COS324:
Introduction To Machine Learning

Prof. Yoram Singer

PRINCETON UNIVERSITY

Topic: Convolutional Neural Networks

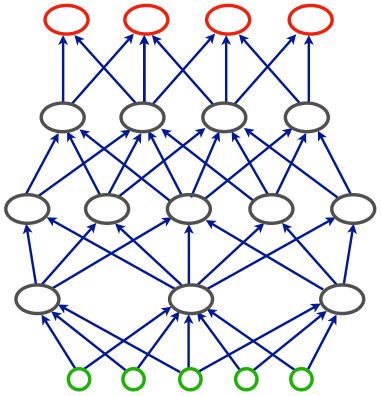
(Almost) Fully Connected Net

$\mathbf{h}^4 = \sigma(A^4 \mathbf{h}^3)$

$\mathbf{h}^3 = \sigma(A^3 \mathbf{h}^2)$

$\mathbf{h}^2 = \sigma(A^2 \mathbf{h}^1)$

$\mathbf{h}^1 = \sigma(A^1 \mathbf{h}^0)$



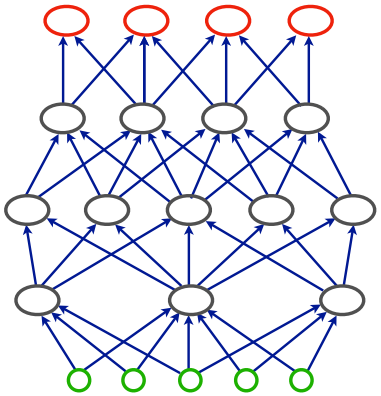
(Almost) Fully Connected Net

$\mathbf{h}^4 = \sigma(A^4 \mathbf{h}^3)$

$\mathbf{h}^3 = \sigma(A^3 \mathbf{h}^2)$

$\mathbf{h}^2 = \sigma(A^2 \mathbf{h}^1)$

$\mathbf{h}^1 = \sigma(A^1 \mathbf{h}^0)$



Number of neurons in layer i: $|\mathbf{h}^i| = d_i$

A^i is $d_i \times d_{i-1}$ matrix

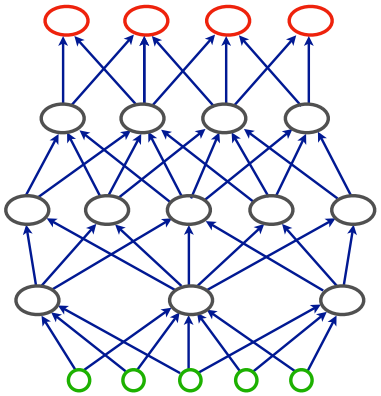
(Almost) Fully Connected Net

$\mathbf{h}^4 = \sigma(A^4 \mathbf{h}^3)$

$\mathbf{h}^3 = \sigma(A^3 \mathbf{h}^2)$

$\mathbf{h}^2 = \sigma(A^2 \mathbf{h}^1)$

$\mathbf{h}^1 = \sigma(A^1 \mathbf{h}^0)$



Number of neurons in layer i: $|\mathbf{h}^i| = d_i$

A^i is $d_i \times d_{i-1}$ matrix

Need to constrain each A^i to save space & runtime

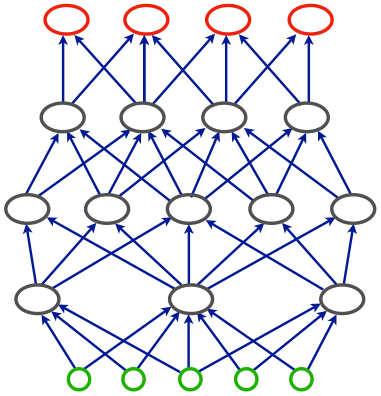
(Almost) Fully Connected Net

$$\mathbf{h}^4 = \sigma\left(A^4 \mathbf{h}^3\right)$$

$$\mathbf{h}^3 = \sigma\left(A^3 \mathbf{h}^2\right)$$

$$\mathbf{h}^2 = \sigma\left(A^2 \mathbf{h}^1\right)$$

$$\mathbf{h}^1 = \sigma\left(A^1 \mathbf{h}^0\right)$$

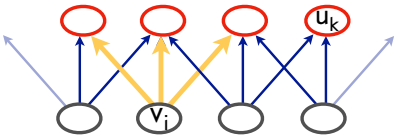


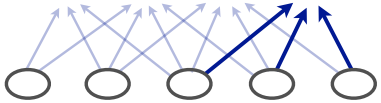
Number of neurons in layer i: $|\mathbf{h}^i| = d_i$

A^i is $d_i \times d_{i-1}$ matrix

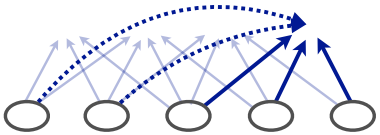
Need to constrain each A^i to save space & runtime

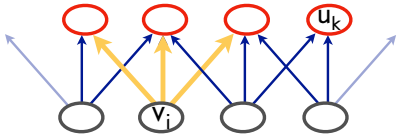
Sparsity Constraints



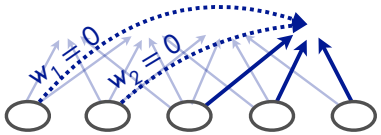


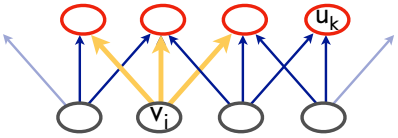
Sparsity Constraints





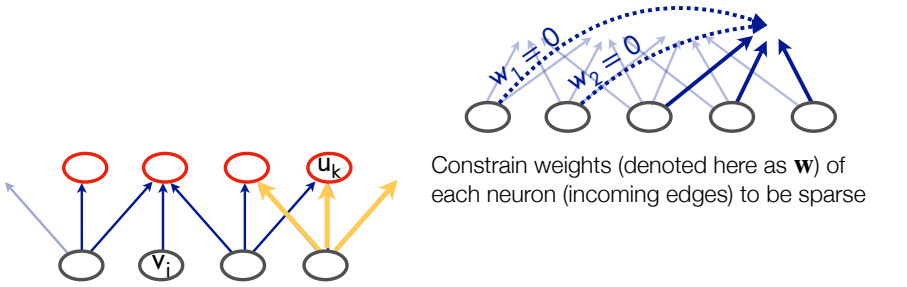
Sparsity Constraints





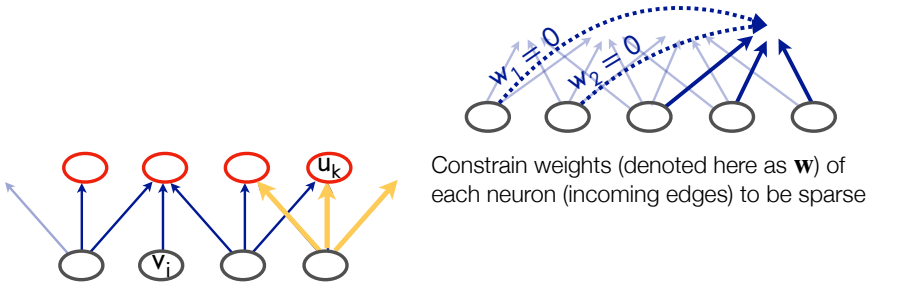
Constrain weights (denoted here as $\mathbf{w}$) of each neuron (incoming edges) to be sparse

Sparsity Constraints



Constrain weights (denoted here as $\mathbf{w}$) of each neuron (incoming edges) to be sparse

Sparsity Constraints



Constrain weights (denoted here as $\mathbf{w}$) of each neuron (incoming edges) to be sparse

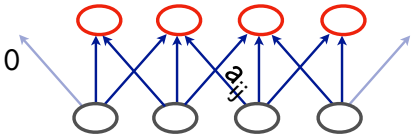
Add self-similarity constraints

$\mathbf{w}[v_i \rightarrow u_k] = 0 \;\Rightarrow\; \mathbf{w}[v_{i+1} \rightarrow u_{k+1}] = 0$

Structural Self-Similarity

$a_{ik} = \mathbf{w}[v_i \rightarrow u_k] = 0$

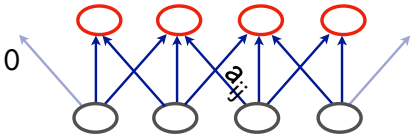
$\Rightarrow\; a_{i+1\,k+1} = \mathbf{w}[v_{i+1} \rightarrow u_{k+1}] = 0$



Structural Self-Similarity

$a_{ik} = \mathbf{w}[v_i \rightarrow u_k] = 0$

$\Rightarrow\; a_{i+1\,k+1} = \mathbf{w}[v_{i+1} \rightarrow u_{k+1}] = 0$

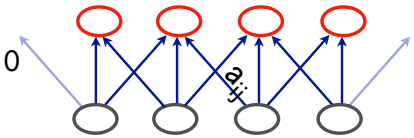


$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & \dots & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 & 0 & \dots & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} & 0 & \dots & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & a_{46} & \dots & 0 \\ & \dots & \dots & \dots & \dots & \dots & & \end{bmatrix}$$

Structural Self-Similarity

$a_{ik} = \mathbf{w}[v_i \rightarrow u_k] = 0$

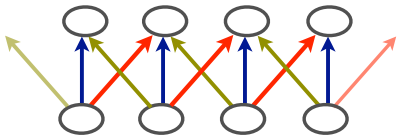
$\Rightarrow \; a_{i+1\,k+1} = \mathbf{w}[v_{i+1} \rightarrow u_{k+1}] = 0$



$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & \dots & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 & 0 & \dots & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} & 0 & \dots & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & a_{46} & \dots & 0 \\ & \dots & \dots & \dots & \dots & \dots & & \end{bmatrix}$$

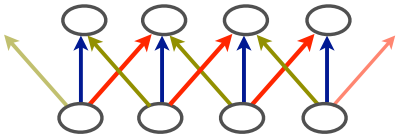
Can we further save ?

Parameter Tying



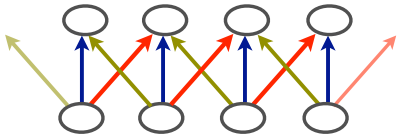
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & \dots & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 & 0 & \dots & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} & 0 & \dots & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & a_{46} & \dots & 0 \\ & \dots & \dots & \dots & \dots & \dots & & \end{bmatrix}$$

Parameter Tying



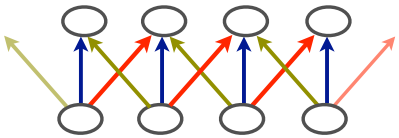
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & \dots & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 & 0 & \dots & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} & 0 & \dots & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & a_{46} & \dots & 0 \\ & \dots & \dots & \dots & \dots & \dots & & \end{bmatrix}$$

Parameter Tying



$$\begin{bmatrix} c_1 & c_2 & c_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_1 & c_2 & c_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 & c_2 & c_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_1 & c_2 & c_3 & 0 & 0 \\ & \dots & \dots & \dots & \dots & \dots & & \end{bmatrix}$$

Parameter Tying



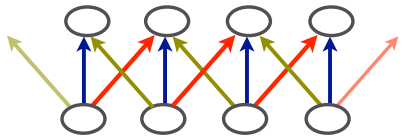
$$\begin{bmatrix} c_1 & c_2 & c_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_1 & c_2 & c_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 & c_2 & c_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_1 & c_2 & c_3 & 0 & 0 \\ & \cdots & \cdots & \cdots & \cdots & \cdots & & \end{bmatrix}$$

$$h_i^{r+1} = \sigma\Big(\sum_{j=-s}^{s} c_j h_{i-j}^r \Big)$$

$$j < 0 : \; c_j \equiv c_{2s+1+j}$$

Parameter Tying

$$a_{ij} = \begin{cases} c_{j-j+1} & 0 \le j - i < s \\ 0 & o.w \end{cases}$$

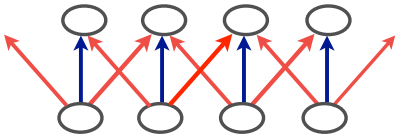


$$\begin{bmatrix} c_1 & c_2 & c_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_1 & c_2 & c_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 & c_2 & c_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_1 & c_2 & c_3 & 0 & 0 \\ & \cdots & \cdots & \cdots & \cdots & \cdots & & \end{bmatrix}$$

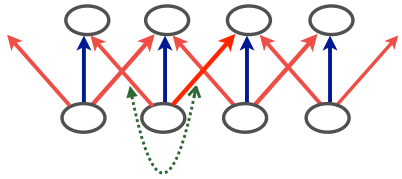
$$h_i^{r+1} = \sigma\Big(\sum_{j=-s}^{s} c_j h_{i-j}^r \Big)$$

$$j < 0 : \; c_j \equiv c_{2s+1+j}$$

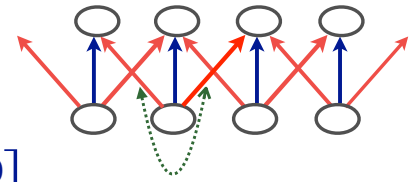
Parameter Symmetrization



Parameter Symmetrization



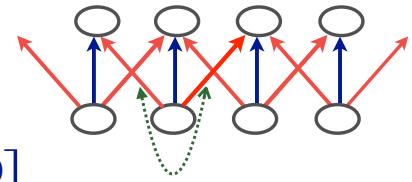
Parameter Symmetrization



$$\begin{bmatrix} c_1 & c_2 & c_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_1 & c_2 & c_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 & c_2 & c_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_1 & c_2 & c_1 & 0 & 0 \\ & \cdots & \cdots & \cdots & \cdots & \cdots & & \end{bmatrix}$$

Parameter Symmetrization

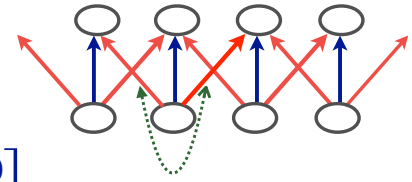
$$a_{ij} = \begin{cases} c_{|i-j|} & |j-i| < \frac{s}{2} \\ 0 & o.w \end{cases}$$



$$\begin{bmatrix} c_1 & c_2 & c_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_1 & c_2 & c_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 & c_2 & c_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_1 & c_2 & c_1 & 0 & 0 \\ & \cdots & \cdots & \cdots & \cdots & \cdots & & \end{bmatrix}$$

Parameter Symmetrization

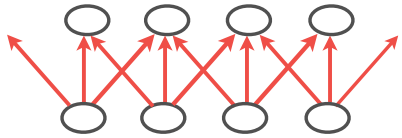
$$a_{ij} = \begin{cases} c_{|i-j|} & |j-i| < \frac{s}{2} \\ 0 & o.w \end{cases}$$



$$\begin{bmatrix} c_1 & c_2 & c_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_1 & c_2 & c_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 & c_2 & c_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_1 & c_2 & c_1 & 0 & 0 \\ & \cdots & \cdots & \cdots & \cdots & \cdots & & \end{bmatrix}$$

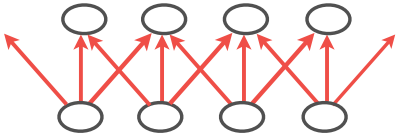
$$h_i^{r+1} = \sigma\left(\sum_{j=0}^{s} c_j \frac{h_{i-j}^r + h_{i+j}^r}{2}\right)$$

Averaging Neuron



Averaging Neuron

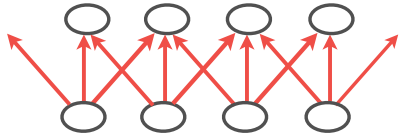
$$a_{ij} = \begin{cases} \frac{1}{2s+1} & |j-i| < \frac{s}{2} \\ 0 & o.w \end{cases}$$



$$\begin{bmatrix} c & c & c & 0 & 0 & 0 & 0 & 0 \\ 0 & c & c & c & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c & c & c & 0 & 0 \\ 0 & 0 & 0 & 0 & c & c & c & 0 \\ 0 & 0 & 0 & 0 & 0 & c & c & c \end{bmatrix}$$

Averaging Neuron

$$a_{ij} = \begin{cases} \frac{1}{2s+1} & |j-i| < \frac{s}{2} \\ 0 & o.w \end{cases}$$

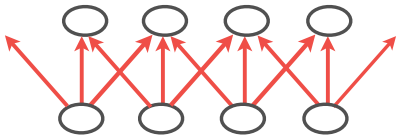


$$\begin{bmatrix} c & c & c & 0 & 0 & 0 & 0 & 0 \\ 0 & c & c & c & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c & c & c & 0 & 0 \\ 0 & 0 & 0 & 0 & c & c & c & 0 \\ 0 & 0 & 0 & 0 & 0 & c & c & c \end{bmatrix}$$

$$h_i^{r+1} = \sigma\left(\frac{1}{2s+1}\sum_{j=-s}^{s} h_{i-j}^{r}\right)$$

© 2020 YORAM SINGER 8

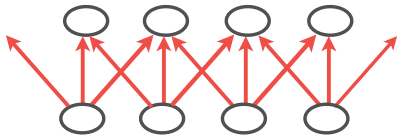
Max-Pooling (Neuron)



© 2020 YORAM SINGER 9

Max-Pooling (Neuron)

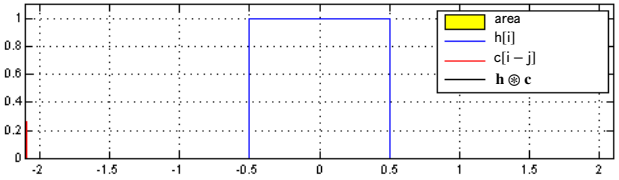
$$h_i^{r+1} = \sigma\left(c \max_{j=-s}^{s} h_{i-j}^{r}\right)$$



© 2020 YORAM SINGER 9

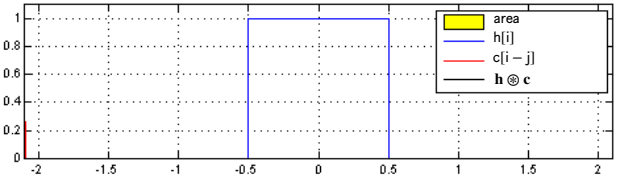
Convolution

$$h_i^{r+1} = \sigma\Big(\underbrace{\sum_{j=-s}^{s} c_j h_{i-j}^r}_{z_i} \Big) \qquad (\mathbf{c} \circledast \mathbf{h})_i = \sum_{j=-s}^{s} c_j h_{i-j}$$



Convolution

$$h_i^{r+1} = \sigma\Big(\underbrace{\sum_{j=-s}^{s} c_j h_{i-j}^r}_{z_i} \Big) \qquad (\mathbf{c} \circledast \mathbf{h})_i = \sum_{j=-s}^{s} c_j h_{i-j}$$



© 2020 YORAM SINGER 10

Boundary Conditions

For $i \le s$ and $i > r - s$:

Convolution is not uniquely defined

Boundary Conditions

For $i \le s$ and $i > r - s$:

Convolution is not uniquely defined

h

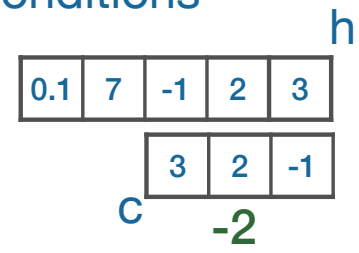
0.1	7	-1	2	3

© 2020 YORAM SINGER 11

Boundary Conditions

For $i \leq s$ and $i > r - s$:

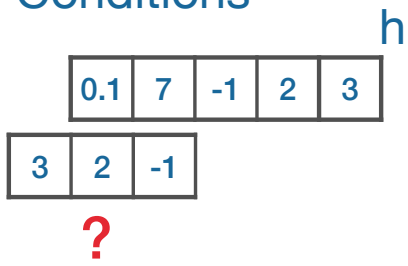
Convolution is not uniquely defined



Boundary Conditions

For $i \leq s$ and $i > r - s$:

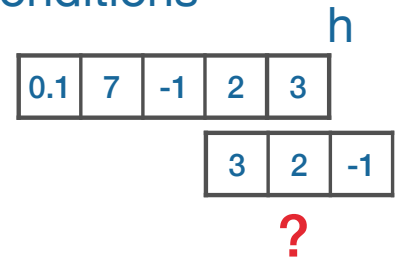
Convolution is not uniquely defined



Boundary Conditions

For $i \leq s$ and $i > r - s$:

Convolution is not uniquely defined



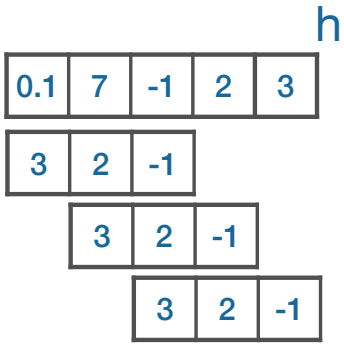
Valid Only Convolution

For $i \leq s$ and $i > r - s$:

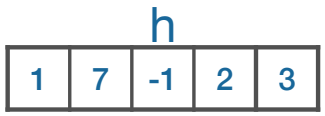
Convolution is not uniquely defined

Use only indices $s < i \leq r - s$:

Size of next layer reduced by $2s$



Padded Convolution



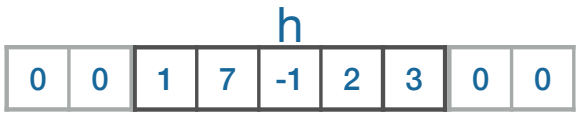
Define:

$h_i = h_0$ for $i \leq 0$

$h_i = h_{r+1}$ for $i > r$

Output defined for $1 \leq i \leq r$

Padded Convolution



Define:

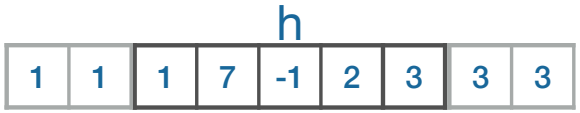
$h_i = h_0$ for $i \leq 0$

$h_i = h_{r+1}$ for $i > r$

Zero padding: $h_{r+1} = h_0 = 0$

Output defined for $1 \leq i \leq r$

Padded Convolution



Define:

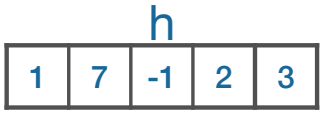
$h_i = h_0$ for $i \leq 0$

$h_i = h_{r+1}$ for $i > r$

Extension padding: $h_0 = h_1$ and $h_{r+1} = h_r$

Output defined for $1 \leq i \leq r$

Padded Convolution



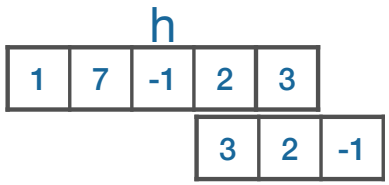
Define:

$h_i = h_0$ for $i \leq 0$

$h_i = h_{r+1}$ for $i > r$

Output defined for $1 \leq i \leq r$

Padded Convolution

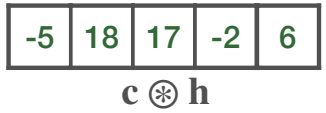


Define:

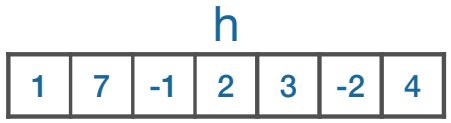
$h_i = h_0$ for $i \le 0$

$h_i = h_{r+1}$ for $i > r$

Output defined for $1 \le i \le r$



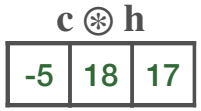
Striding

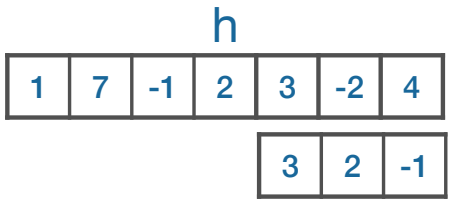


Reduce size of next layer through sub-sampling

$$h_i^{r+1} = \sigma\left(\sum_{j=-s}^{s} c_j \, h^r_{\mu(i)-j} \right) \text{ where } \mu(i) = b(i-1)+1$$

Striding

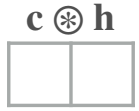


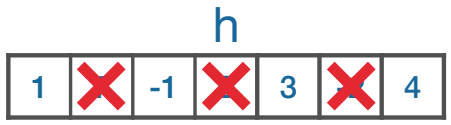


Reduce size of next layer through sub-sampling

$$h_i^{r+1} = \sigma\left(\sum_{j=-s}^{s} c_j \, h^r_{\mu(i)-j} \right) \text{ where } \mu(i) = b(i-1)+1$$

Sub-Sampling (Dilation)

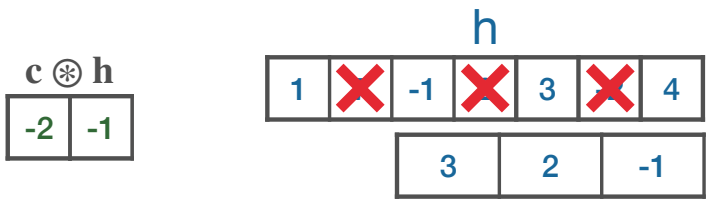




Reduce size of next layer through sub-sampling

$$h_i^{r+1} = \sigma\left(\sum_{j=-s}^{s} c_j \, h^r_{i-\mu(j)} \right) \text{ where } \mu(i) = b\,j$$

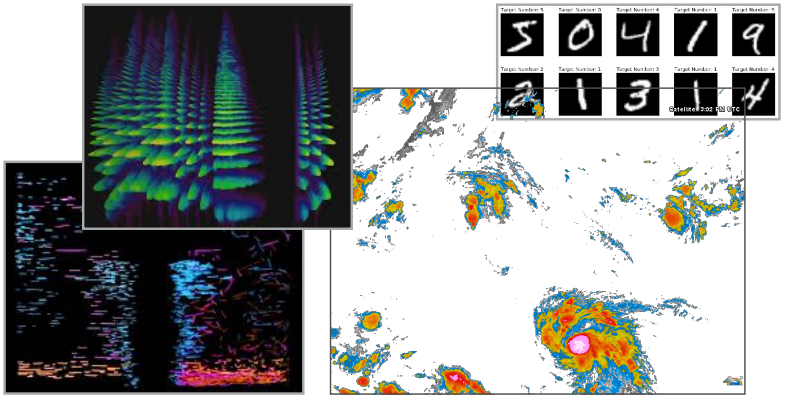
Sub-Sampling (Dilation)



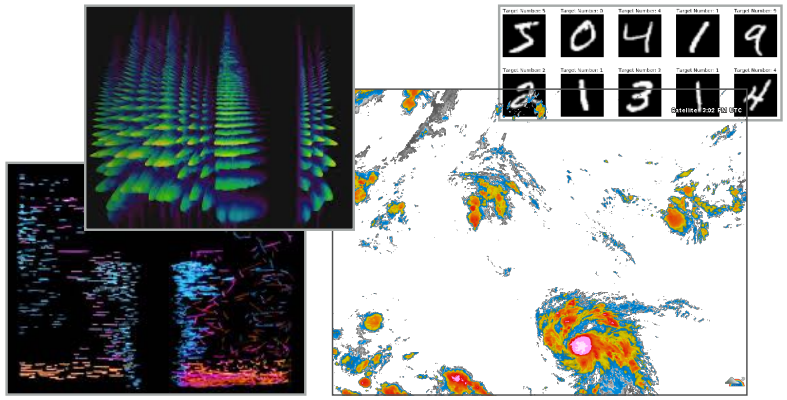
Reduce size of next layer through sub-sampling

$$h_i^{r+1} = \sigma\left(\sum_{j=-s}^{s} c_j \, h^r_{i-\mu(j)} \right) \text{ where } \mu(i) = b\,j$$

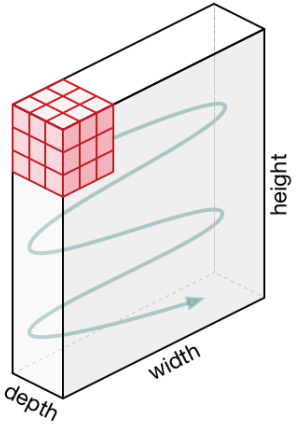
Multidimensional Convolutions

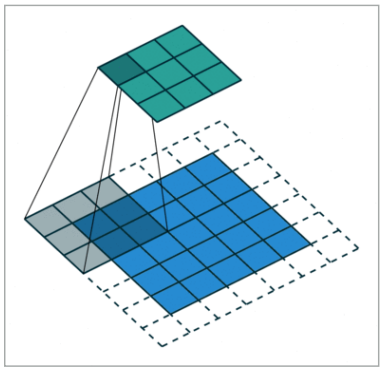


Multidimensional Convolutions

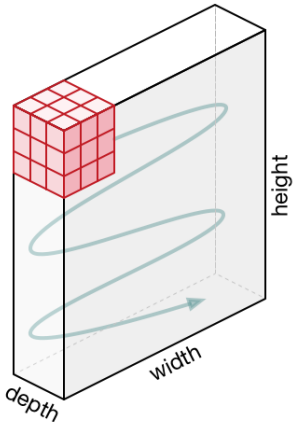


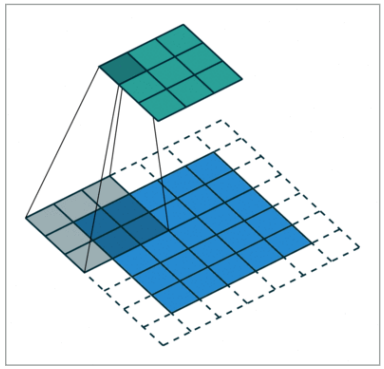
2D & 3D Convulutions

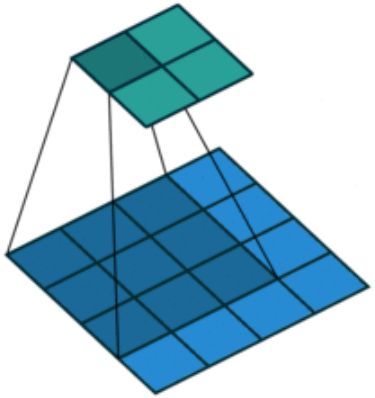




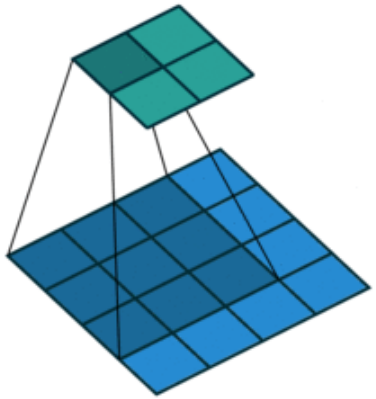
2D & 3D Convulutions



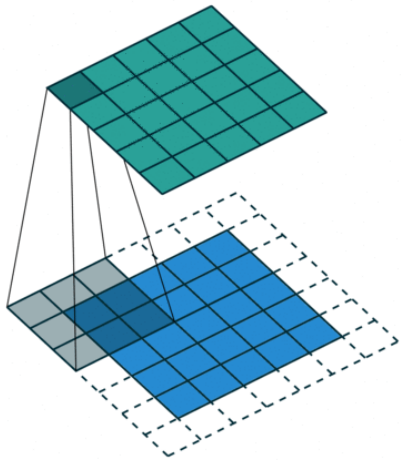




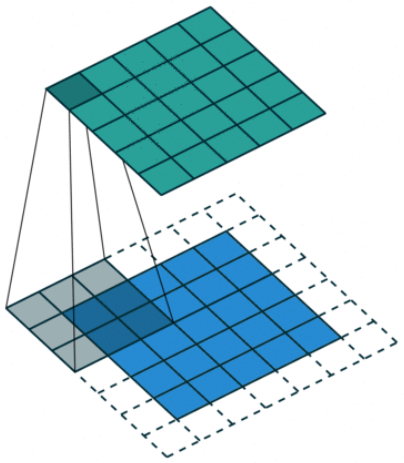
No Padding
No Striding
$\mathbf{c} \in \mathbf{R}^{3\times3}$



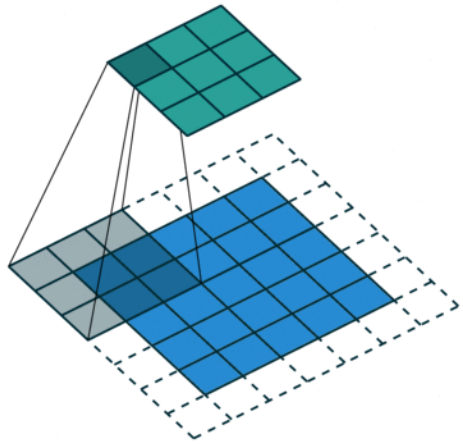
No Padding
No Striding
$\mathbf{c} \in \mathbf{R}^{3\times3}$



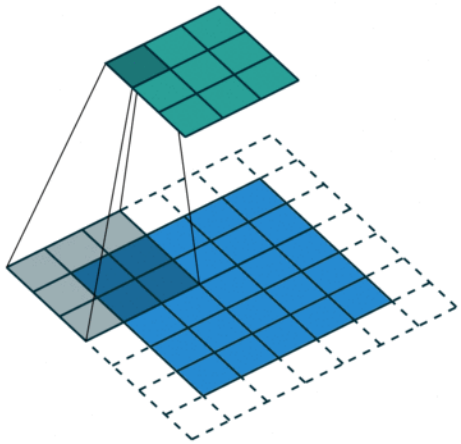
Padding
No Striding
$\mathbf{c} \in \mathbf{R}^{3\times3}$



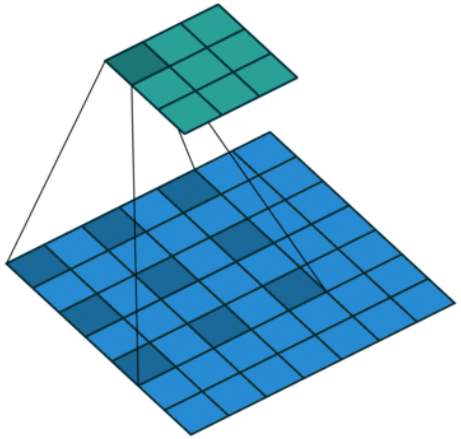
Padding
No Striding
$\mathbf{c} \in \mathbf{R}^{3\times3}$



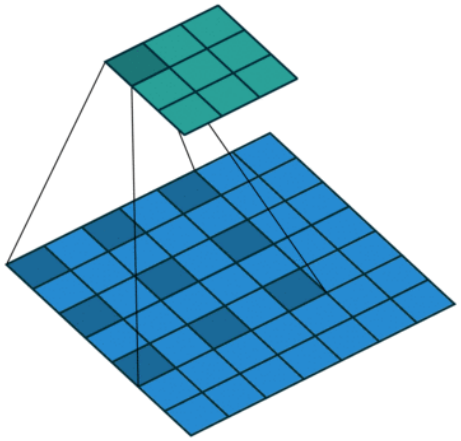
Padding
Striding
$\mathbf{c} \in \mathbf{R}^{3\times3}$



Padding
Striding
$\mathbf{c} \in \mathbf{R}^{3\times3}$



No Padding
Dilation
$\mathbf{c} \in \mathbf{R}^{3\times3}$



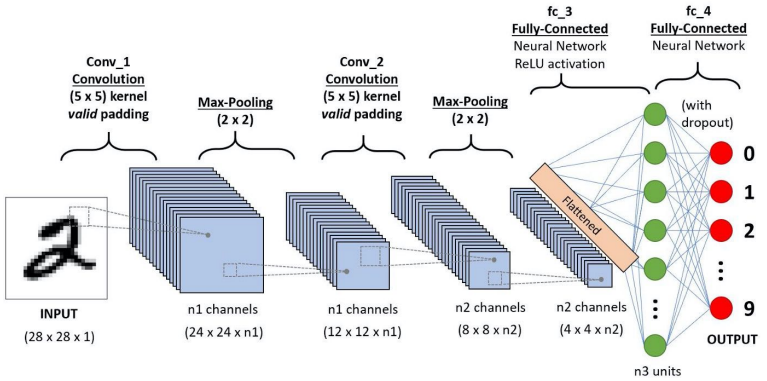
No Padding

Dilation

$\mathbf{c} \in \mathbf{R}^{3\times3}$

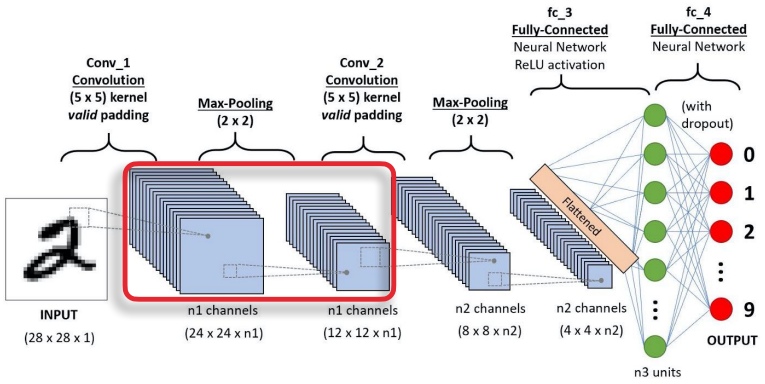
LeNet

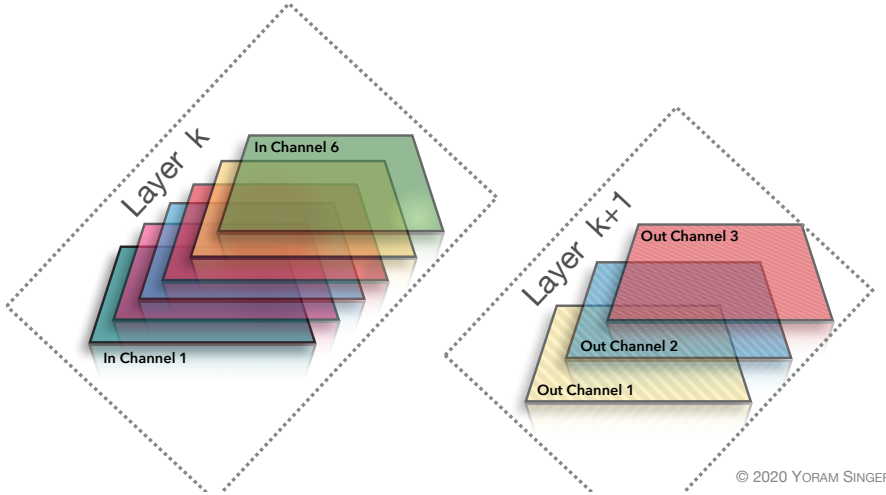
LeCun, Boser, Denker, Henderson, Howard, Hubbard, Jackel
Backpropagation applied to handwritten zip code recognition.
Neural Computation, 1989

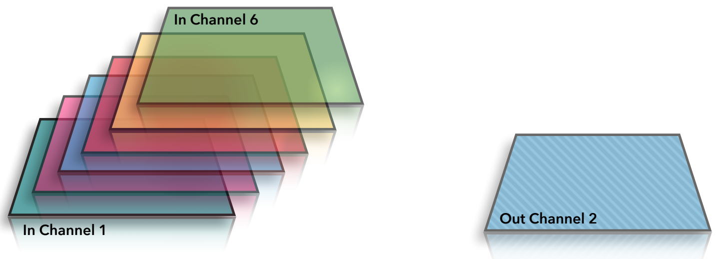


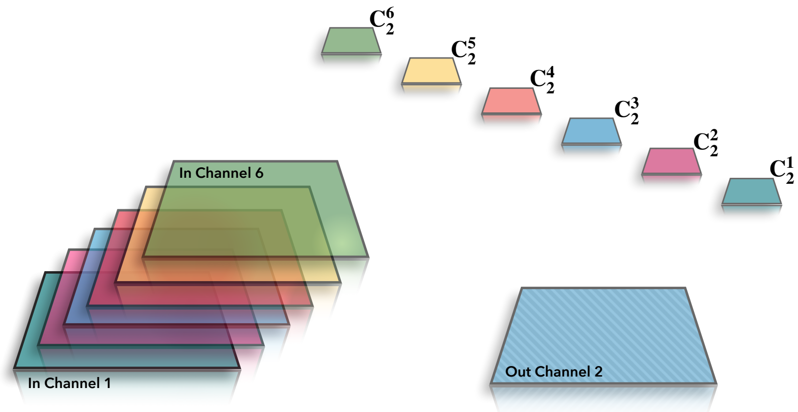
LeNet

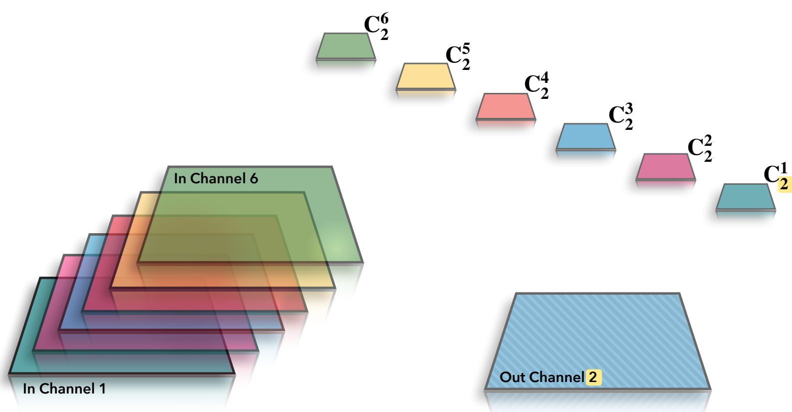
LeCun, Boser, Denker, Henderson, Howard, Hubbard, Jackel
Backpropagation applied to handwritten zip code recognition.
Neural Computation, 1989

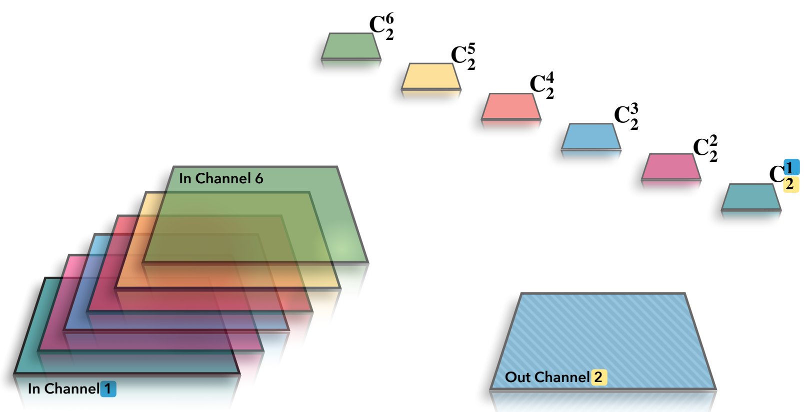


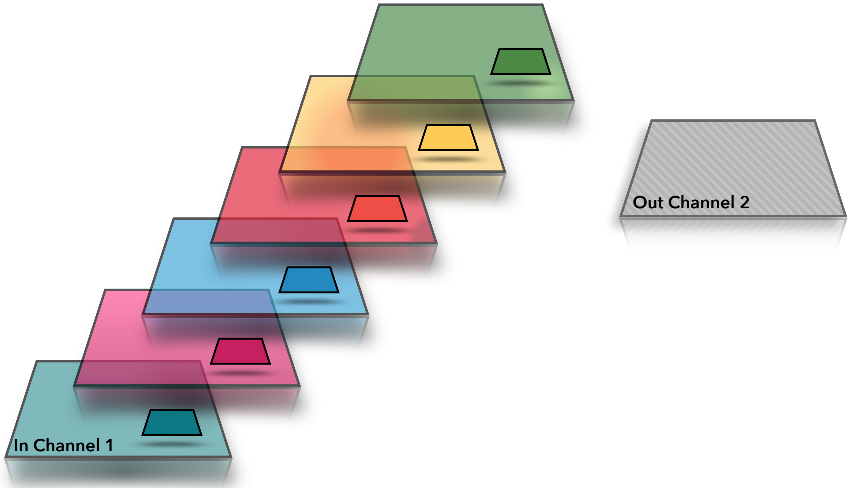


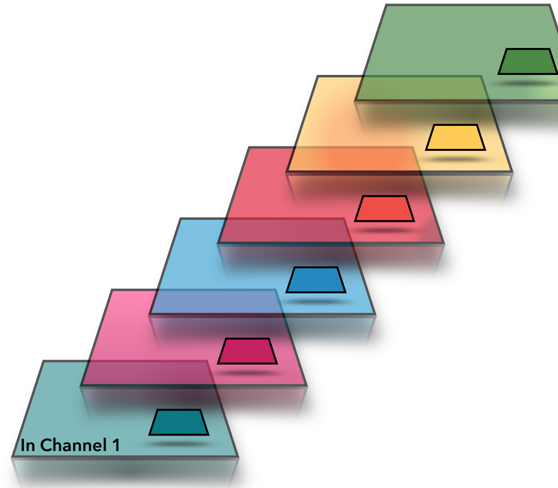

In Channel 6
In Channel 1
Out Channel 2


C_2^6
C_2^5
C_2^4
C_2^3
C_2^2
C_2^1
In Channel 6
In Channel 1
Out Channel 2

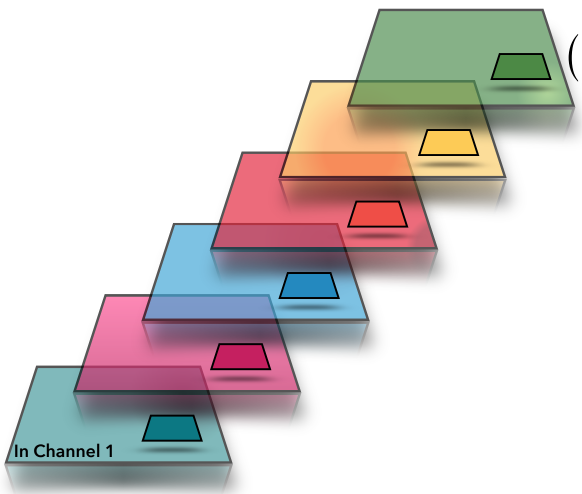

C_2^6
C_2^5
C_2^4
C_2^3
C_2^2
C_2^1
In Channel 6
In Channel 1
Out Channel 2


C_2^6
C_2^5
C_2^4
C_2^3
C_2^2
C_2^1
In Channel 6
In Channel 1
Out Channel 2

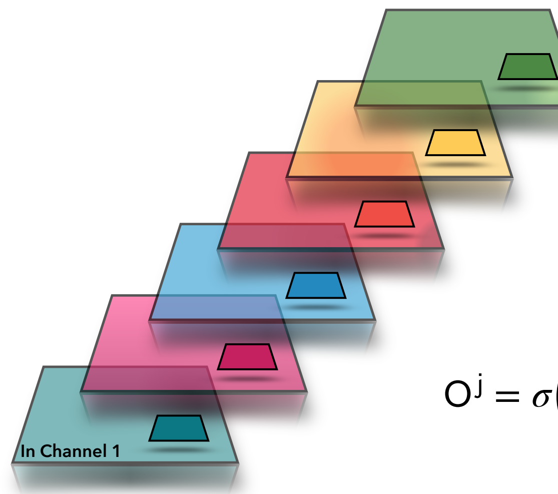




$$\left(\mathrm{In}^6 \circledast \mathbf{c}_2^6\right)[x, y]$$

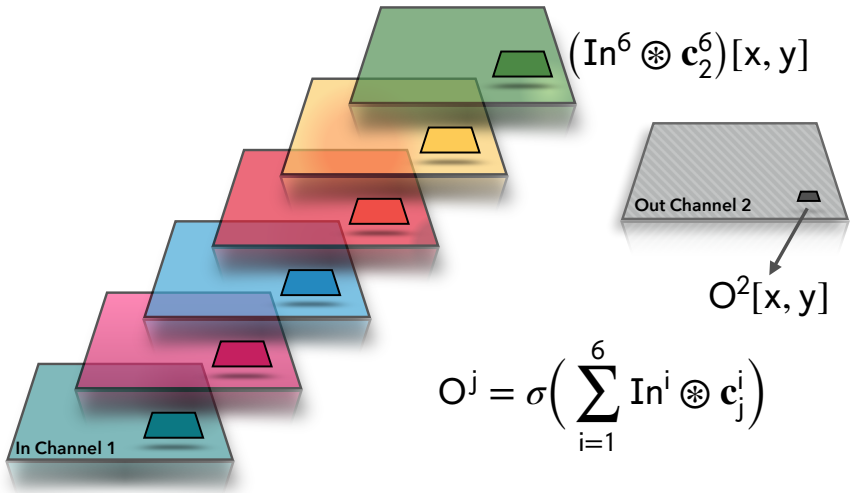


$$\left(\mathrm{In}^6 \circledast \mathbf{c}_2^6\right)[x, y]$$

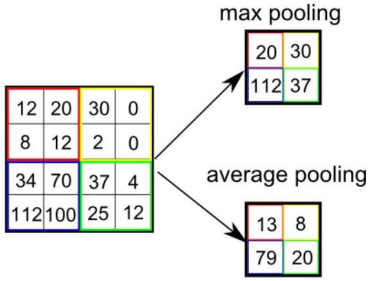


$$\left(\mathrm{In}^6 \circledast \mathbf{c}_2^6\right)[x, y]$$

$$\mathrm{O}^j = \sigma\left(\sum_{i=1}^{6} \mathrm{In}^i \circledast \mathbf{c}_j^i\right)$$



2D Pooling & Averaging



2xx2 Max-Pooling reduces #neurons by 4

Tool for building a pyramid of layers

Channels flattened layer d-2 or d-3

Two-three fully connected layers

2D Max & Avg Pooling

-2	5	-2	0
2	3	1	1
4	4	-2	1
4	4	3	2

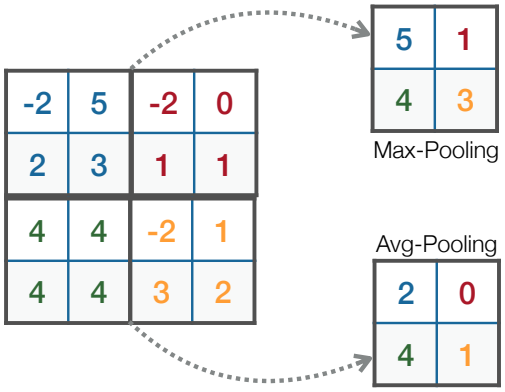
2D Max & Avg Pooling

-2	5	-2	0
2	3	1	1
4	4	-2	1
4	4	3	2

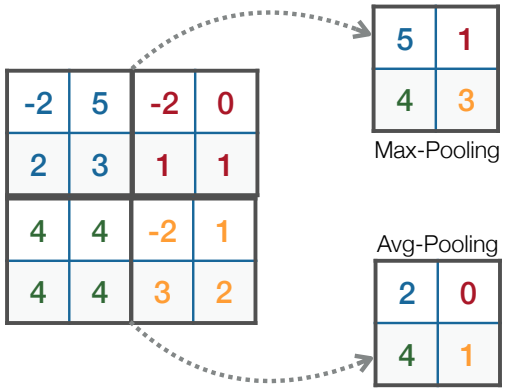
5	1
4	3

Max-Pooling

2D Max & Avg Pooling

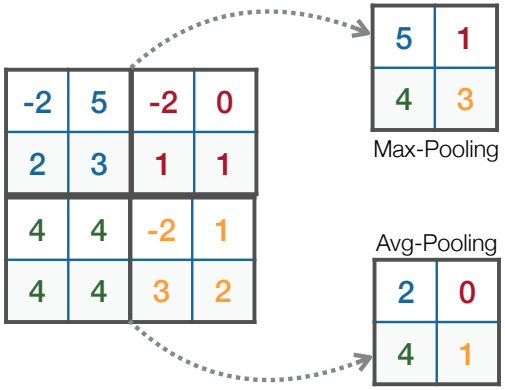


2D Max & Avg Pooling



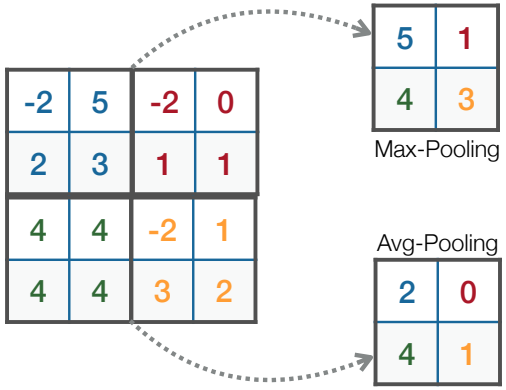
- Pooling:
  - Reduces #neurons
  - Forms pyramid of layers

2D Max & Avg Pooling



- Pooling:
  - Reduces #neurons
  - Forms pyramid of layers
- Channels flattened to vector before top layers

2D Max & Avg Pooling



- Pooling:
  - Reduces #neurons
  - Forms pyramid of layers
- Channels flattened to vector before top layers
- Top layers fully connected

Code

1. Dataset: CIFAR10 for image classification, more difficult than MNIST for OCR
2. Notebooks: cifar10_mlp & cifar10_conv
3. Uses PyTorch with autograd: no need to calculate gradients by hand
4. Main loop written explicitly — examine carefully
5. Load and read notebook once before running cell by cell