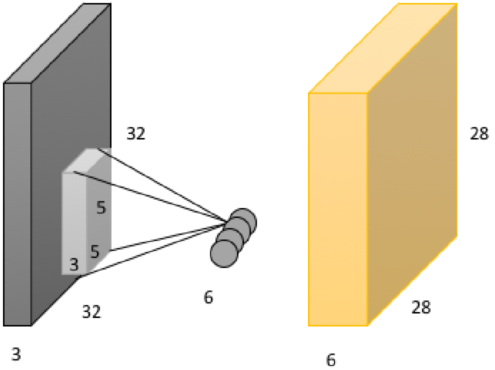
Second neuron

CNN activation and filter



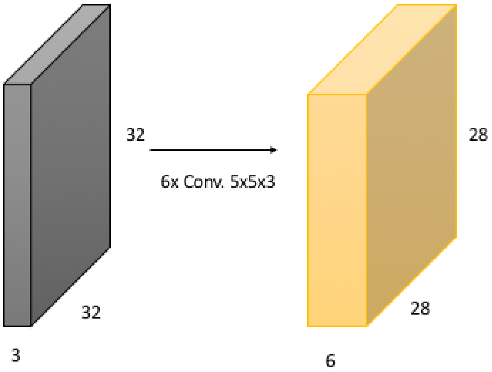
Third neuron

CNN activation and filter

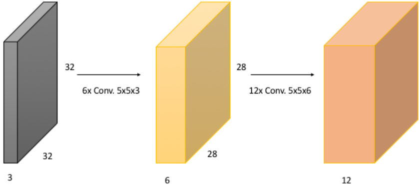


Combined representation

CNN activation and filter



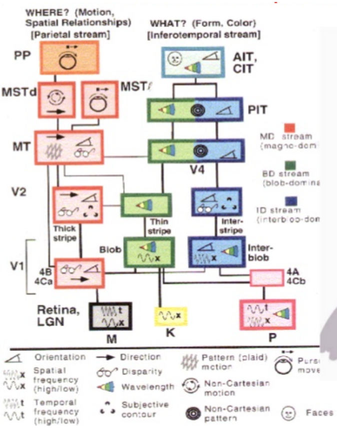
CNN activation and filter

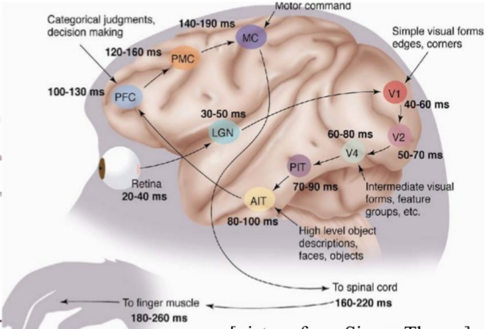


Convolutional layer

- Accepts:
  - $W_1 \times H_1 \times D_1$
- Outputs:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$
  - $D_2 = K$
- Where:
  - F is the filter size
  - P is the padding size
  - S is the stride
  - K is the layer depth (number of neurons)

CNN Biological inspiration





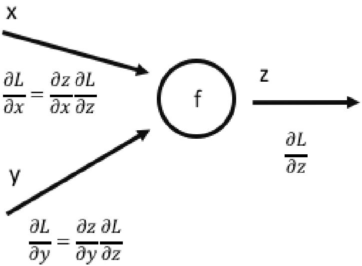
[picture from Simon Thorpe]

[Gallant & Van Essen]

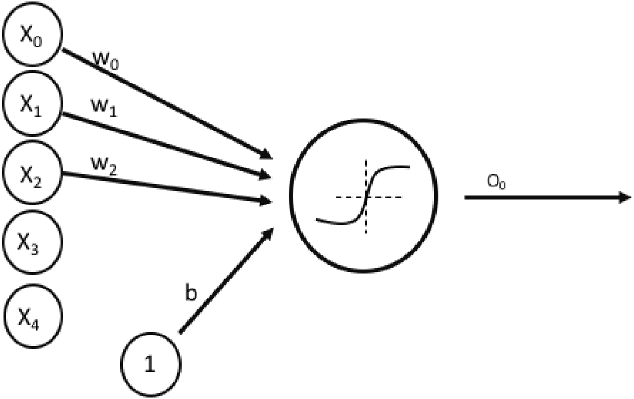
Training: Gradient Decent

$$\frac{\partial L}{\partial \theta}$$

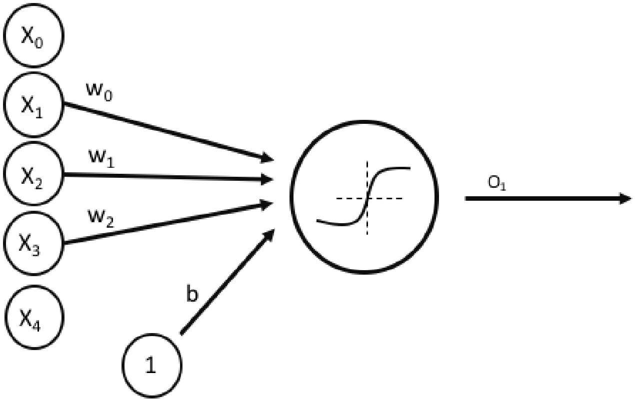
Back propagation



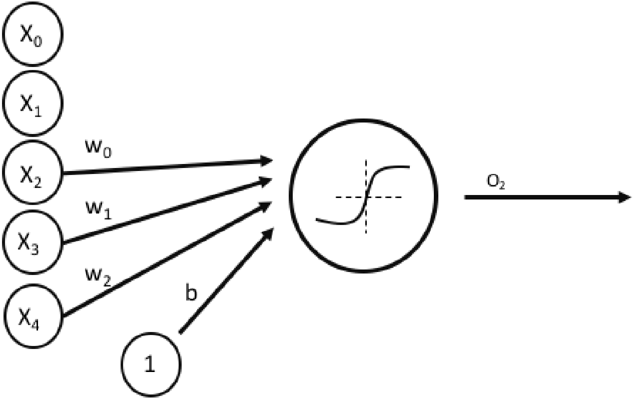
Convolutional Neuron (Step 1)



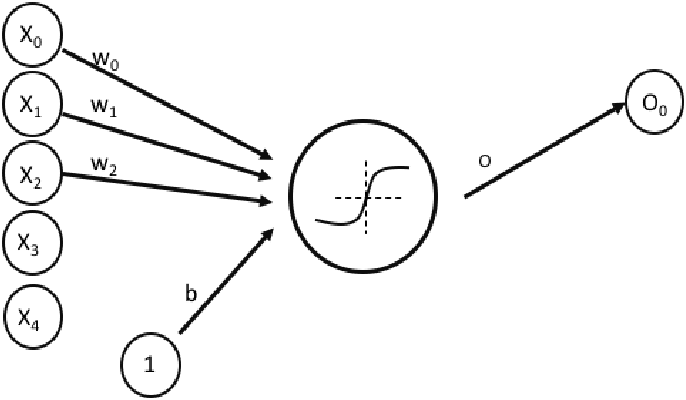
Convolutional Neuron (Step 2)



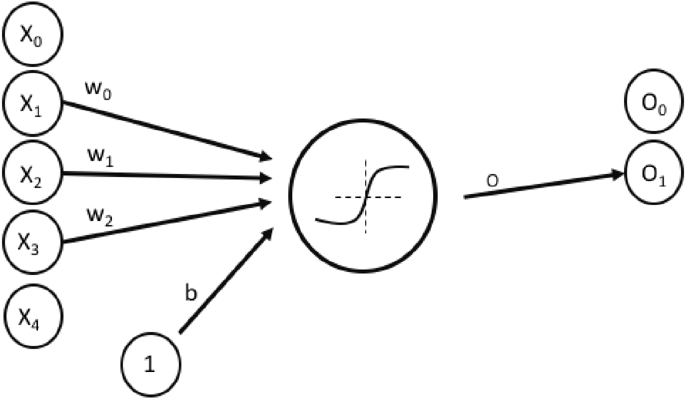
Convolutional Neuron (Step 3)



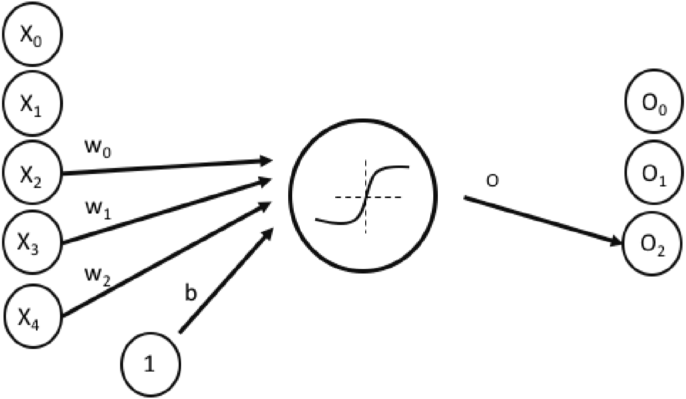
Convolutional Neuron Forward pass



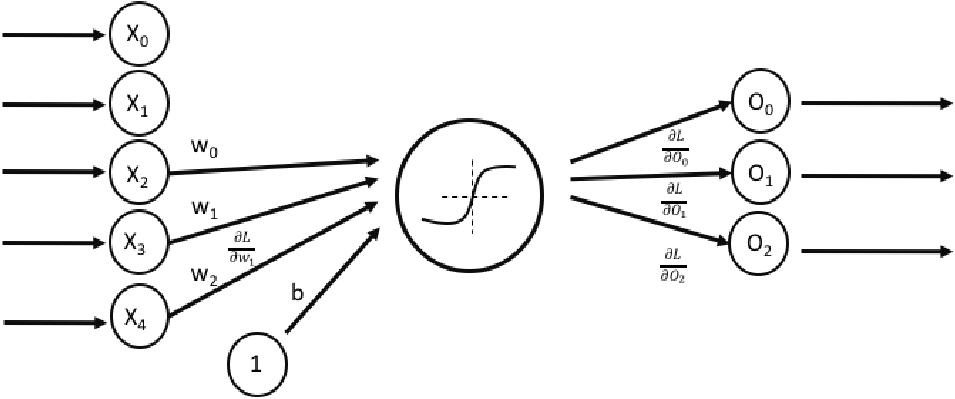
Convolutional Neuron Forward pass



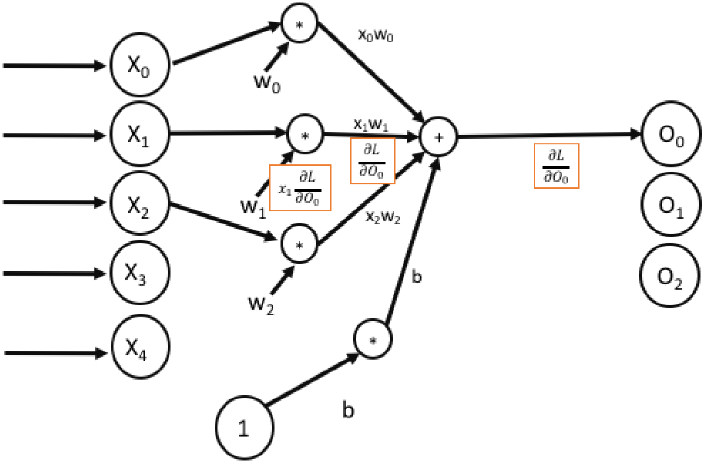
Convolutional Neuron Forward pass



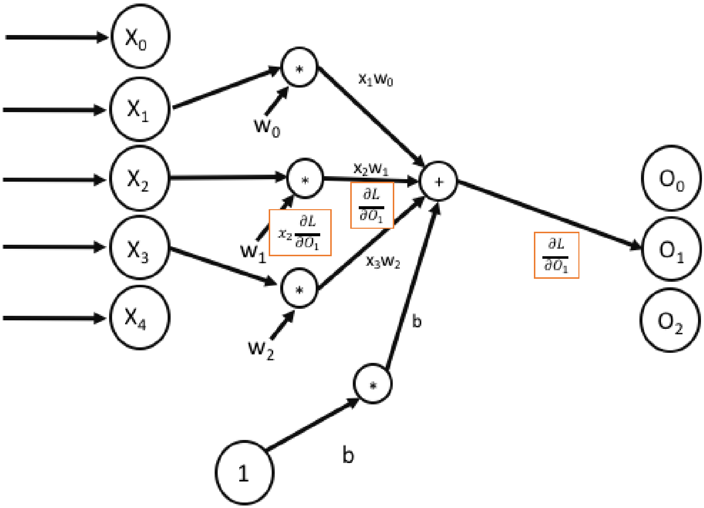
Convolutional Neuron Backward pass



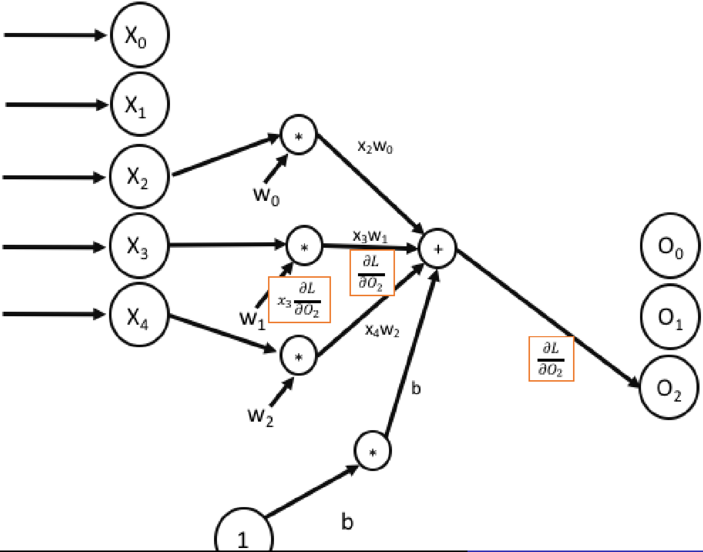
Backward pass on w_1 (Step 1)



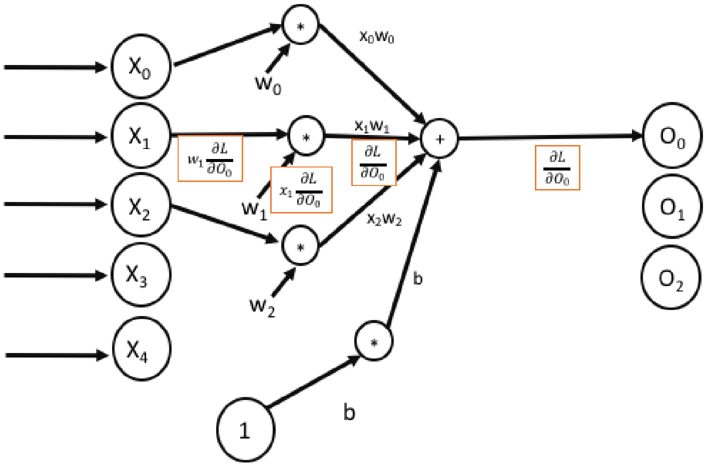
Backward pass on w_1 (Step 2)



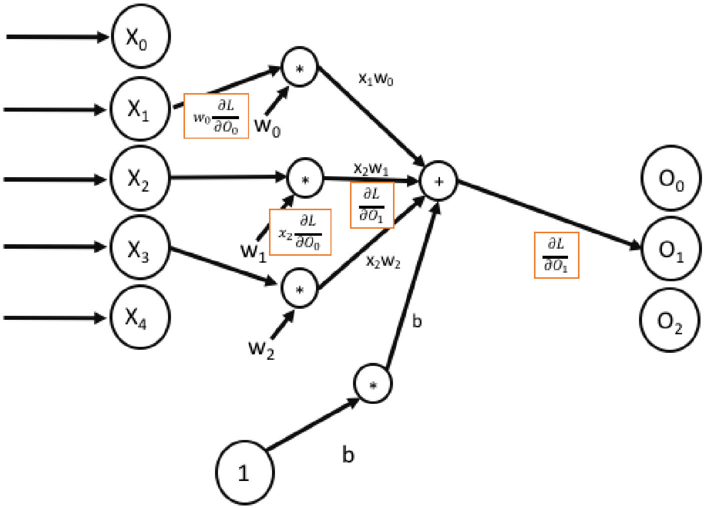
Backward pass on w_1 (Step 3)



Backward pass on x_1 (Step 1)

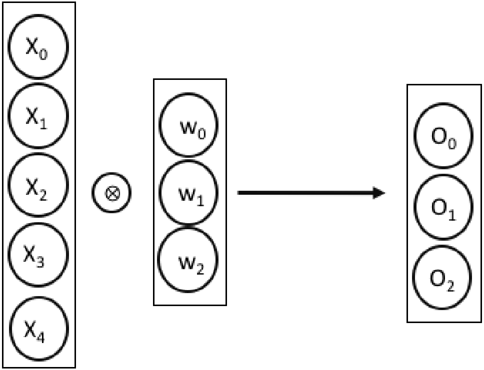


Backward pass on x_1 (Step 2)



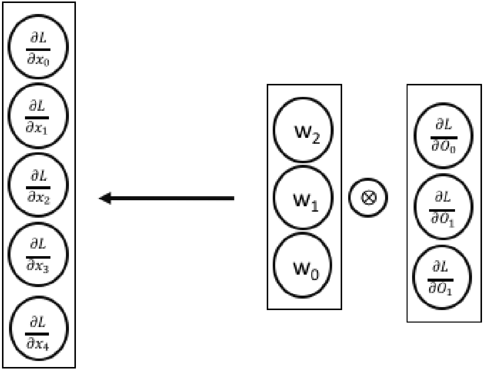
Propagating activation forward

Vector form



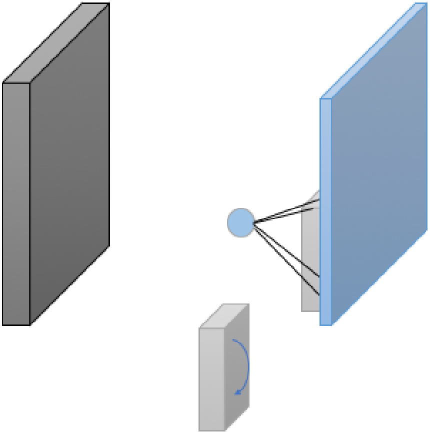
Propagating activation backward

Vector form



Propagating gradient backward 2D

Vector form 2D

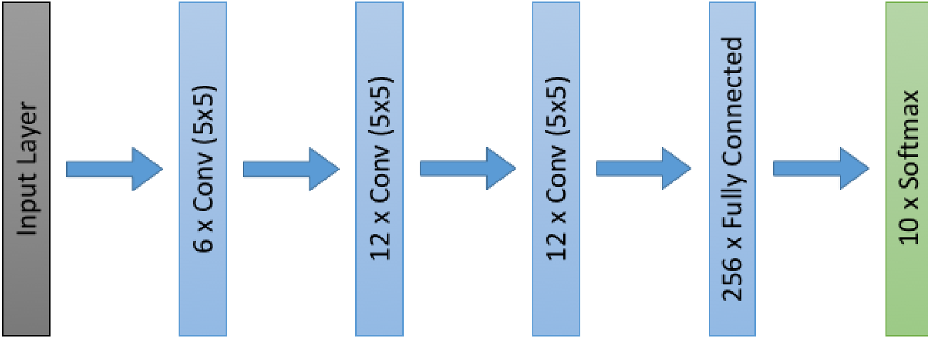


Propagating gradient backward 2D

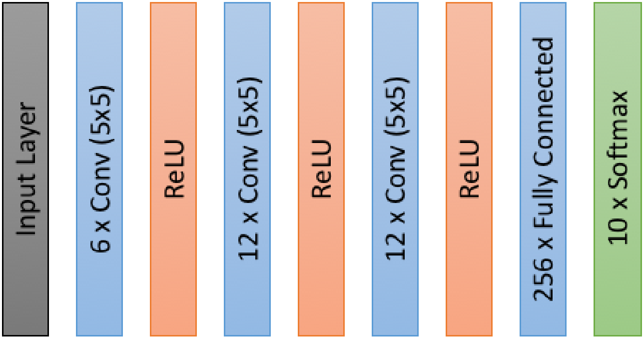
Backprop through a convolutional layer

- The grad of each parameter is: pre-activations times the grad of the loss wrt to post activations
- $\sum_{n}^{|x|-|w|+1} \frac{\partial L}{\partial y_n} x_{n+i-1}$
- The gradient flows in blocks back analogously as the activations flow forward
- To achive this the convolutional operation is done backwards. Or by transposing the filter
- $\delta_i = \sum_{i=1}^{|w|} \frac{\partial L}{\partial y_{n-i+1}} w_i$
- $\delta = \frac{\partial L}{\partial y} * flip(w)$

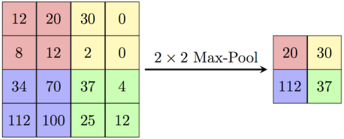
Architectural Depiction



Architectural Depiction



Subsampling - maxpooling



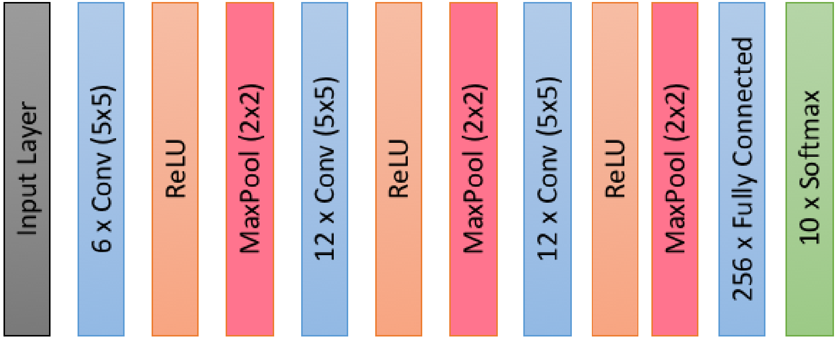
Backpropagation - maxpooling

$$a(x) = max(x), \frac{\partial a(x)}{\partial x_i} \begin{cases} 1, & if \;\; x_i = max(x) \\ 0, & otherwise \end{cases}$$

Backpropagation - maxpooling

$$a(x) = \frac{1}{m}\sum_{m}(x), \frac{\partial a(x)}{\partial x} = \frac{1}{m}$$

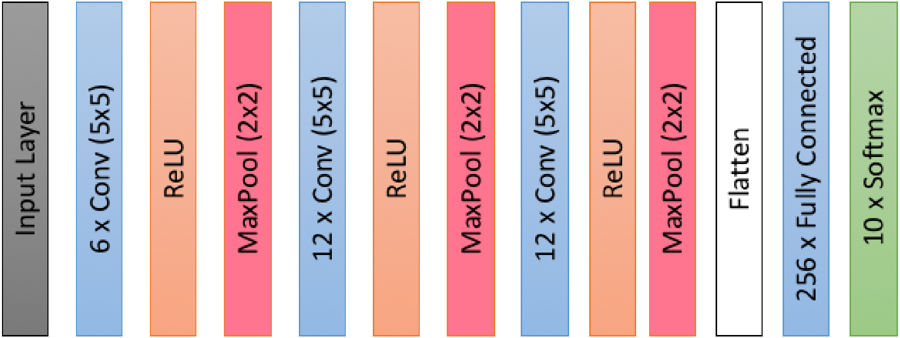
CNN - diagram


Input Layer
6 x Conv (5x5)
ReLU
MaxPool (2x2)
12 x Conv (5x5)
ReLU
MaxPool (2x2)
12 x Conv (5x5)
ReLU
MaxPool (2x2)
256 x Fully Connected
10 x Softmax

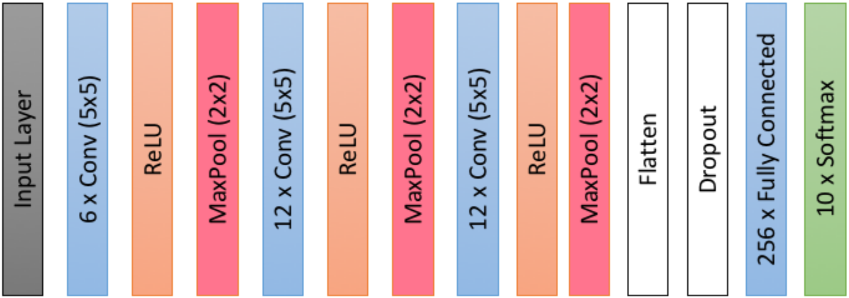
Typical CNN architecture

- Input
- Convolutions
- Flatten
- MLP
- Output

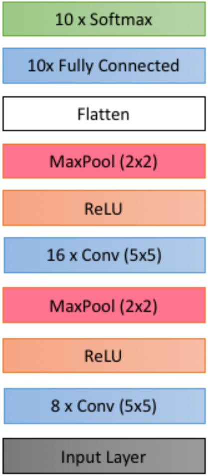
CNN - diagram


Input Layer
6 x Conv (5x5)
ReLU
MaxPool (2x2)
12 x Conv (5x5)
ReLU
MaxPool (2x2)
12 x Conv (5x5)
ReLU
MaxPool (2x2)
Flatten
256 x Fully Connected
10 x Softmax

CNN - diagram


Input Layer
6 x Conv (5x5)
ReLU
MaxPool (2x2)
12 x Conv (5x5)
ReLU
MaxPool (2x2)
12 x Conv (5x5)
ReLU
MaxPool (2x2)
Flatten
Dropout
256 x Fully Connected
10 x Softmax

CNN - diagram


10 x Softmax
10x Fully Connected
Flatten
MaxPool (2x2)
ReLU
16 x Conv (5x5)
MaxPool (2x2)
ReLU
8 x Conv (5x5)
Input Layer