Convolution Neural Networks

Week 3

COMP6252 (Deep Learning Technologies)

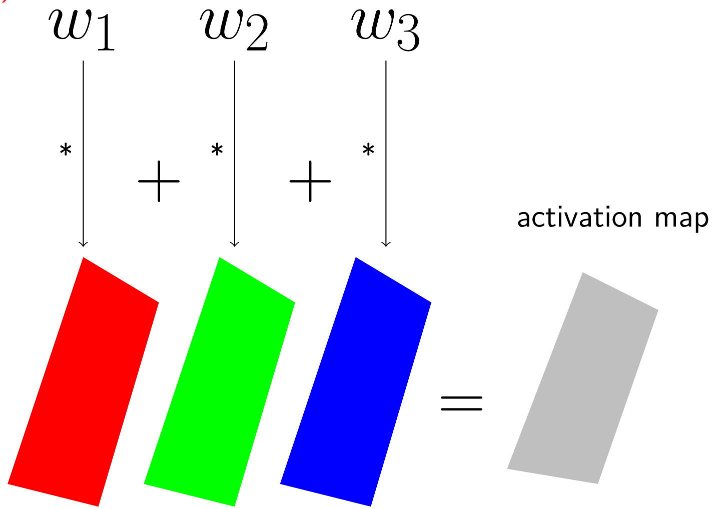
ECS, University of Southampton

28 April 2022

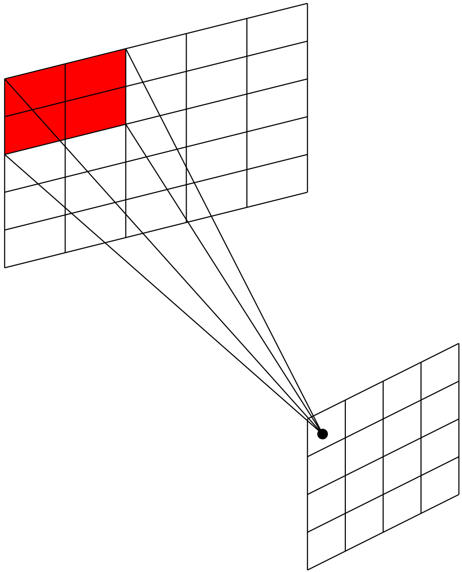
Introduction

- A convolution network usually has at least one convolution layer
- A convolution operation (slightly different from the usual definitio) is done by multiplying weights element-wise with a portion of the input
- The same operation with the same weights is repeated over all the input
- The set of weights are usually referred to as the **kernel**
- The result of the convolution is referred to as the **feature map** or **activation map**

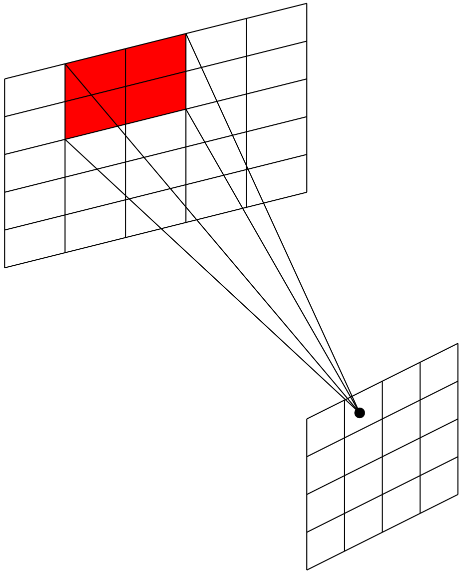
Convolution operation on an input image with 3 channels (RGB)



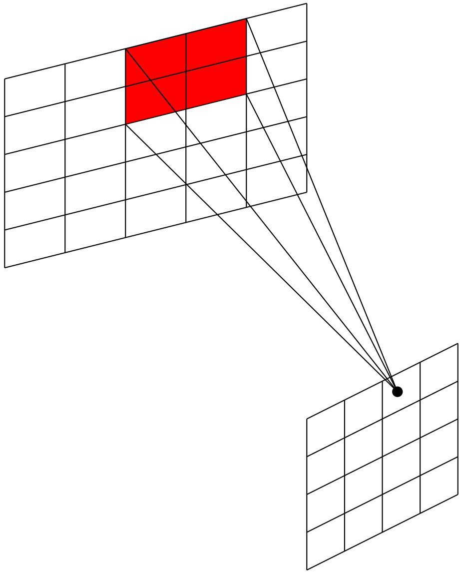
Conv operation on one channel



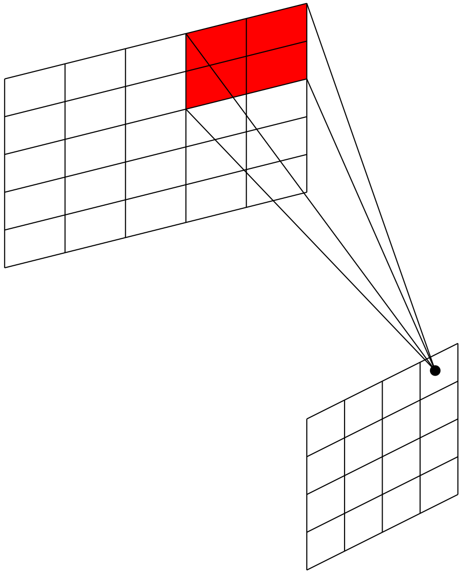
Conv operation on one channel



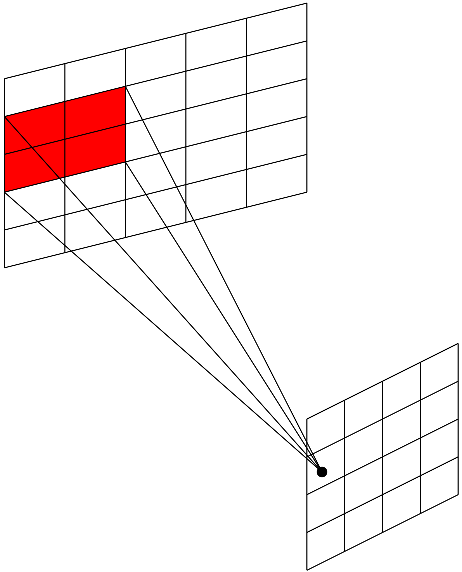
Conv operation on one channel



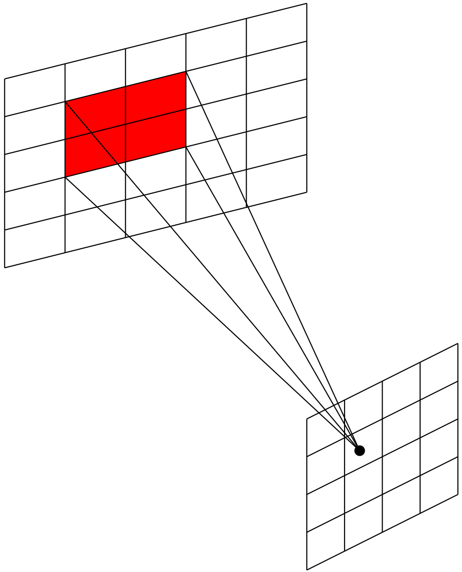
Conv operation on one channel



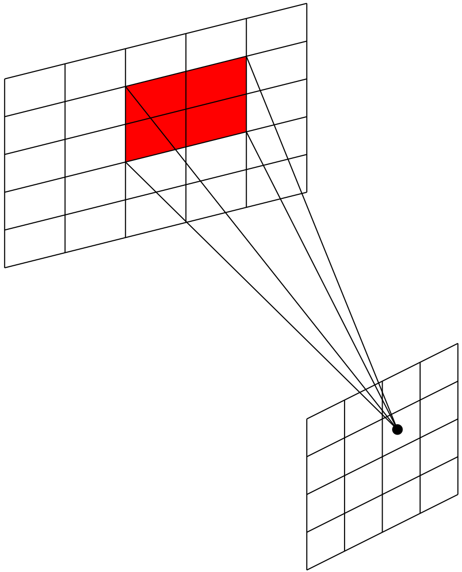
Conv operation on one channel



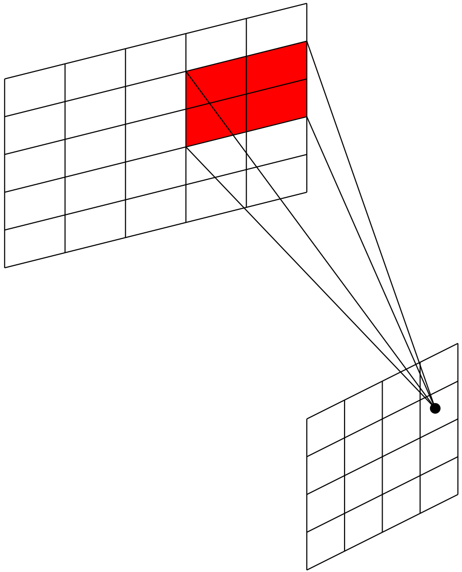
Conv operation on one channel



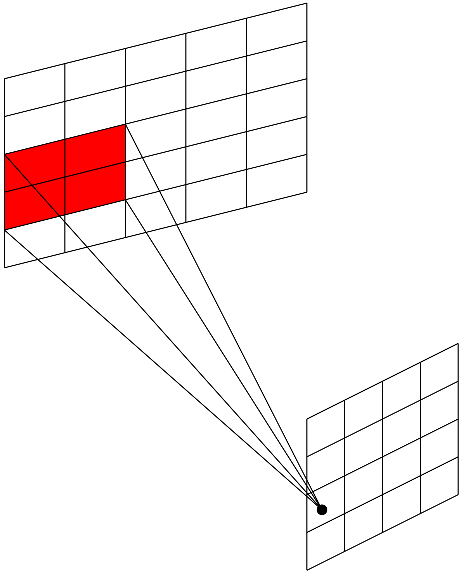
Conv operation on one channel



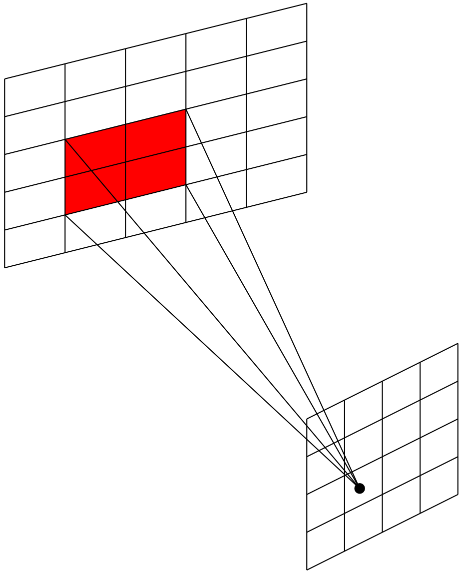
Conv operation on one channel



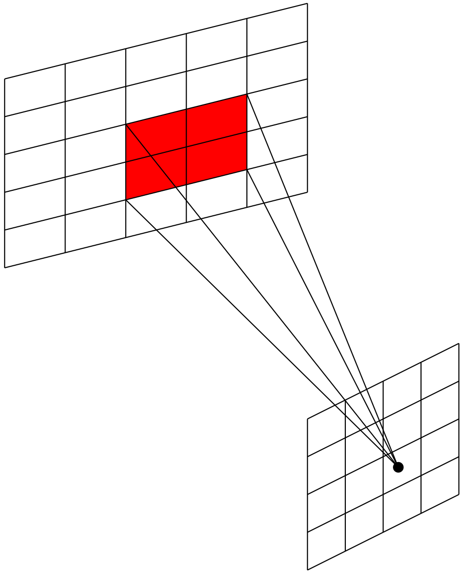
Conv operation on one channel



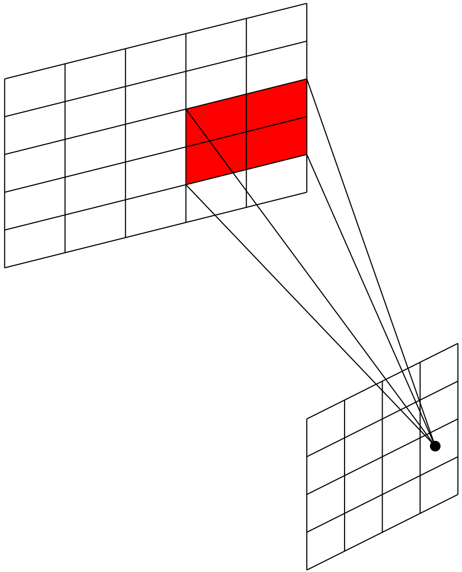
Conv operation on one channel



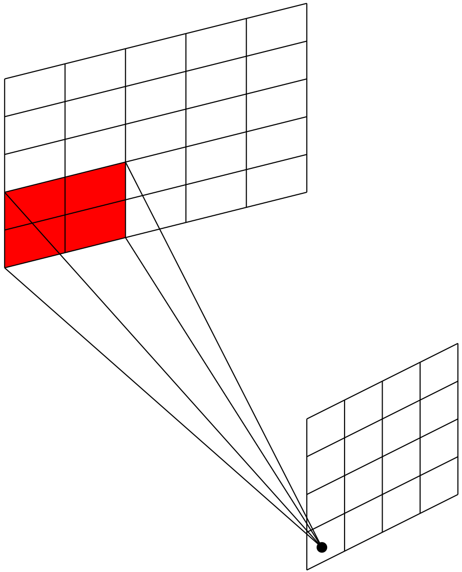
Conv operation on one channel



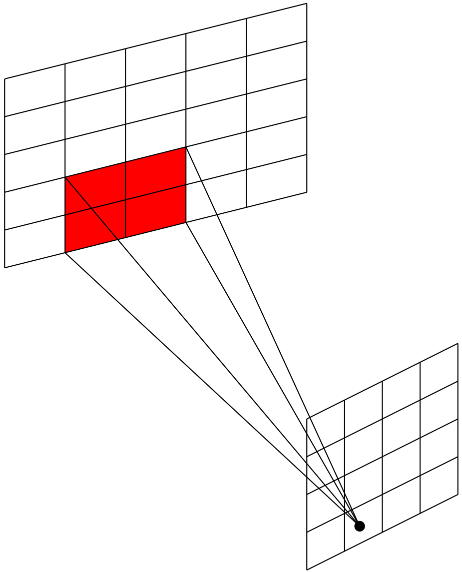
Conv operation on one channel



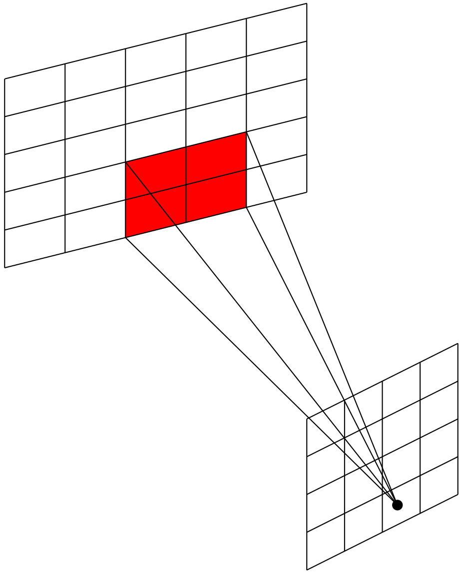
Conv operation on one channel



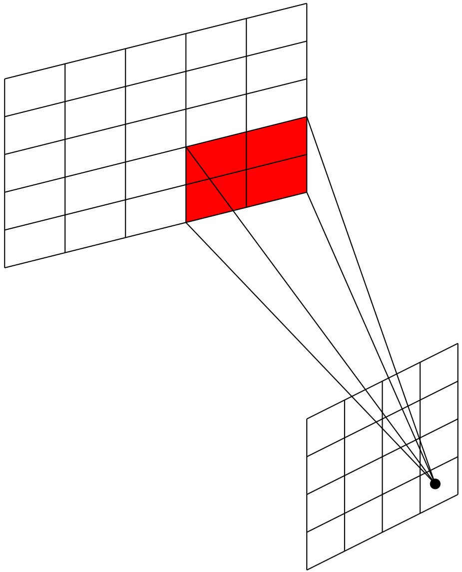
Conv operation on one channel



Conv operation on one channel



Conv operation on one channel



Example: Conv. operation on one channel

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

$\odot$

Kernel

1	-2
-3	4

activation map

$=$

6		

Example: Conv. operation on one channel

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

$\odot$

Kernel

1	-2
-3	4

activation map

$=$

6	8	

Example: Conv. operation on one channel

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

$\odot$

Kernel

1	-2
-3	4

activation map

$=$

6	8	2

Example: Conv. operation on one channel

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

$\odot$

Kernel

1	-2
-3	4

activation map

$=$

6	8	2
6		

Example: Conv. operation on one channel

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

$\odot$

Kernel

1	-2
-3	4

activation map

$=$

6	8	2
6	8	

Example: Conv. operation on one channel

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

$\odot$

Kernel

1	-2
-3	4

activation map

=

6	8	2
6	8	2

Example: Conv. operation on one channel

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

$\odot$

Kernel

1	-2
-3	4

activation map

$=$

6	8	2
6	8	2
6		

Example: Conv. operation on one channel

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

$\odot$

Kernel

1	-2
-3	4

activation map

$=$

6	8	2
6	8	2
6	8	

Example: Conv. operation on one channel

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

$\odot$

Kernel

1	-2
-3	4

activation map

$=$

6	8	2
6	8	2
6	8	-50

Add stride 2 example here

Stride and receptive field

- In the previous example we used a **stride** equal to 1
- The kernel **receptive field** was equal to 2x2
- The size of the output (activation map) was 3x3
- Are those numbers the same in all applications? No

Stride and receptive field

- In general one can use a stride of any size S. Stride of size 1 is the most common.
- Let $F \times F$ be the kernel **receptive field**
- Let $H_i \times W_i \times C_i$ be the size of the input
- The size of the resulting activation map is $H_o \times W_o$ where

$$H_o = \left\lfloor \frac{H_i - F}{S} \right\rfloor + 1$$
$$W_o = \left\lfloor \frac{W_i - F}{S} \right\rfloor + 1$$

Example

- Consider an input image of size (3,28,28)
- We are using the PyTorch convention with the channel dimension first.
- What would be the size of the activation map if a kernel of size 3x3 and stride 2 were used?
- Height and width are the same and equal to $\left\lfloor \frac{28-3}{2} \right\rfloor + 1 = 13$
- So the activation map has size 13x13
- Note that the output of **one** kernel is a 2-d object: the activation map.
- The number of output channels is determined by the **number** of kernels

Parameters

- Consider an image of size (3,28,28) used as input to 32 filters of size 3x3
- Since the input has 3 channels each filter has 3x3x3+1(bias)=28 Parameters
- With a stride of 1 the output has size (32,26,26), i.e., the layer has 32x26x26=21632 nodes
- Total number of parameters=32x28=896
- Contrast the above with a feedforward with the first layer having 32x26x26=21632 nodes
- It would have (3x28x28)*21632=50878464 parameters !(without bias)

Padding

- We saw previously that convolution reduces the size of the input
- When multiple layers are used the size is reduced at each layer
- Sometimes we don't want the size to change
- more importantly the "edges" of the input do not contribute as much as the "middle"
- One solution is to pad the input with zeros.

Zero padding

Input

1	2	1
4	2	3
2	1	1

0 padded

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

- The output size in the presence of padding

$$H_o = \left\lfloor \frac{H_i + 2P - F}{S} \right\rfloor + 1$$
$$W_o = \left\lfloor \frac{W_i + 2P - F}{S} \right\rfloor + 1$$

- Same formula as before if we consider the **effective** height $=H_i + 2P$ and width$=W_i + 2P$

Convolution operation mathematically

- Let f, i, j be the filter index,output height index, and output width respectively
- For a stride of 1 (most common), and bias per filter b_f
- The convolution operation is defined as

$$O_{f,i,j} = b_f + \sum_c \sum_{m,n} X_{c,i+m,j+n} * K_{f,c,m,n}$$

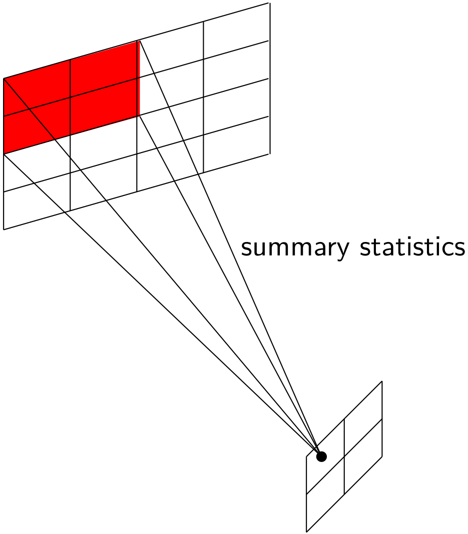
- If one includes the sample index s, needed in the case of batches

$$O_{s,f,i,j} = b_f + \sum_c \sum_{m,n} X_{s,c,i+m,j+n} * K_{f,c,m,n}$$

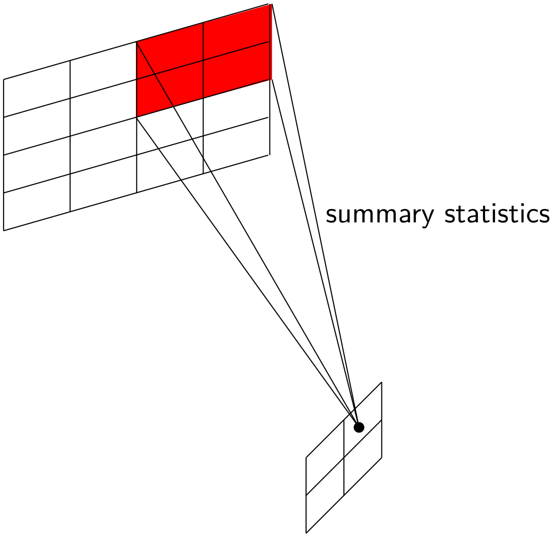
Pooling

- Typically, convolutional networks performs three steps
  1. Convolution operation
  2. followed by a Non-linear activation such as ReLU
  3. followed by **pooling**
- Pooling computes a summary statistics for a small area of the result
- Taking the max value is the most common pooling operation.

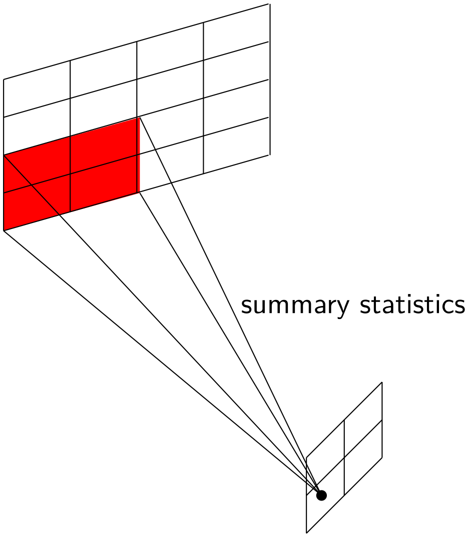
Pooling size 2x2, stride=2



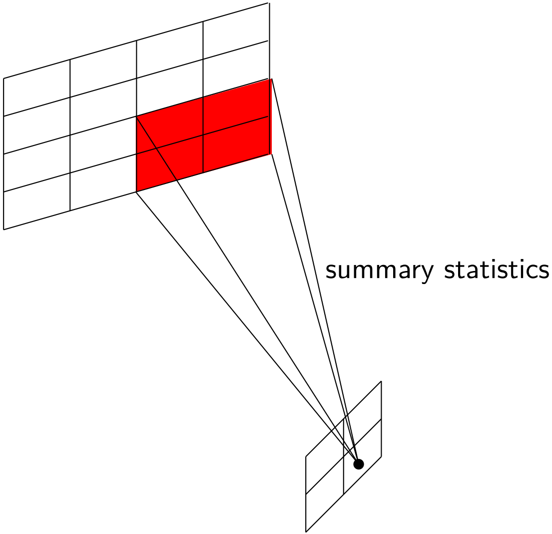
Pooling size 2x2, stride=2



Pooling size 2x2, stride=2



Pooling size 2x2, stride=2



Max pooling

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

Max Pool=

Result

6	

Max pooling

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

Max Pool=

Result

6	8

Max pooling

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

Max Pool=

Result

6	8
14	

Max pooling

Input

1	2	4	3
5	6	8	7
9	10	12	11
13	14	16	2

Max Pool=

Result

6	8
14	16