Convolution Network

11-785 Introduction to Deep Learning

- lecture 10 & 11 -

TAVE Research DL001
Heeji Won

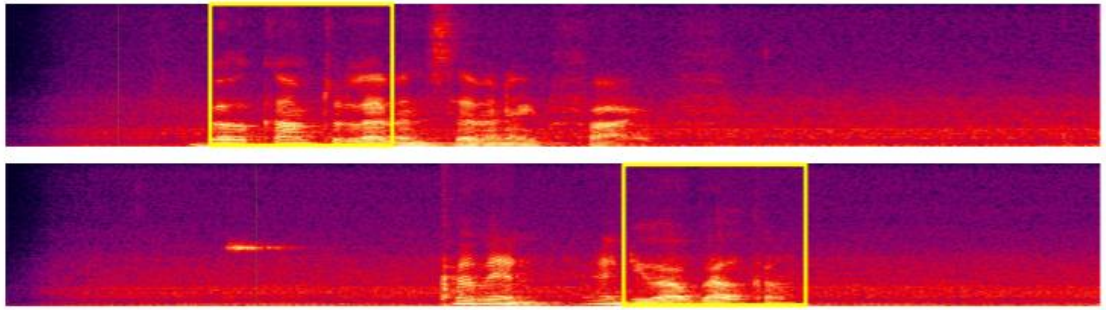
Contents

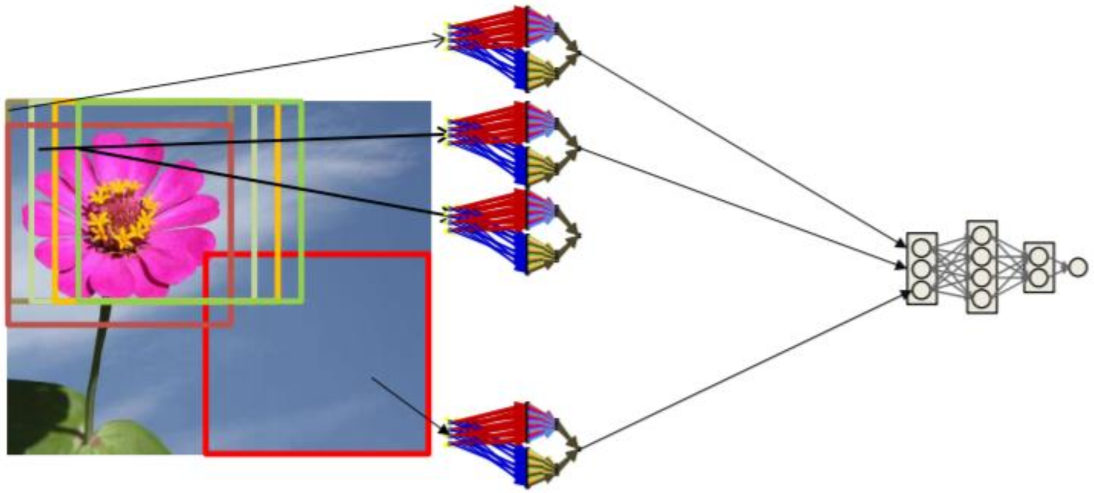
Contents

01. Benefits of CNN

- CNN

– The need for **shift invariance**



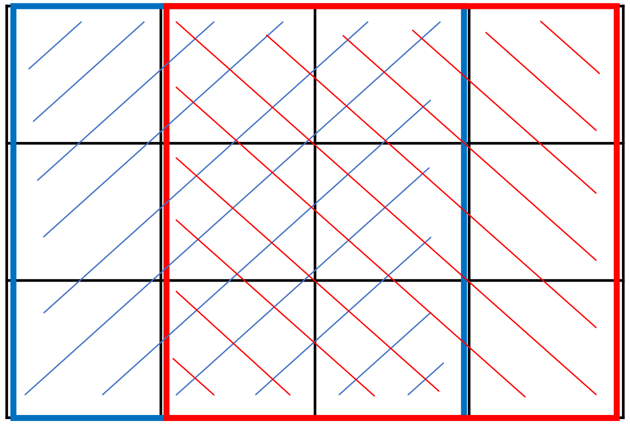


- Benefits of CNN

– The number of **params**

– More **generalization** (hierarchical features)

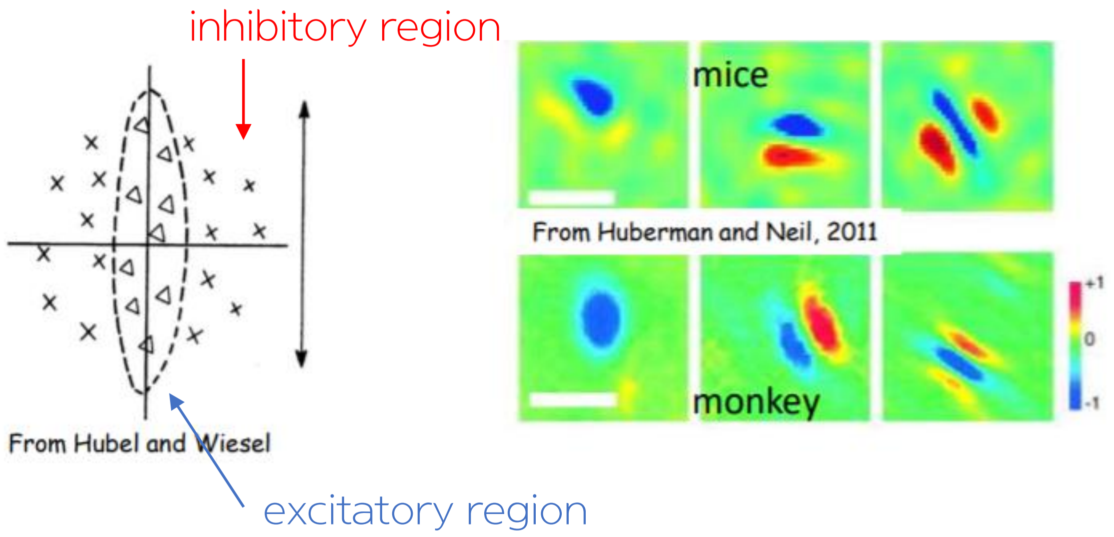
– Shared **computation**



Contents

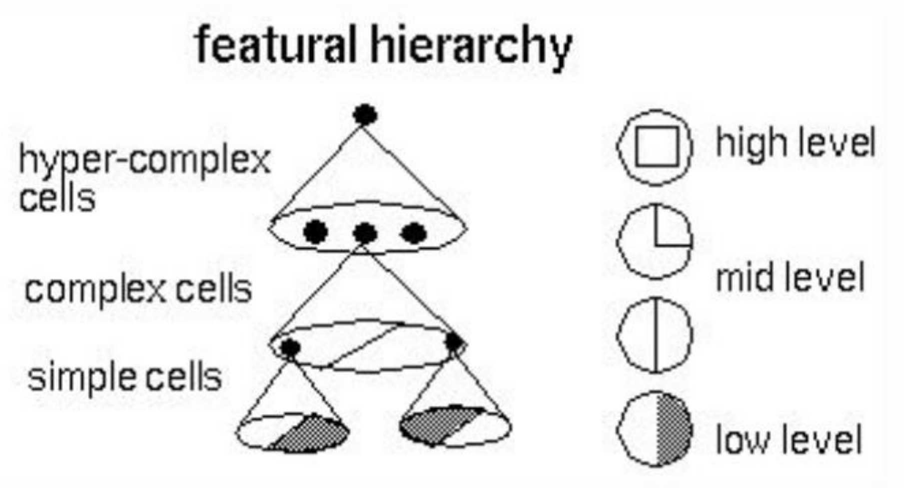
02. History of CNN

➢ Hubel and Wiesel, 1959



- Receptive fields consists of **excitatory and inhibitory regions.**
- Light must fall on excitatory region and NOT fall on inhibitory region.
- These fields could be oriented in a vertical, horizontal or oblique manner.

- Simple S-cells & Complex C-cells
  - Two levels of processing were identified; Simple S-cells and Complex C-cells
  - Both types responded to oriented slits of light, but complex cells were not 'confused' by spots of light while simple cells could be confused

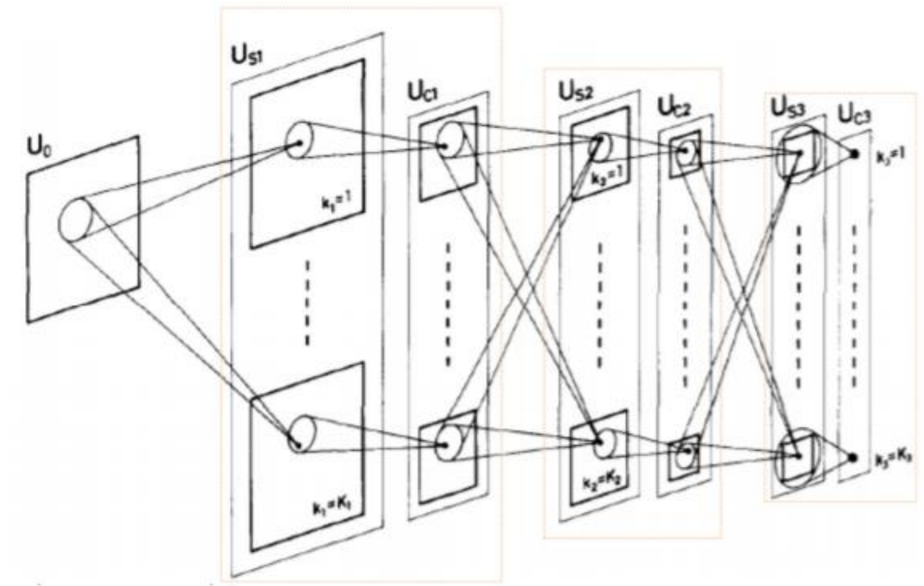


✓ However, this model cannot accommodate the color, spatial frequency and many other features

02. History of CNN

➢ NeoCognitron, 1980

- One of the chief problems is **Position Invariance of input.**
- grandmother cell fires even if grandmother moves to different location in field of vision



✓ Only S-cells are '**plastic**' (i.e. learnable), C-cells are fixed in their response

✓ C-cells' strides are bigger than 1, so C-planes are smaller than S-planes.

✓ The deeper the layer,
- the larger the receptive field of each neuron
- the smaller cell planes
- the bigger the number of planes

02. History of CNN

➢ NeoCognitron, 1980

- S-cells & C-cells

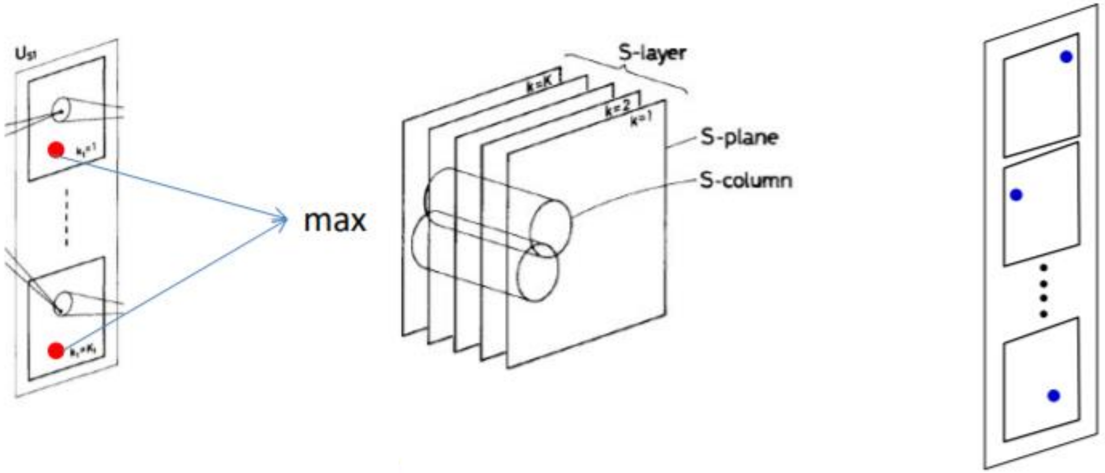
– S-cells (like ReLU function)

$$u_{Sl}(k_l, \mathbf{n}) = r_l \cdot \varphi\left[\frac{1 + \sum_{k_{l-1}=1}^{K_{l-1}} \sum_{\mathbf{v} \in S_l} a_l(k_{l-1}, \mathbf{v}, k_l) \cdot u_{Cl-1}(k_{l-1}, \mathbf{n}+\mathbf{v})}{1 + \frac{2r_l}{1+r_l} \cdot b_l(k_l) \cdot v_{Cl-1}(\mathbf{n})} - 1\right]$$

– C-cells : fires if weighted combination of S cells fires strongly enough (like Max function)

$$u_{Cl}(k_l, \mathbf{n}) = \psi\left[\frac{1 + \sum_{\mathbf{v} \in D_l} d_l(\mathbf{v}) \cdot u_{Sl}(k_l, \mathbf{n}+\mathbf{v})}{1 + v_{Sl}(\mathbf{n})} - 1\right]$$

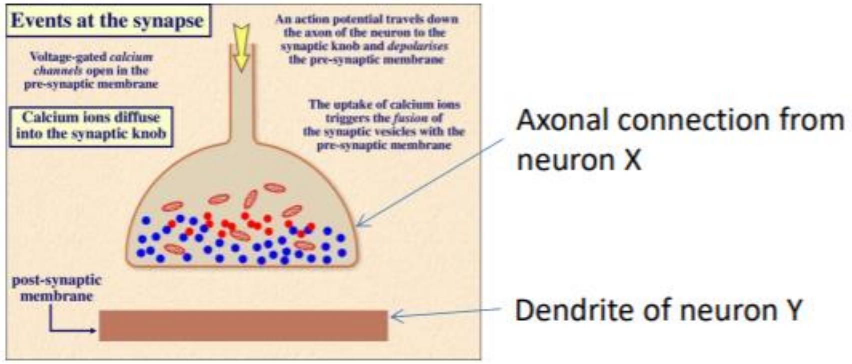
$$\psi[x] = \varphi[x/(\alpha + x)]$$



✓ Unsupervised Learning

✓ Randomly initialize S cells, perform Hebbian learning updates in response to input

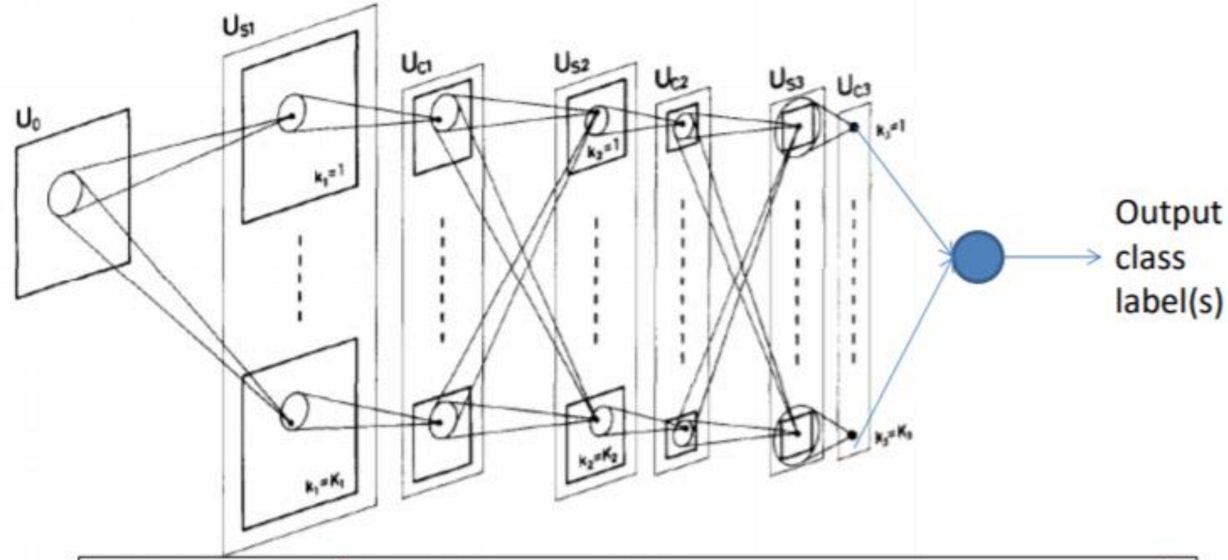
cf) Hebbian learning $w_{xy} = w_{xy} + \eta xy$



02. History of CNN

➢ LeNet

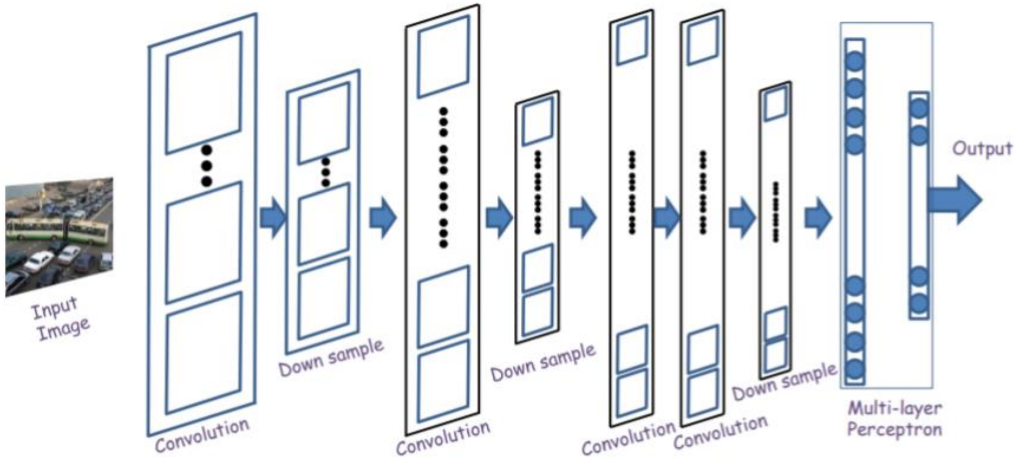
- supervising the neocognition



$$U_{S,l,n}(i,j) = \sigma\left(\sum_{p}\sum_{k=1}^{K_l}\sum_{l=1}^{K_l} w_{S,l,n}(p,k,l)\,U_{C,l-1,p}(i+l-1,\, j+k-1)\right)$$

$$U_{C,l,n}(i,j) = \max_{k\in(i,i+L_l),\, j\in(l,l+L_l)} \left(U_{S,l,n}(i,j)\right)$$

- The general architecture of a convolution neural network



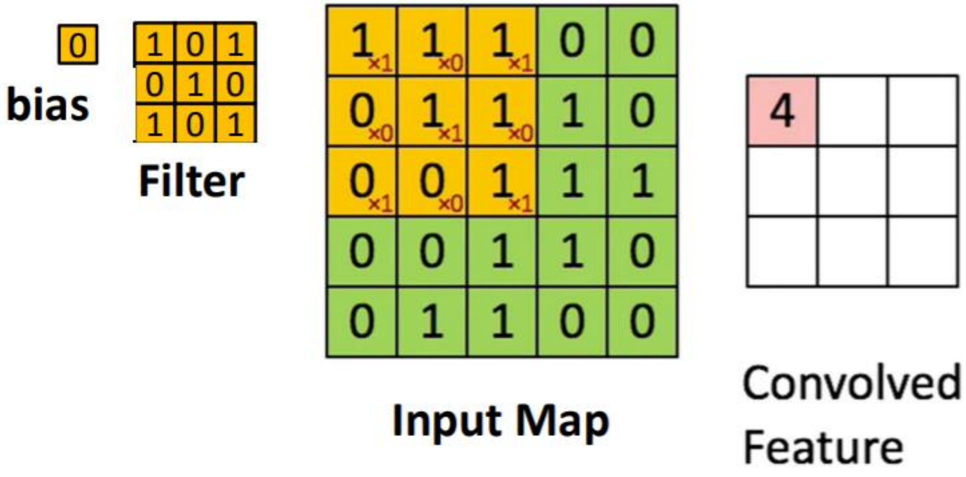
– A convolutional neural network comprises 'convolutional' layers (correspond to S planes) and 'down-sampling' layers (correspond to C planes)

Contents

03. Definition of Convolution

- Convolution

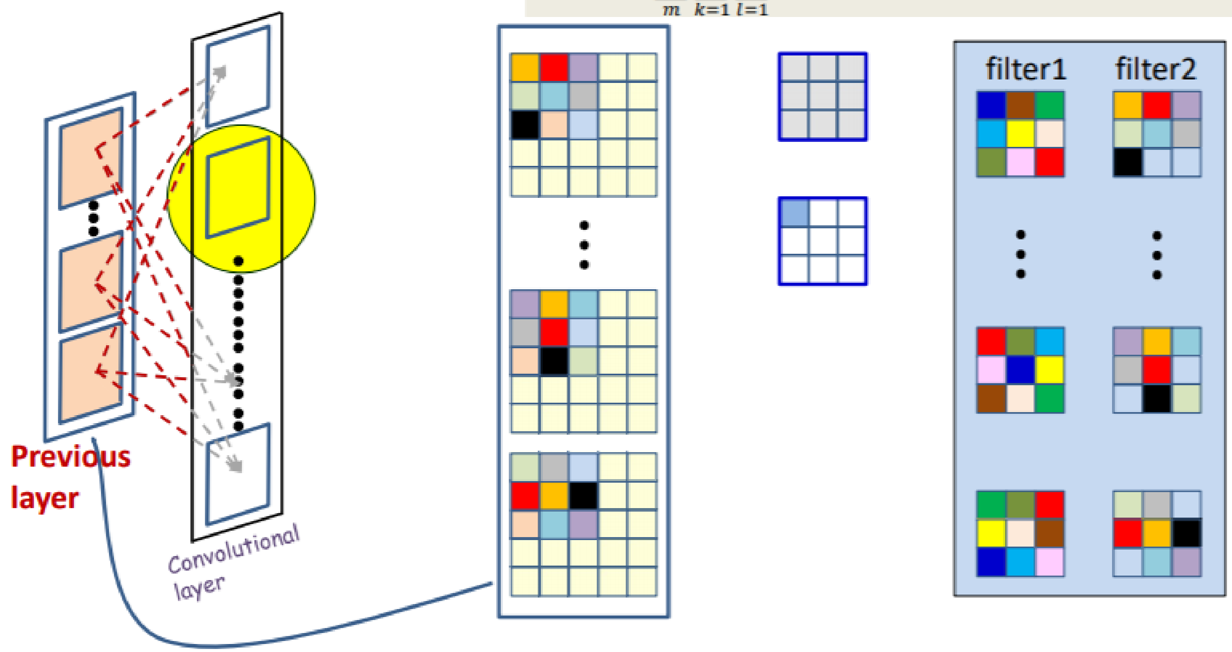
– scanning an image with a 'filter'



✓ The 'stride' between adjacent scanned locations need not be 1

- What really happen

$$z(2,i,j) = \sum_{m} \sum_{k=1}^{3} \sum_{l=1}^{3} w(2,m,k,l) I(m, i+l-1, j+k-1) + b(2)$$

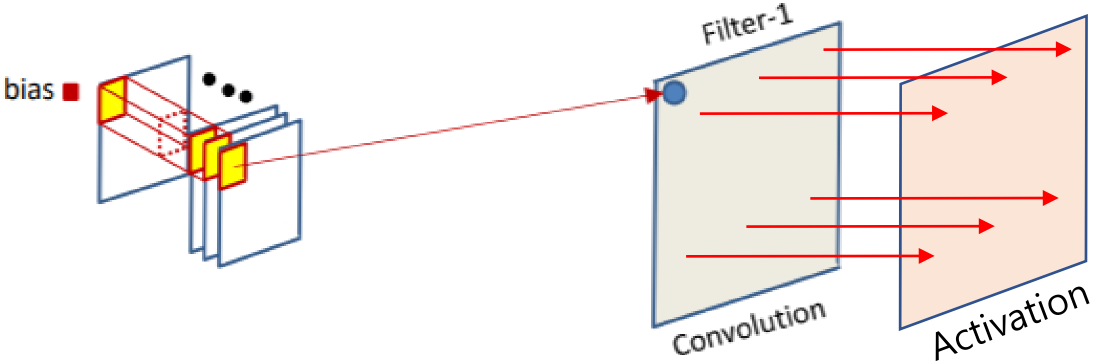


✓ There are as many weights as size of the filter × no. of maps in previous layer × no. of output maps

03. Definition of Convolution

- A different view

– the 'cube' view of input maps



$$z(s,i,j) = \sum_{p} \overbrace{\sum_{k=1}^{L} \sum_{l=1}^{L} w(s,p,k,l) Y(p, i+l-1, j+k-1) + b(s)}^{\text{One map}}$$

- Output size (each side) = $\lfloor (N-M)/S \rfloor + 1$

– Image size : N × N
– Filter : M × M
– Stride : S

- Zero padding

– Zero-pad the input

0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0

– To the result of the convolution is the same size as the original image, $P_L + P_R = M - 1$

✓ The number of filters is **a power of 2**

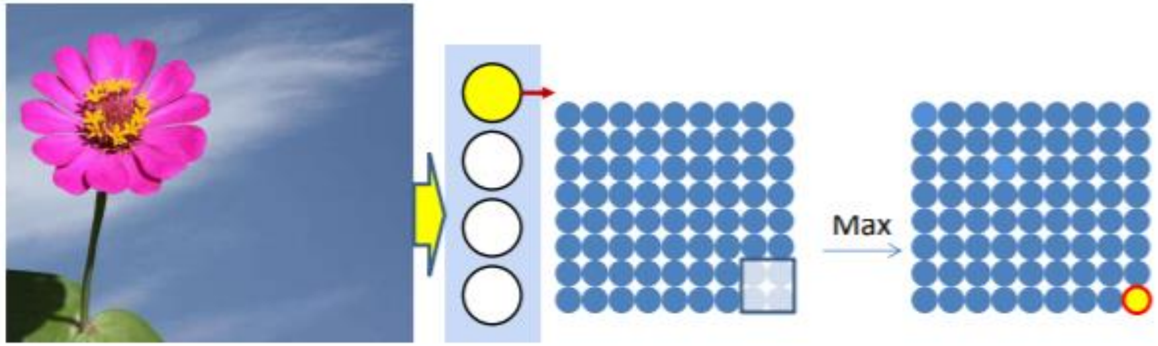
✓ Filters are typically **5×5, 3×3, or ever 1×1**

– operates over the depth of the stack of maps, but has no spatial extent

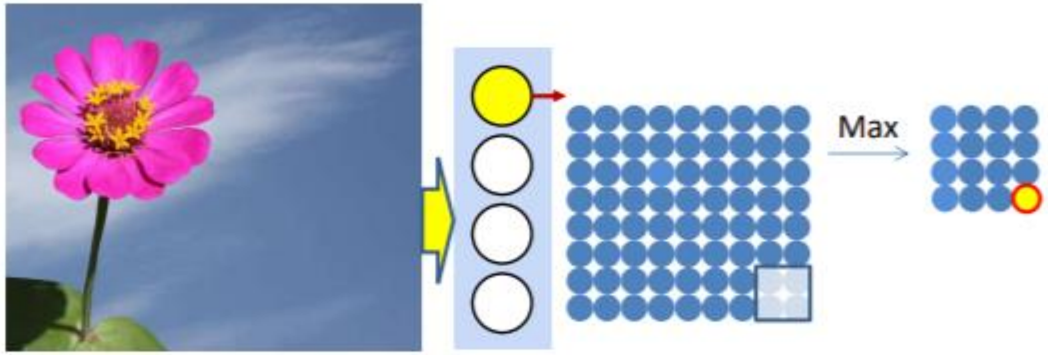
Contents

04. Pooling

- Max pooling



- Down-sampling requires Strides > 1



- Alternative to Max pooling
  - Mean pooling
  - p-norm

$$y = \sqrt[p]{\frac{1}{K^2}\sum_{i,j} x_{ij}^p}$$

  - Network in network

- Formula of Max pooling

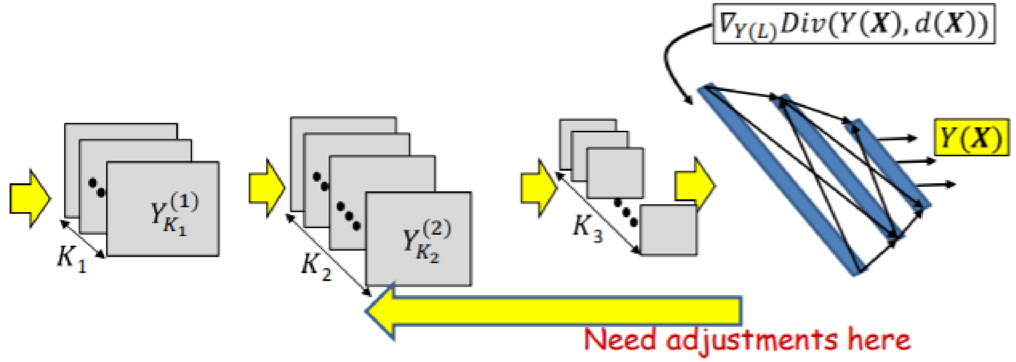
$$P_m^{(2)}(i,j) = \underset{\substack{k \in Xwin(i), \\ l \in Ywin(j)}}{\operatorname{argmax}} \; Y_m^{(1)}(k,l) \quad \Rightarrow \quad \text{find index!}$$

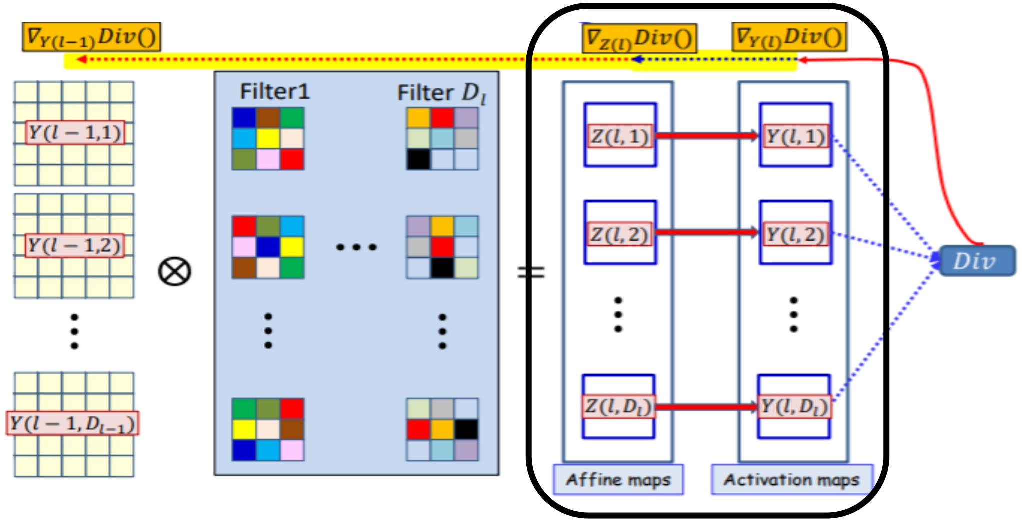
$$Y_m^{(2)}(i,j) = Y_m^{(1)}(P_m^{(2)}(i,j))$$

Contents

05. Learning the network

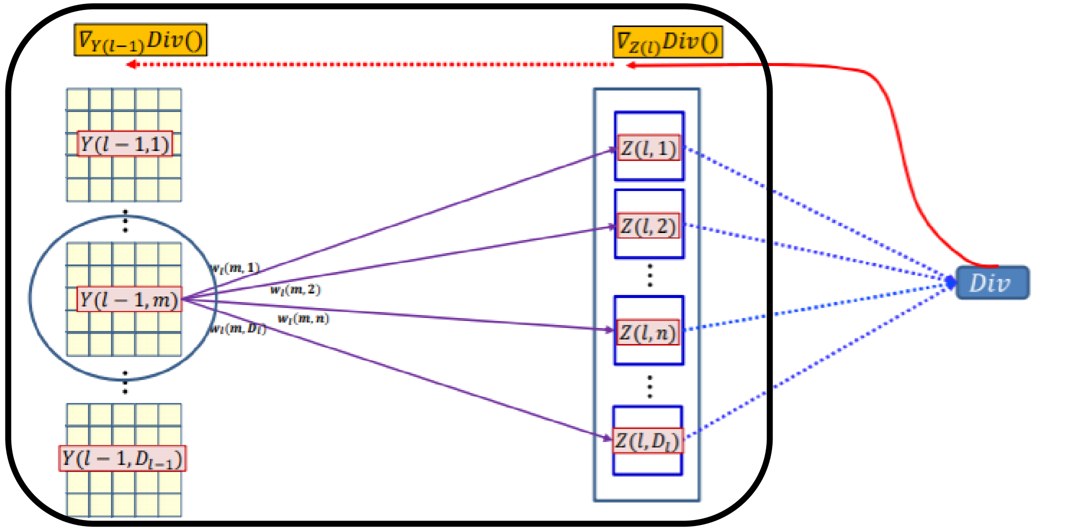
- Backpropagation





$$y(l,m,x,y) = f(z(l,m,x,y))$$

$$\frac{dDiv}{dz(l,m,x,y)} = \frac{dDiv}{d\,y(l,m,x,y)} f'(z(l,m,x,y))$$



$$\nabla_{Y(l-1,m)} Div(.) = \sum_n \nabla_{Z(l,n)} Div(.)\, \underbrace{\nabla_{Y(l-1,m)} Z(l,n)}_{\text{Need to compute}}$$

- **For convolutional layers:**
  - Given derivative w.r.t. activation $Y(l)$ compute the derivatives w.r.t. the affine combination $Z(l)$ maps
  - From derivative w.r.t. $Z(l)$ compute the derivative w.r.t. $Y(l-1)$ and $w(l)$
- **For pooling layers:**
  - How to compute the derivative w.r.t. $Y(l-1)$ given derivatives w.r.t. $Y(l)$

Thank you