Unit 7_02 ConvNets – Introduction to Convolutional Neural Networks & Pooling

TFIP-AI Artificial Neural Networks and Deep Learning

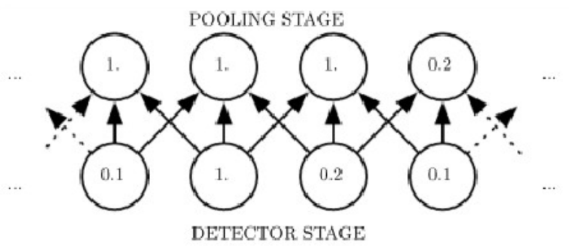
Outline

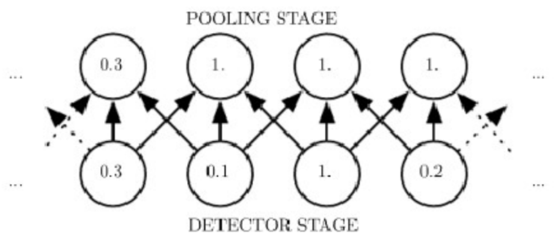
- Pooling Layer
- Architecture
    - Layer pattern
    - Layer sizing pattern
- LeNet 5
- Back Propagation

Pooling Layer

- Its function is to reduce the amount of parameters and computation in the network, and hence to also control overfitting.
- Pooling Layer operates independently on every depth slice.
- Most common form: Filters 2×2, Stride 2, not common to pad the input
- Input volume size: (W, H, D), filter size F, Stride S
- Output volume size
  - $W' = (W-F)/S + 1$, $H' = (H-F)/S + 1$
  - $D' = D$

Max Pooling





Max pooling

o[:,:,0]

-2	3	3
4	-6	-1
-2	2	1

o[:,:,1]

$\boxed{0}$	-1	5
5	0	3
-1	2	1

Filter 2 × 2
Stride 1
Max pooling

AVG Pooling

`o[:,:,0]`

-2	3	3
4	-6	-1
-2	2	1

`o[:,:,1]`

$\boxed{0}$	-1	5
5	0	3
-1	2	1

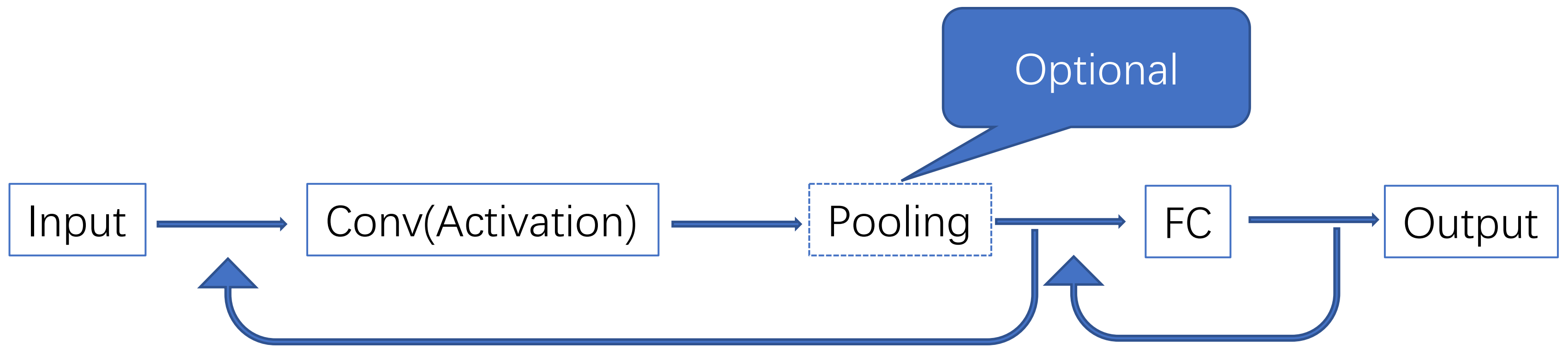
Filter 2 × 2
Stride 1
Avg pooling

$\longrightarrow$

Get rid of Pooling

- Many people dislike the pooling operation and think that we can get away without it.
- Larger stride could also reduce the representation.
- Discarding pooling layers has also been found to be important in training good generative models, such as variational autoencoders (VAEs) or generative adversarial networks (GANs). It seems likely that future architectures will feature very few to no pooling layers.

Architecture – Layer Pattern



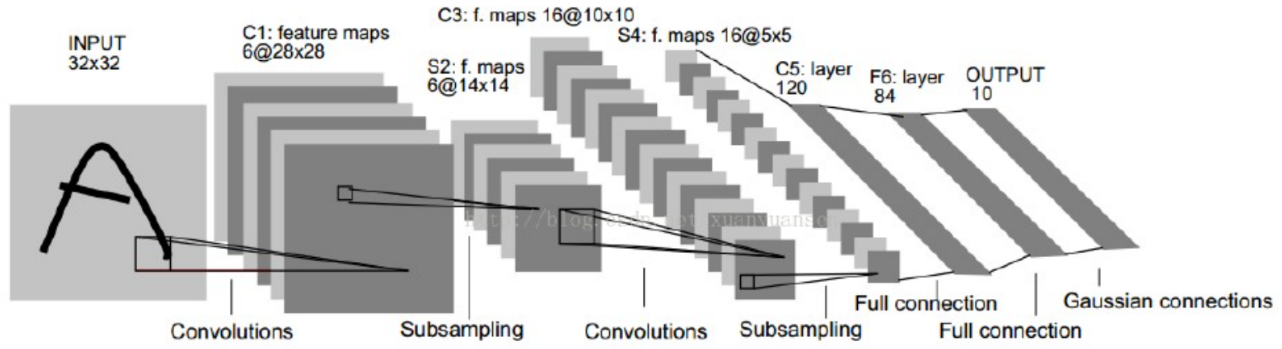


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Architecture – Layer sizing pattern

- Input layer
  - should be divisible by 2 many times
- Conv layers
  - Using small filters (3, or 5), stride=1, zero-padding only to retain the input size
- Pool layers
  - Max pooling with filter $2*2$, stride=2

Architecture – In practice

- d*on’ t be a hero*
- Instead of rolling your own architecture for a problem, you should look at whatever architecture currently works best on ImageNet,
- Download a pretrained model and finetune it on your data.
- You should rarely ever have to train a ConvNet from scratch or design one from scratch.

Parameters – C1

- C1 - ConvLayer
  - 6 kernel, each with size 5 by 5
  - Number of weights 5*5=25
  - Number of parameters 25 + bias 1 = 26
  - 6 kernel/filters, 26 * 6=156

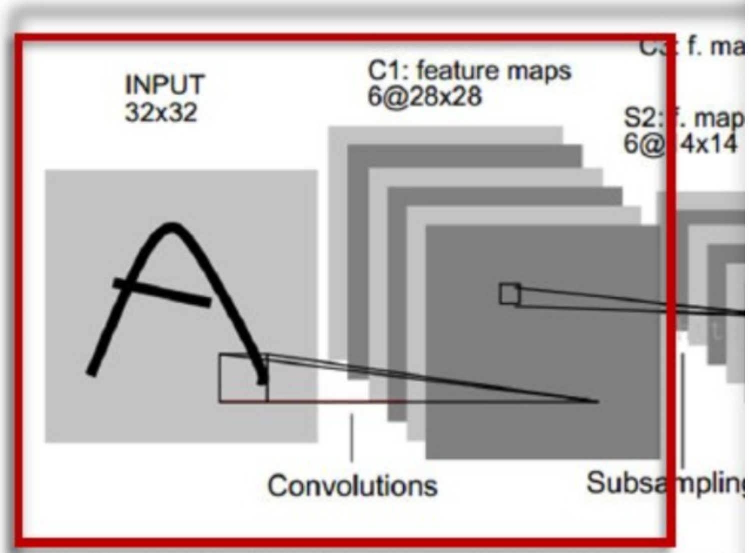


Fig. 2. Architecture of LeNet-5, a Convolutional Neural
whose weights are constrained to be identical.

Parameters – S2

- S2
  - 2 by 2 pooling
  - Pooling method: (a1+a2+a3+a4)*w+b
  - 2 * 6=12

a1	a2
a3	a4

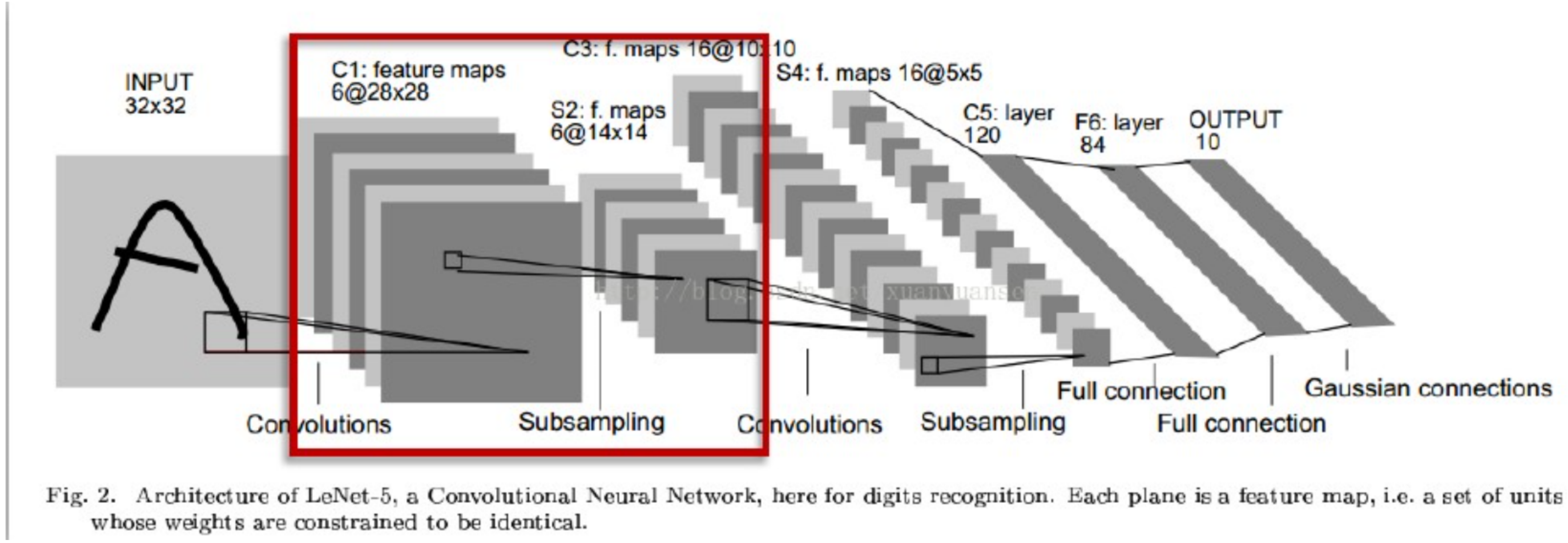


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Parameters – (C3)

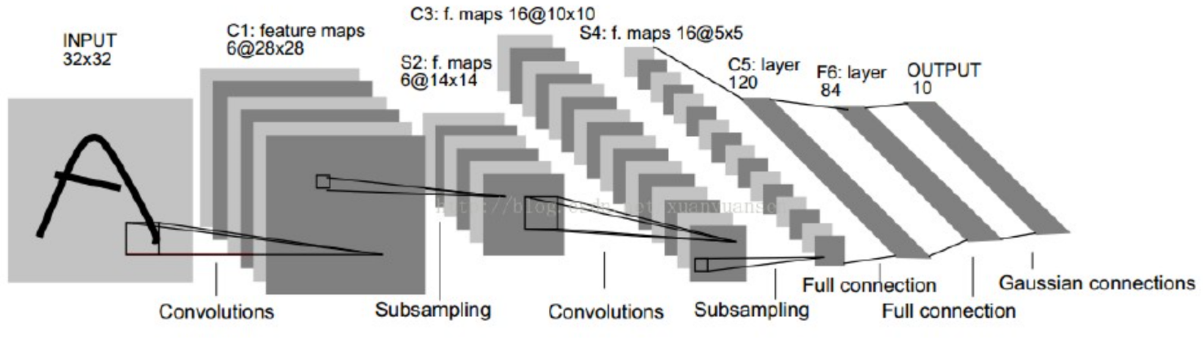
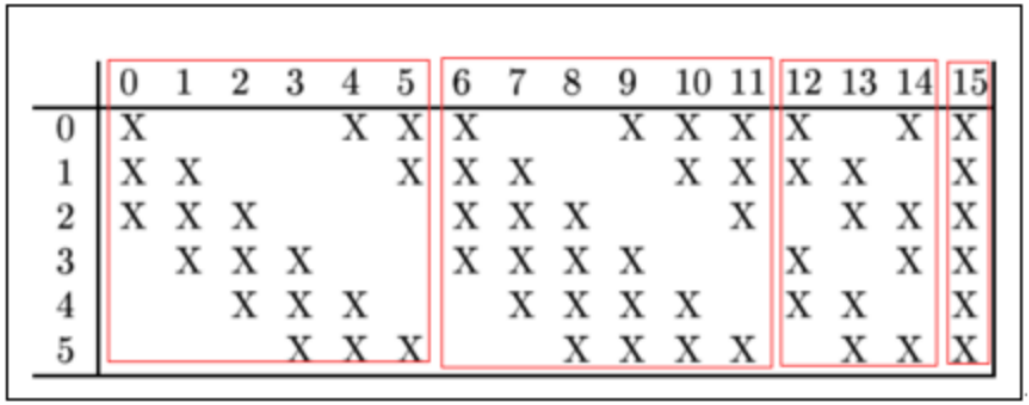
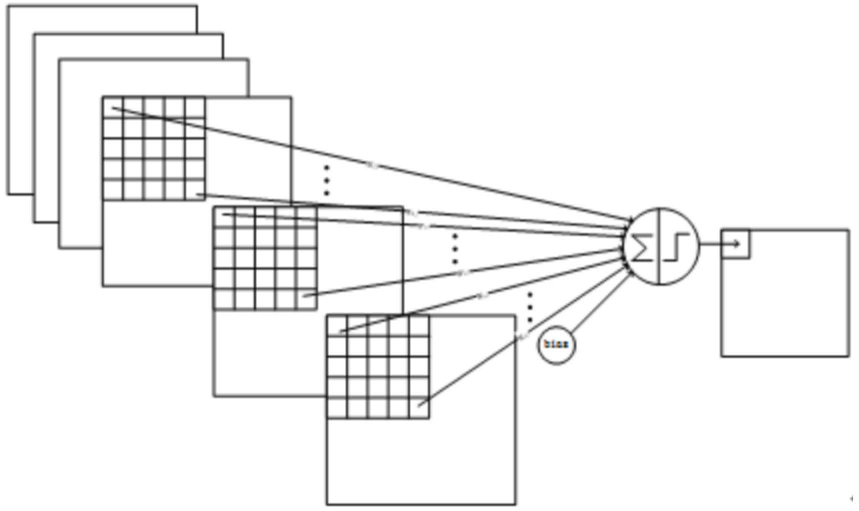


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

- S2(input): 6@14×14
- C3: 16@10×10
  - 6*(3*5*5+1) + 6*(4*5*5+1) + 3*(4*5*5+1)+1*(6*5*5+1) = 1516



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X



Parameters – (S4)

- C3: 16@10×10
- S4: 16@5×5
    - 2 by 2 pooling, (a1+a2+a3+a4)*w+b, 2*16=32

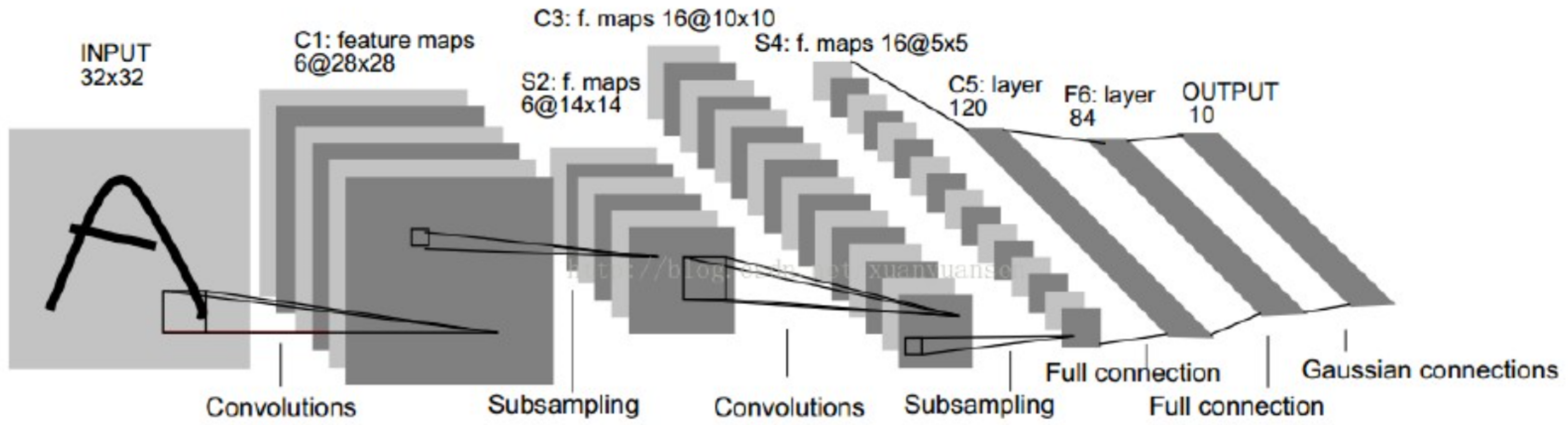


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Parameters – (S4 C5)

- S4: 16@5×5
- C5: size of input is the same as size of kernel, 120 kernels
  120*(16*5*5+1)=48120
- F6: Full connected, (120+1)*84

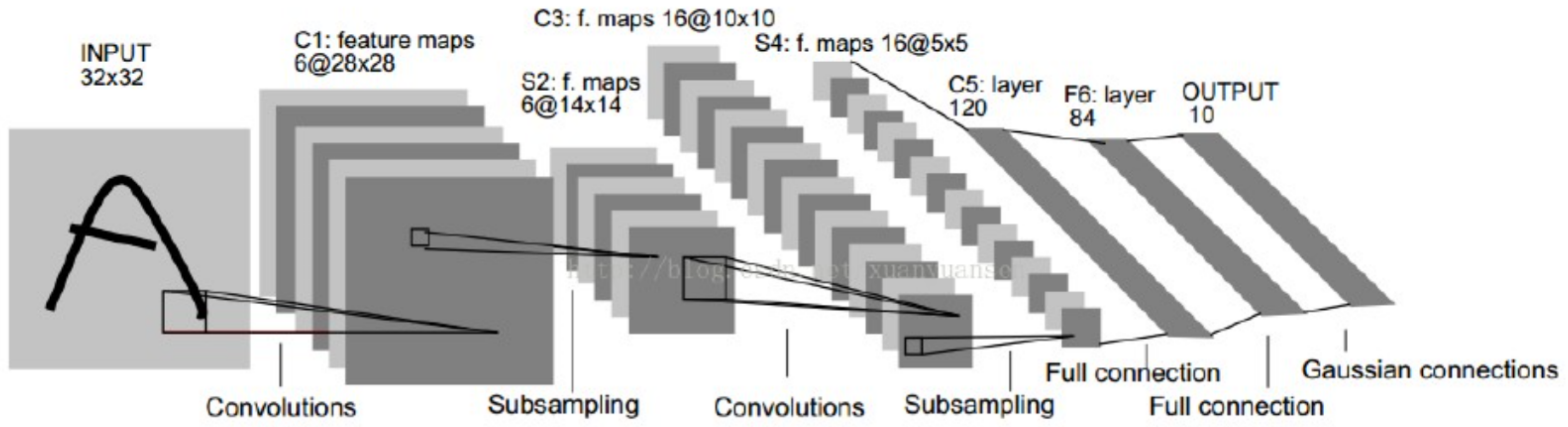


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

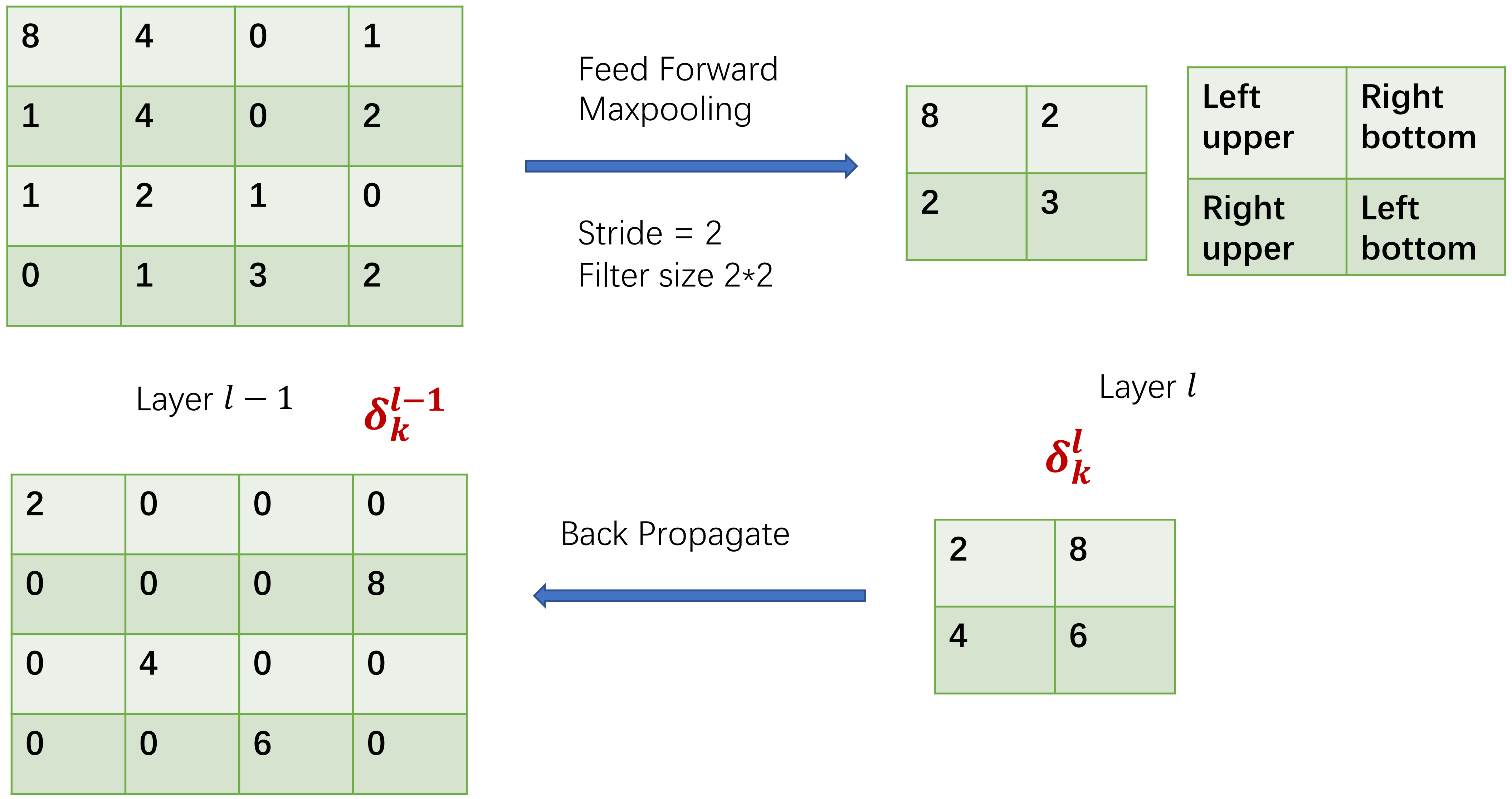
BP - Pooling layer sensitivity

- Pooling operation between layers: $(l-1) \rightarrow l$
- Up-sample
  - Max-pooling, 2 by 2
  - Should save the position of the max value when feed-forwarding
    - left-upper
    - right-bottom
    - right-upper
    - left-bottom

δ_k^l

2	8
4	6

2	0	0	0
0	0	0	8
0	4	0	0
0	0	6	0



8	4	0	1
1	4	0	2
1	2	1	0
0	1	3	2

8	2
2	3

Left upper	**Right bottom**
Right upper	**Left bottom**

2	8
4	6

2	0	0	0
0	0	0	8
0	4	0	0
0	0	6	0

Pooling layer sensitivity

- Pooling operation between: $(l-1) \rightarrow l$
- Up-sample
  - Avg-pooling, 2 by 2

δ_k^l

2	8
4	6

0.5	0.5	2	2
0.5	0.5	2	2
1	1	1.5	1.5
1	1	1.5	1.5

Convolution layer sensitivity

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} conv \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} = \begin{pmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{pmatrix}$$

$$z_{11} = a_{11}w_{11} + a_{12}w_{12} + a_{21}w_{21} + a_{22}w_{22}$$

$$z_{12} = a_{12}w_{11} + a_{13}w_{12} + a_{22}w_{21} + a_{23}w_{22}$$

$$z_{21} = a_{21}w_{11} + a_{22}w_{12} + a_{31}w_{21} + a_{32}w_{22}$$

$$z_{22} = a_{22}w_{11} + a_{23}w_{12} + a_{32}w_{21} + a_{33}w_{22}$$

$$\delta a_{11} = \delta z_{11}w_{11}$$

$$\delta a_{12} = \delta z_{11}w_{12} + \delta z_{12}w_{11}$$

$$\delta a_{13} = \delta z_{12}w_{12}$$

$$\delta a_{21} = \delta z_{11}w_{21} + \delta z_{21}w_{11}$$

$$\delta a_{22} = \delta z_{11}w_{22} + \delta z_{12}w_{21} + \delta z_{21}w_{12} + \delta z_{22}w_{11}$$

$$\delta a_{23} = \delta z_{12}w_{22} + \delta z_{22}w_{12}$$

$$\delta a_{31} = \delta z_{21}w_{21}$$

$$\delta a_{32} = \delta z_{21}w_{22} + \delta z_{22}w_{21}$$

$$\delta a_{33} = \delta z_{22}w_{22}$$

Kernel rotated !

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \delta z_{11} & \delta z_{12} & 0 \\ 0 & \delta z_{21} & \delta z_{22} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} conv \begin{pmatrix} w_{22} & w_{21} \\ w_{12} & w_{11} \end{pmatrix} = \begin{pmatrix} \delta a_{11} & \delta a_{12} & \delta a_{13} \\ \delta a_{21} & \delta a_{22} & \delta a_{23} \\ \delta a_{31} & \delta a_{32} & \delta a_{33} \end{pmatrix}$$

Convolution layer sensitivity – example

1	3
2	2

$\delta 1$

2	1
1	1

$\delta 2$

0.1	0.2
0.2	0.4

$k1$

-0.3	0.1
0.1	0.2

$k2$

0.4	0.2
0.2	0.1

180 degree Flipped $k1$

0.2	0.1
0.1	-0.3

180 degree Flipped $k2$

P=1
S=1

0.1	0.5	0.6
0.4	1.6	1.6
0.4	1.2	0.8

-0.6	0.1	0.1
-0.1	0.3	0.3
0.1	0.3	0.2