



Convolutional Neural Networks

CS5242

Lee Hwee Kuan & Wang Wei

Teaching assistant:

Connie Kou Khor Li, Ji Xin, Ouyang Kun

cs5242@comp.nus.edu.sg

Questions

SMS to **76767** OR <https://peerq.nus.edu.sg>.

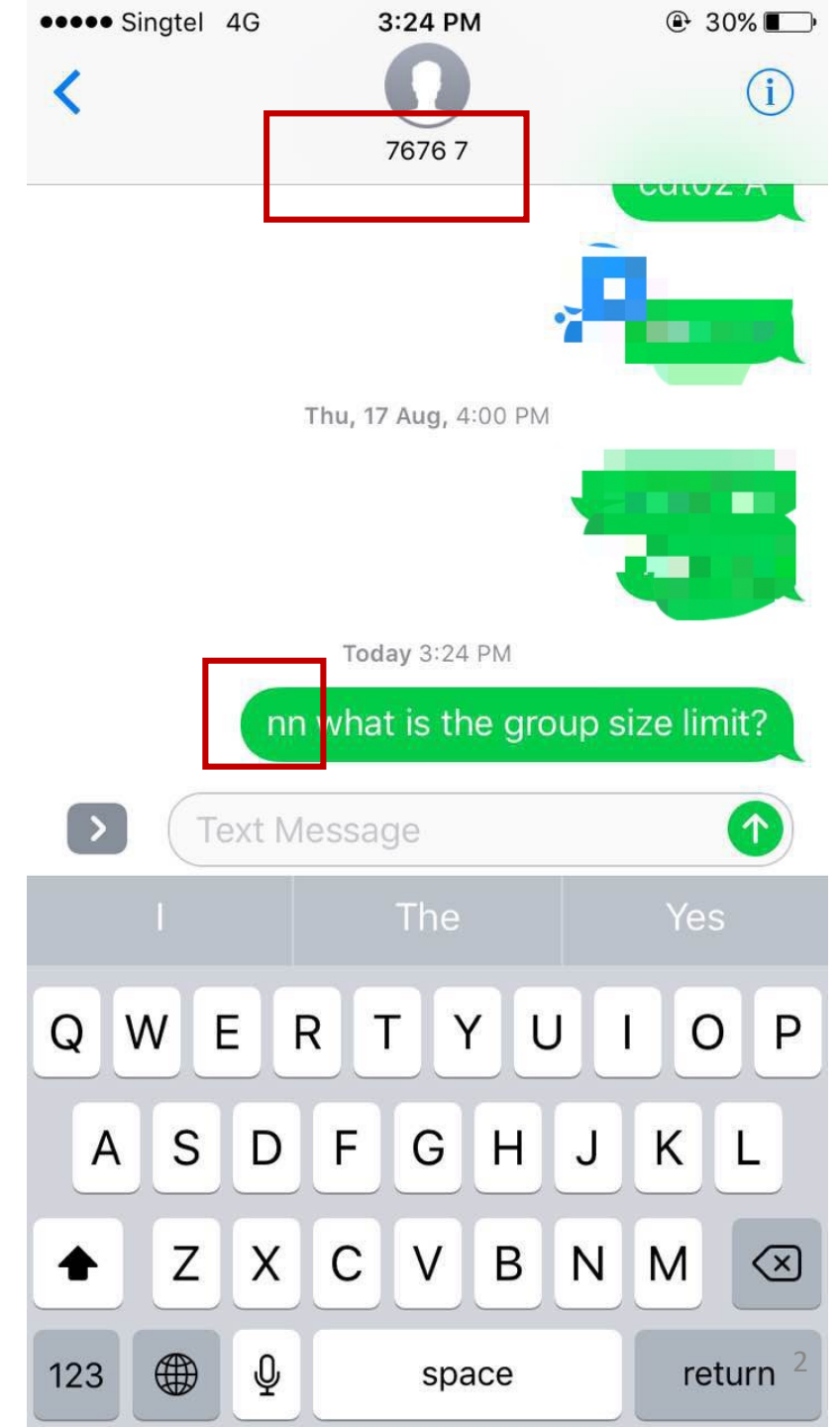
Content: <code><space><answer or question>

For exmaple: “nn what is the group size limit”

code

question

The code expires in one day.



Administrative

- Project
- Assignment

Roadmap

- Convolution and Pooling
 - 1D convolution
 - 2D convolution
 - Pooling
- Architectures
 - AlexNet, VGG, InceptionNet, ResNet, DenseNet, XceptionNet
- Training
 - Activation functions
 - Normalization functions
 - Hyper-parameters
- Applications
 - Classification, Detection, NeuralStyle

Intended learning outcome

Know	Know the difference between convolution and MLP
Understand	Understand the characteristic of convolution layers
Calculate	Calculate the size of convolution outputs and kernel size
Implement	Implement 1D and 2D conv operations from scratch

Last week

- MLP

- Simple model
- Good for short feature vectors
- Problems:
 - $W^i \in R^{|h^{i-1}| \times |h^i|}$
 - $2500 \times 2000 = 5,000,000$
 - Too many parameters
→ overfitting

- CNN

- Complex model
- Good for images
- How CNN work?
- Why CNN work?

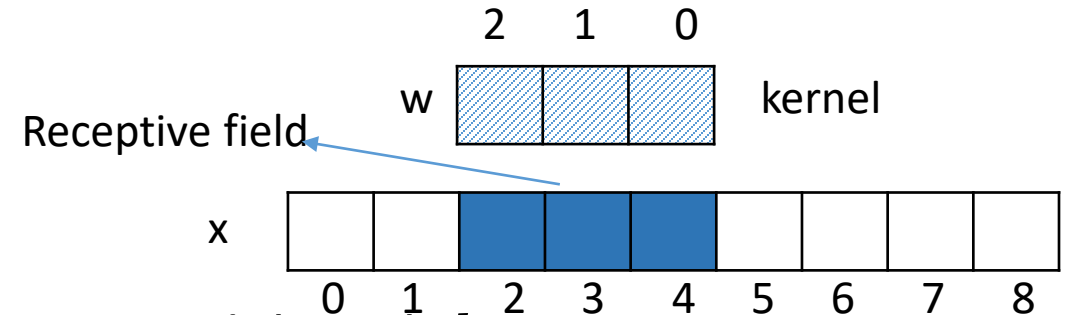
NN architecture	Dataset	Distortions	Test Error [%]
MLP:2500-2000-1500-1000-500-10	MNIST	no	1.47
MLP:2000-2000-2000-2000-2000-2000-10	MNIST	no	1.531 ± 0.051
MLP:1500-1500-1500-1500-1500-1500-10	MNIST	no	1.513 ± 0.052
MLP:1000-1000-1000-1000-1000-1000-1000-1000-10	MNIST	no	1.628 ± 0.035
MLP:1000-1000-1000-1000-1000-1000-1000-10	MNIST	no	1.542 ± 0.052
MLP:1000-1000-1000-1000-1000-1000-10	MNIST	no	1.517 ± 0.069
MLP:1000-1000-1000-1000-1000-1000-10	MNIST	no	1.529 ± 0.078
MLP:1000-1000-1000-1000-1000-10	MNIST	no	1.571 ± 0.046
MLP:1000-1000-1000-1000-10	MNIST	no	1.549 ± 0.038
MLP:1000-1000-1000-10	MNIST	no	1.650 ± 0.030
MLP:500-500-500-500-500-500-500-10	MNIST	no	1.744 ± 0.038
MLP:500-500-500-500-500-500-10	MNIST	no	1.702 ± 0.064
MLP:500-500-500-500-500-10	MNIST	no	1.719 ± 0.069
MLP:500-500-500-500-10	MNIST	no	1.728 ± 0.028
MLP:500-500-500-10	MNIST	no	1.765 ± 0.040
MLP:2000-1500-1000-500-10	MNIST	5% translation	0.94
MLP:2500-2000-1500-1000-500-10	MNIST	affine + elastic	0.35
MLP committee:2500-2000-1500-1000-500-10	MNIST	affine + elastic	0.31
CNN 20M-40M-60M-80M-100M-120M-150N	MNIST	affine + elastic	0.35

Source from: <http://people.idsia.ch/~cirosan/results.htm>

Last week

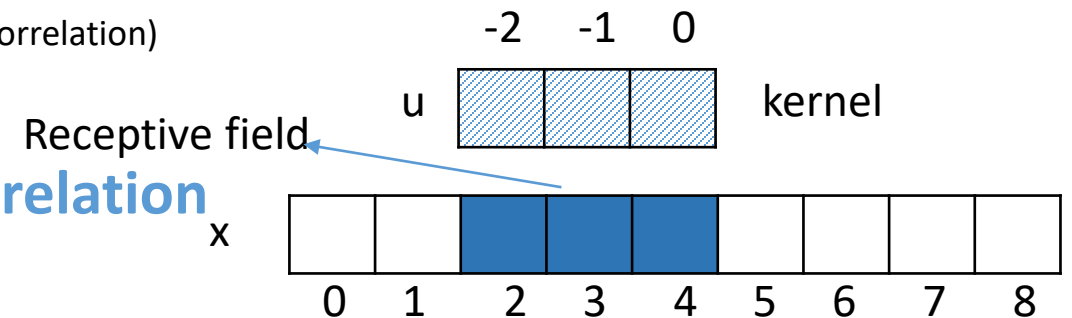
- Formal definition

- $y_t = \sum_{i=-\infty}^{\infty} w_i \times x_{t-i}$
- w is called **kernel**; the **parameters** to be trained; length k
- x is the input; length l
- the input area, i.e. $t, t-1, \dots, t-(k-1)$ is called the receptive field for CNN
- y_t is the output feature; length o



- Cross-correlation (<https://en.wikipedia.org/wiki/Cross-correlation>)

- $y_t = \sum_{i=-\infty}^{\infty} u_i \times x_{t+i}, u_i = w_{-i}$
- **In CNN, convolution refers to cross-correlation**



Padding

- Manual padding (p)
 - Kernel size/length: k
 - Input length: l
 - Output feature values for $p = 1$
 - $3 \times 1 + 2 \times 0 + 2 \times 2 = 7$
 - $3 \times 0 + 2 \times 2 + 2 \times 2 = 8$
 - $3 \times 2 + 2 \times 2 + 2 \times 3 = 16$
 - $3 \times 2 + 2 \times 2 + 2 \times 1 = 14$
 - $3 \times 3 + 2 \times 1 + 2 \times 0 = 11$
- Torch, PyTorch, Caffe, SINGA

w	3	2	2				3	2	2
x	?	1	0	2	2	3	1	?	

w			3	2	2				
x	1	0	2	2	3	1	0		

w				3	2	2			
x	1	0	2	2	3	1	0		

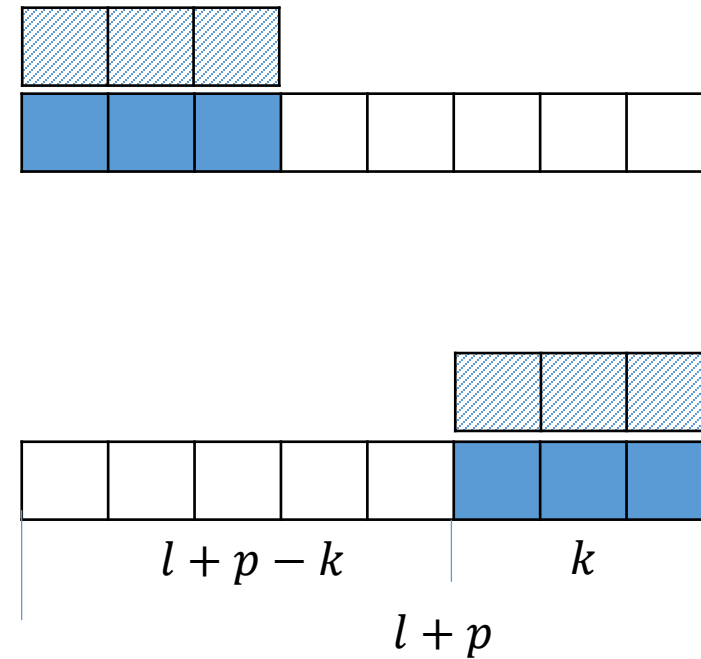
w					3	2	2		
x	1	0	2	2	3	1	0		

w						3	2	2	
x	1	0	2	2	3	1	0		

w							3	2	2
x	1	0	2	2	3	1	0		

Padding

- Manual padding (p)
 - Kernel size/length: k
 - Input length: l
 - # outputs $o = l + p - k + 1$



Padding

- Valid/No padding ($p = 0$)
 - # inputs denoted as l
 - # outputs $o = l - k + 1 = 6 - 3 + 1 = 4$
 - Output feature values
 - $3 \times 1 + 2 \times 0 + 2 \times 2 = 7$
 - $3 \times 0 + 2 \times 2 + 2 \times 2 = 8$
 - $3 \times 2 + 2 \times 2 + 2 \times 3 = 16$
 - $3 \times 2 + 2 \times 2 + 2 \times 1 = 14$
 - Outputs become shorter
 - TensorFlow, Keras

w	<div>322</div>							<div>322</div>		
x	?	<div>1</div>	<div>0</div>	<div>2</div>	<div>2</div>	<div>3</div>	<div>1</div>	?		

w			3	2	2		
x		1	0	2	2	3	1

w			3	2	2	
x	1	0	2	2	3	1

w				3	2	2
x	1	0	2	2	3	1

W				3	2	2
x	1	0	2	2	3	1

Padding

- Same padding ($p?$)
 - $l + p - k + 1 = l$
 - $p = k - 1 = 2$
 - Left padding = $\lfloor p/2 \rfloor$
 - Right padding = $\lceil p/2 \rceil$
 - Output values
 - $3 \times 0 + 2 \times 1 + 2 \times 0 = 2$
 - $3 \times 1 + 2 \times 0 + 2 \times 2 = 7$
 - $3 \times 0 + 2 \times 2 + 2 \times 2 = 8$
 - $3 \times 2 + 2 \times 2 + 2 \times 3 = 16$
 - $3 \times 2 + 2 \times 3 + 2 \times 1 = 14$
 - $3 \times 3 + 2 \times 1 + 2 \times 0 = 11$
 - TensorFlow, Keras

w		3	2	2				3	2	2	
x		?	1	0	2	2	3	1		?	

w		3	2	2							
x	0	1	0	2	2	3	1	0			

w			3	2	2						
x	0	1	0	2	2	3	1	0			

w				3	2	2					
x	0	1	0	2	2	3	1	0			

w					3	2	2				
x	0	1	0	2	2	3	1	0			

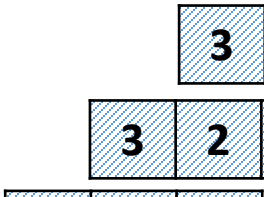
w						3	2	2			
x	0	1	0	2	2	3	1	0			

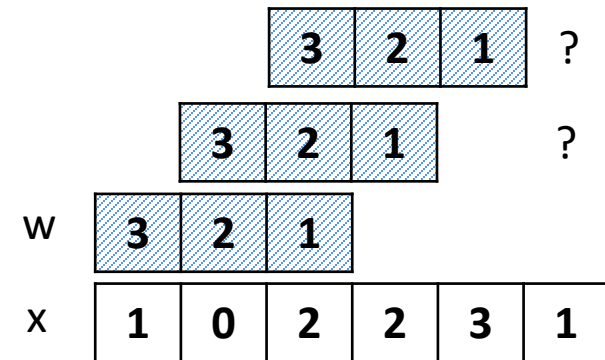
w							3	2	2		
x	0	1	0	2	2	3	1	0			

Why Padding

- $o = l - k + 1 \leq l - 1$ ($k \geq 2$)
 - By stacking multiple convolution layers, the output is shorter and shorter
 - Less information per layer
- Information loss at the boundary
 - If the image is cropped from a big image, the boundary pixels could also be important

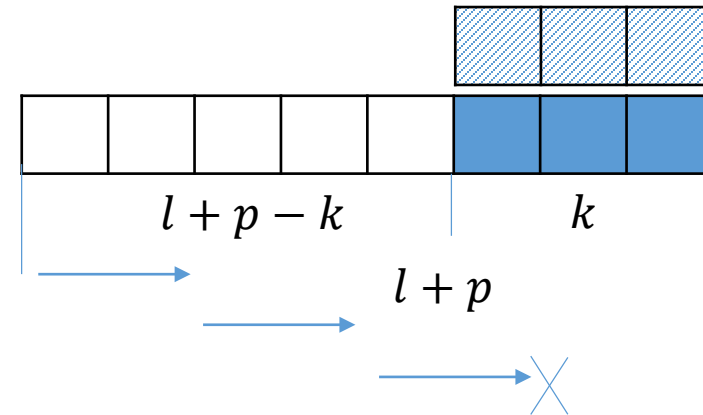
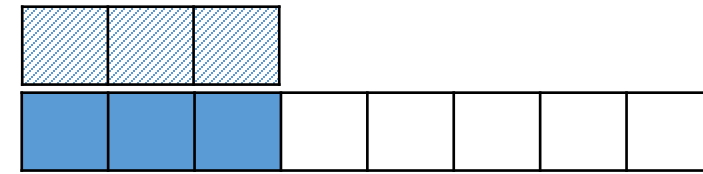
Stride

- How many steps to move towards the next receptive field
 - $s=1$, every receptive field is considered \rightarrow many outputs
 - $s>1$, some receptive fields are skipped.
 - Miss some (redundant) information
 - Faster
 - Fewer outputs
- 
- The diagram shows a 3x3 grid of squares. The top-right 2x2 subgrid is highlighted in blue. The squares in this subgrid contain the numbers 3, 2, 3, and 2 from top-left to bottom-right. The bottom-left square of the 3x3 grid is empty.



Stride

- $$o = \left\lfloor \frac{l+p-k}{s} \right\rfloor + 1$$



Stride

- Exact matching
 - With padding p ($=1$)
 - $o = \left\lfloor \frac{l+p-k}{s} \right\rfloor + 1$
 - $(6+1-3)/2+1=3$

W	3	2	1			
X	1	0	2	2	3	1

w	3	2	2				
x	1	0	2	2	3	1	0

w			3	2	2		
x	1	0	2	2	3	1	0

w					3	2	2
x	1	0	2	2	3	1	0

Stride

- Not exact matching
 - With padding p (=2)
 - $o = \left\lfloor \frac{l+p-k}{s} \right\rfloor + 1$
 - $(6+2-3)/2+1=3$

W	3	2	1			
X	1	0	2	2	3	1

w	3	2	2					
x	1	0	2	2	3	1	0	0

W	<table><tr><td>3</td><td>2</td><td>2</td></tr></table>					3	2	2		
3	2	2								
X	1	0	2	2	3	1	0	0		

w				3	2	2		
x	1	0	2	2	3	1	0	0

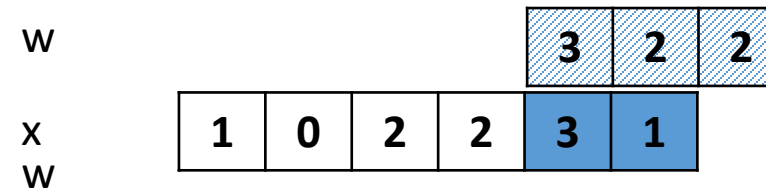
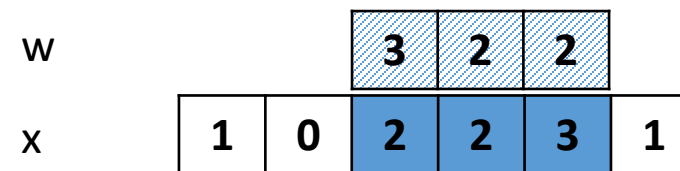
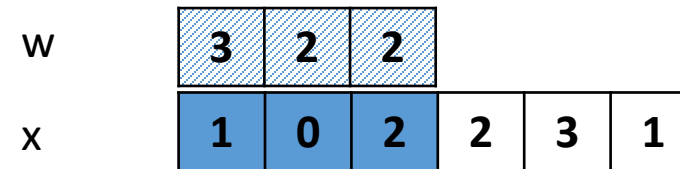
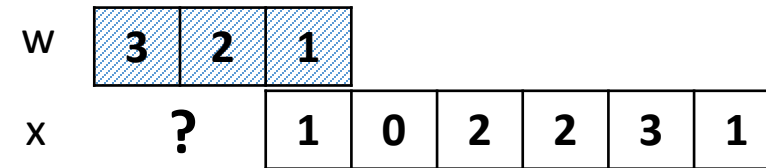
w							3	2	2
x	1	0	2	2	3	1	0	0	



Stride (for Tensorflow)

- Given stride s , what is the padding size if we want Valid padding?

$$p = 0, o = \left\lfloor \frac{l+p-k}{s} \right\rfloor + 1 = 2$$



Stride (for Tensorflow)

- Given stride s , what is the padding size if we want Same padding?

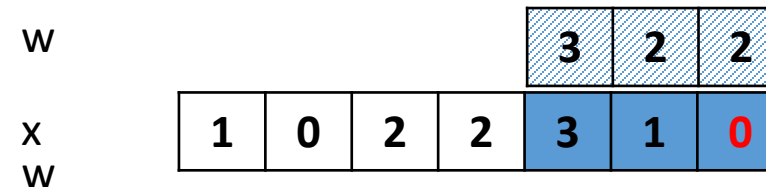
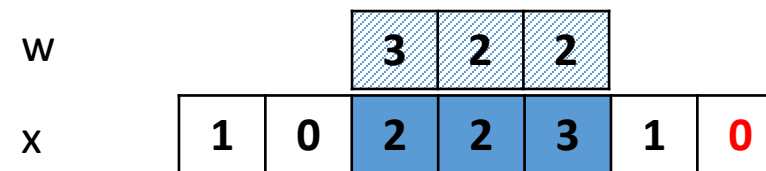
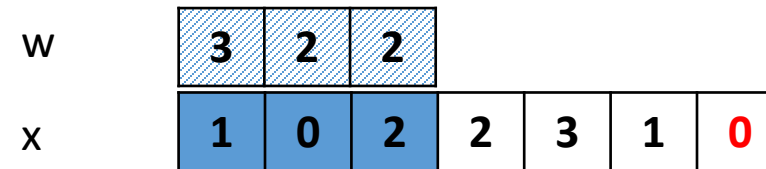
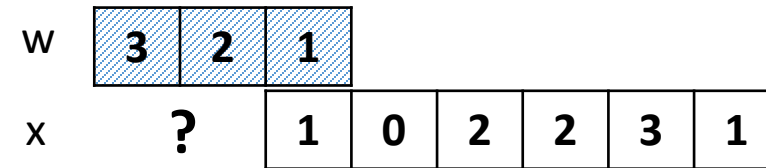
$$o = \left\lceil \frac{l}{s} \right\rceil$$

$$o = \left\lceil \frac{l+p-k}{s} \right\rceil + 1$$

$$\frac{l+p-k}{s} \geq o - 1$$

$$p \geq s(o - 1) + k - l$$

$$p = \max(s(o - 1) + k - l, 0)$$



Question

- Given input size, kernel size, stride $s=2$ and padding type, what is the padding size and the output length?
 - $l = 224, k = 5, s = 2$
 - *SMS to **76767***, e.g: qn1 xxxx

Code	Question
Qn1	Valid, p=?
Qn2	Valid, o=?
Qn3	Same, p=?
Qn4	Same, o=?

Implementation

- For loop

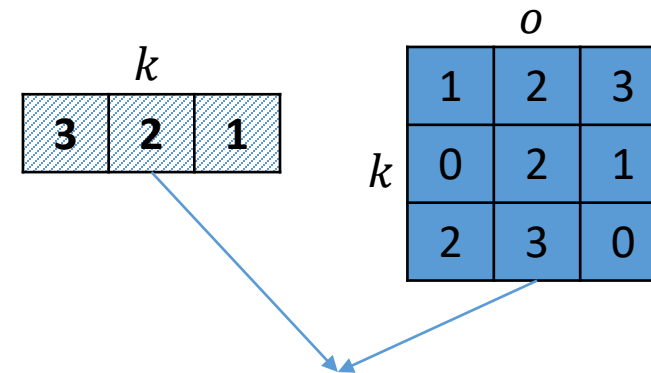
```
for t in range(o):
    for i in range(k):
        y[t] += w[k]*x[t+k]
```

- Row to column

```
for t in range(o):
    for i in range(k):
        a[i, t]= x[t+k]
y = dot(w, a)
```

W	3	2	1						
X	?	1	0	2	2	3	1		

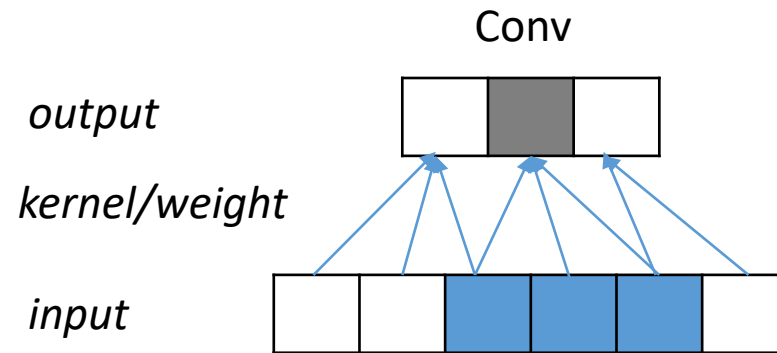
1	0	2	2	3	1	0
1	0	2	2	3	1	0
1	0	2	2	3	1	0



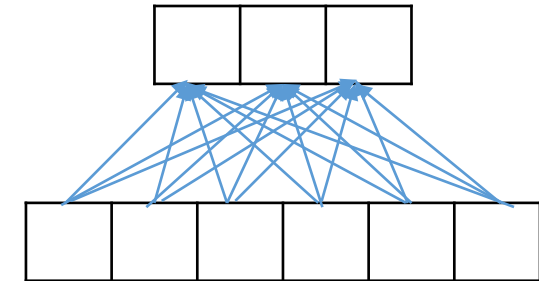
dot() **Convolution -> affine transformation**

Properties (Why Convolution?)

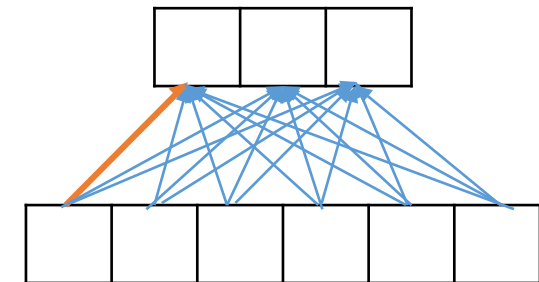
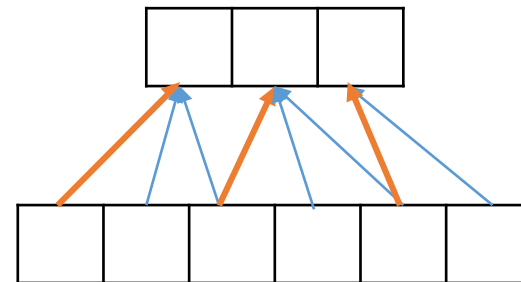
- Sparse connection



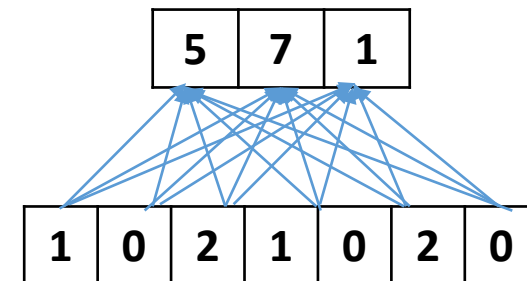
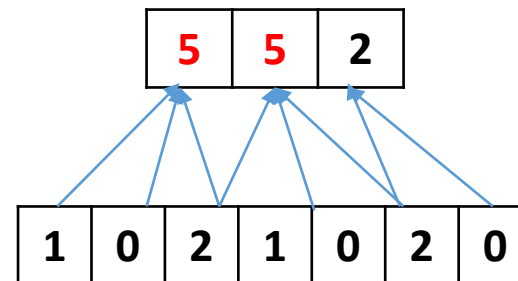
MLP (fully connected)



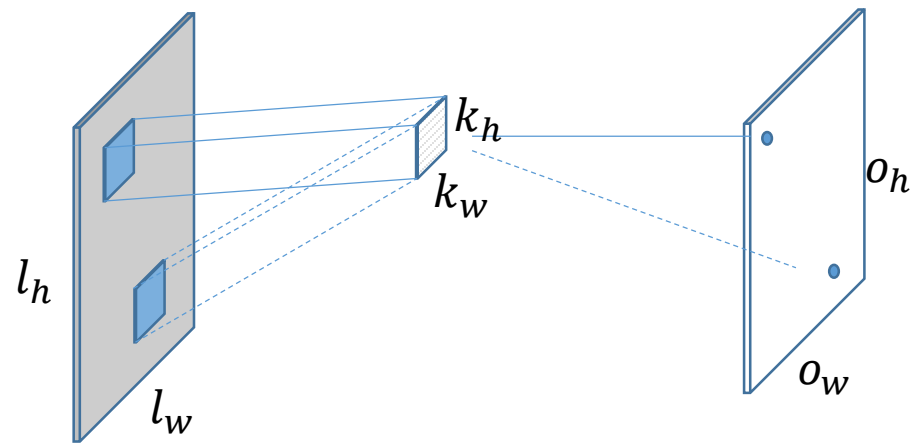
- Weight sharing



- Location invariant



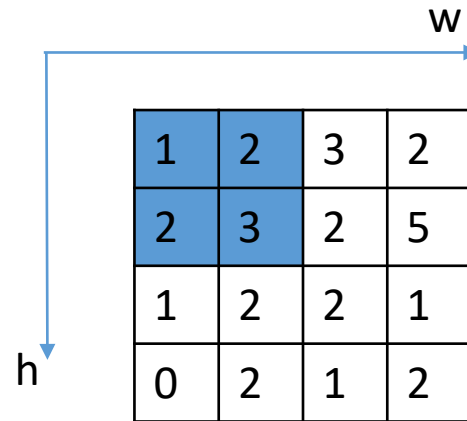
2D Convolution



2D convolution

- Input $x \in R^{l_h \times l_w}$
- Kernel $W \in R^{k_h \times k_w}$
- Output $y \in R^{o_h \times o_w}$
- $o_h \leftarrow l_h, k_h, s_h, p_h$
- $o_w \leftarrow l_w, k_w, s_w, p_w$

$$y_{i,j} = \sum_{a=0}^{k_h-1} \sum_{b=0}^{k_w-1} x_{i+a,j+b} \times W_{a,b}$$



1	2	3	2
2	3	2	5
1	2	2	1
0	2	1	2



1	2	3	2
2	3	2	5
1	2	2	1
0	2	1	2

-1	1
-1	1



2		

2D convolution

- Input $x \in R^{l_h \times l_w}$
- Kernel $W \in R^{k_h \times k_w}$
- Output $y \in R^{o_h \times o_w}$
- $o_h \leftarrow l_h, k_h, s_h, p_h$
- $o_w \leftarrow l_w, k_w, s_w, p_w$

$$y_{i,j} = \sum_{a=0}^{k_h-1} \sum_{b=0}^{k_w-1} x_{i+a,j+b} \times W_{a,b}$$

1	2	3	2
2	3	2	5
1	2	2	1
0	2	1	2

-1	1
-1	1



2	0	



1	2	3	2
2	3	2	5
1	2	2	1
0	2	1	2

2D convolution

- Input $x \in R^{l_h \times l_w}$
- Kernel $W \in R^{k_h \times k_w}$
- Output $y \in R^{o_h \times o_w}$
- $o_h \leftarrow l_h, k_h, s_h, p_h$
- $o_w \leftarrow l_w, k_w, s_w, p_w$

$$y_{i,j} = \sum_{a=0}^{k_h-1} \sum_{b=0}^{k_w-1} x_{i+a,j+b} \times W_{a,b}$$

1	2	3	2
2	3	2	5
1	2	2	1
0	2	1	2

-1	1
-1	1



2	0	2



1	2	3	2
2	3	2	5
1	2	2	1
0	2	1	2

2D convolution

- Input $x \in R^{l_h \times l_w}$
- Kernel $W \in R^{k_h \times k_w}$
- Output $y \in R^{o_h \times o_w}$
- $o_h \leftarrow l_h, k_h, s_h, p_h$
- $o_w \leftarrow l_w, k_w, s_w, p_w$

$$y_{i,j} = \sum_{a=0}^{k_h-1} \sum_{b=0}^{k_w-1} x_{i+a,j+b} \times W_{a,b}$$

1	2	3	2
2	3	2	5
1	2	2	1
0	2	1	2

-1	1
-1	1



2	0	2
2		



1	2	3	2
2	3	2	5
1	2	2	1
0	2	1	2

2D convolution

- Input $x \in R^{l_h \times l_w}$
- Kernel $W \in R^{k_h \times k_w}$
- Output $y \in R^{o_h \times o_w}$
- $o_h \leftarrow l_h, k_h, s_h, p_h$
- $o_w \leftarrow l_w, k_w, s_w, p_w$

$$y_{i,j} = \sum_{a=0}^{k_h-1} \sum_{b=0}^{k_w-1} x_{i+a,j+b} \times W_{a,b}$$

1	2	3	2
2	3	2	5
1	2	2	1
0	2	1	2

-1	1
-1	1

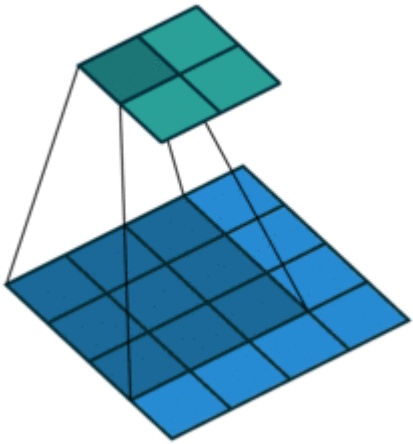


2	0	2
2	-1	2
3	-1	0

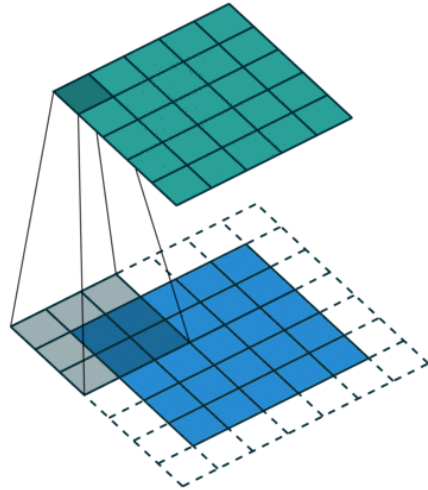


1	2	3	2
2	3	2	5
1	2	2	1
0	2	1	2

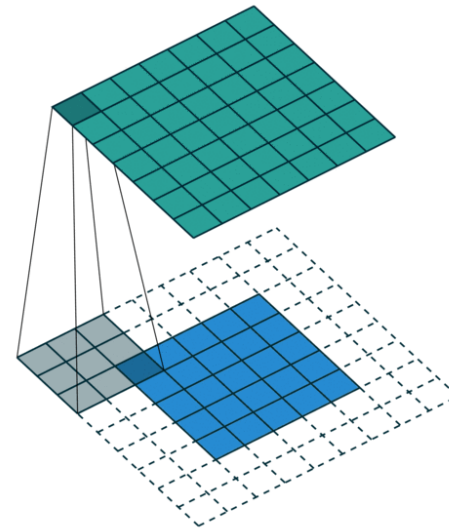
2D Convolution



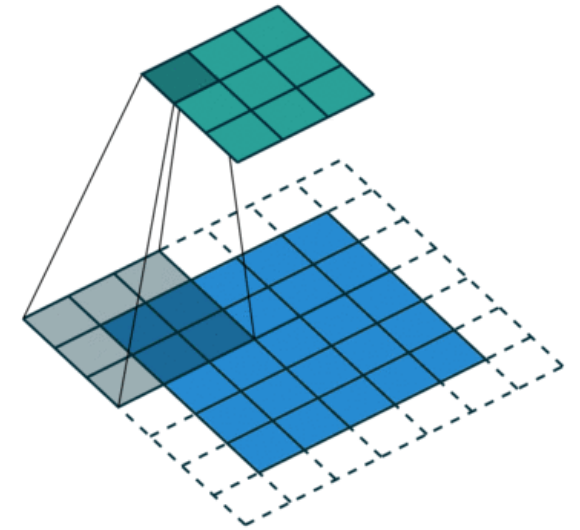
$k=3, p=0, s=1$ (Valid)



$k=3, p=2, s=1$ (Same)



$k=3, p=4, s=1$ (Full)



$k=3, p=2, s=2$

Source from [2]

2D convolution

- **Img2Col**

- Convert each receptive field into a column
- All columns construct a matrix
- Convolution = vector matrix product

- Parameter size

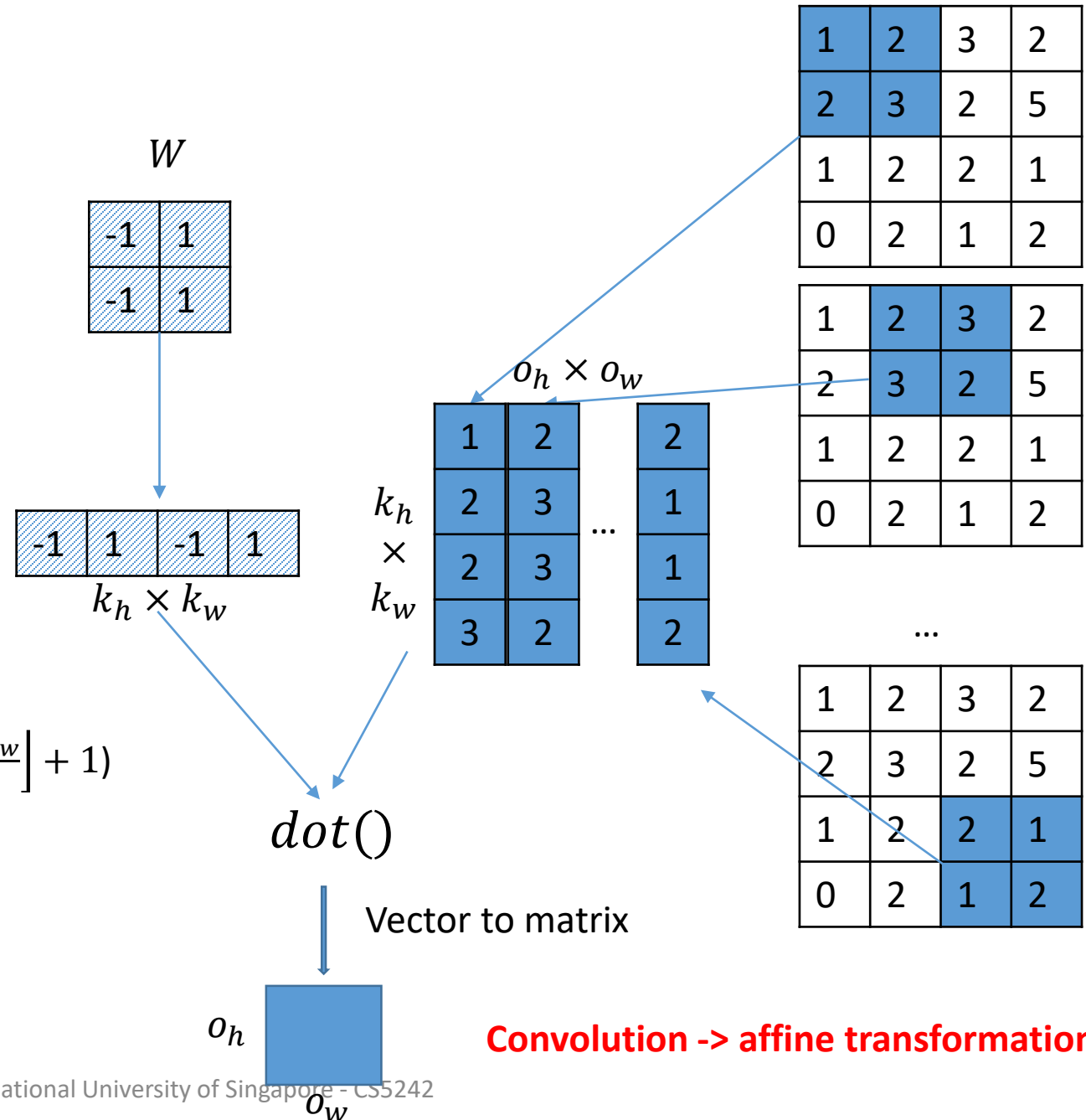
- $k_h \times k_w$

- Output shape

- $(o_h, o_w) = (\lfloor \frac{l_h + p_h - k_h}{s_h} \rfloor + 1, \lfloor \frac{l_w + p_w - k_w}{s_w} \rfloor + 1)$

- Computation cost

- $O(k_h \times k_w \times o_h \times o_w)$
(float multiplication ops, FLOP)



Convolution -> affine transformation

Question

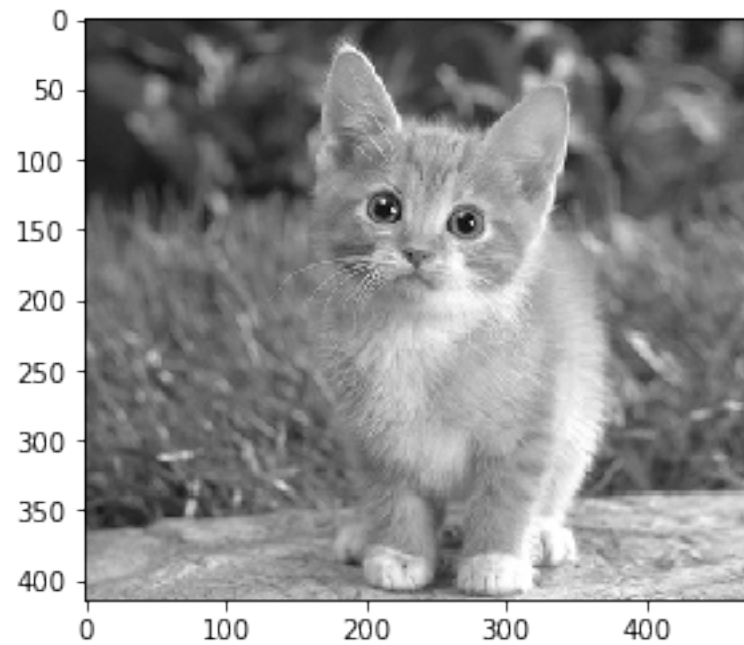
- Given input data/image of shape, kernel shape, padding size and stride size, what is the output shape?
 - qn5: Input 224x224, kernel 5x5, padding (1,1), stride (2,2)
 - qn6: Input 224x224, kernel 5x1, padding (1,1), stride (2,2)

*SMS to **76767**, e.g: qn5 xxx yyy*

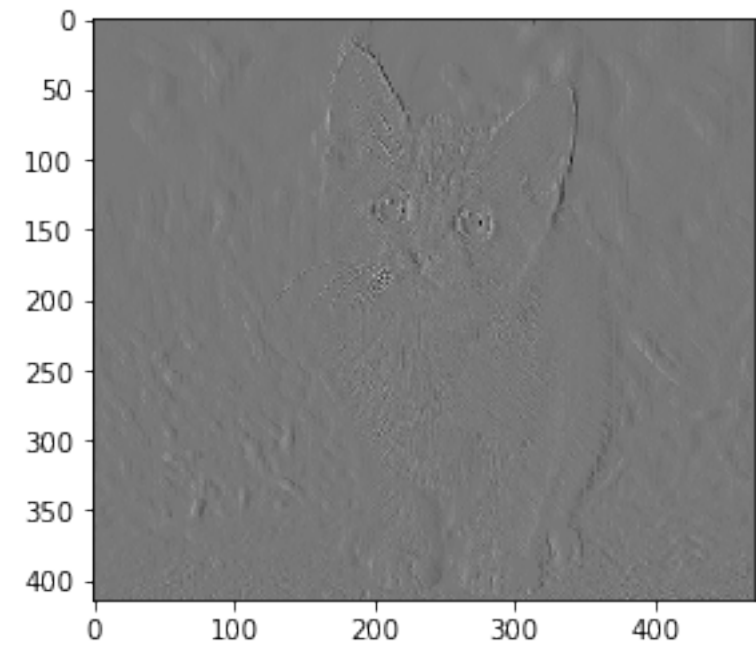
Question

- Given input data/image of shape, kernel shape, padding size and stride size, what is the output shape?
 - qn5: Input 224x224, kernel 5x5, padding (1,1), stride (2,2), output 111x111
 - qn6: Input 224x224, kernel 5x1, padding (1,1), stride (2,2), output 111x113

2D convolution

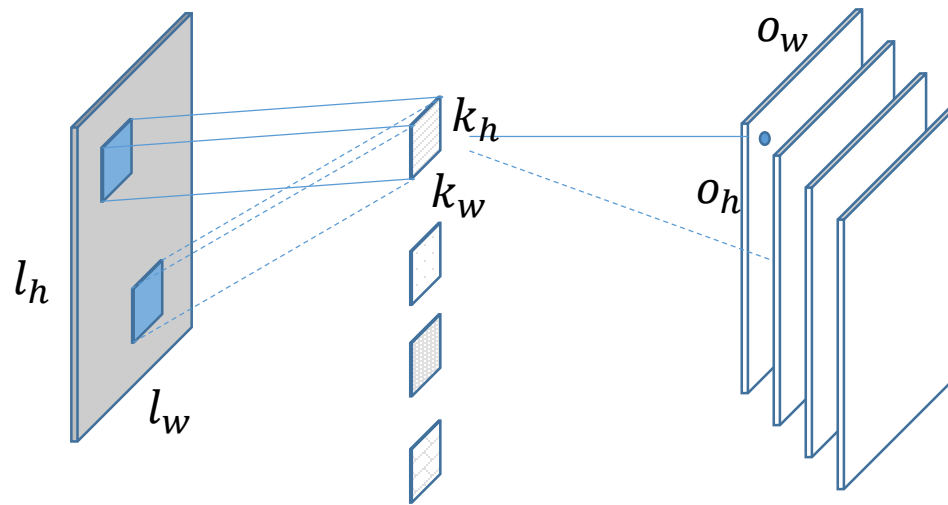


-1	1
-1	1

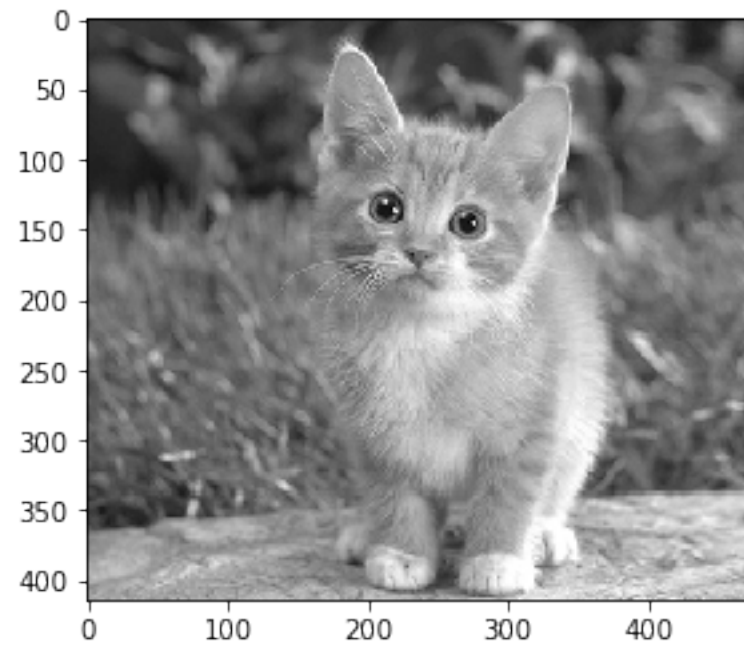


2D Convolution

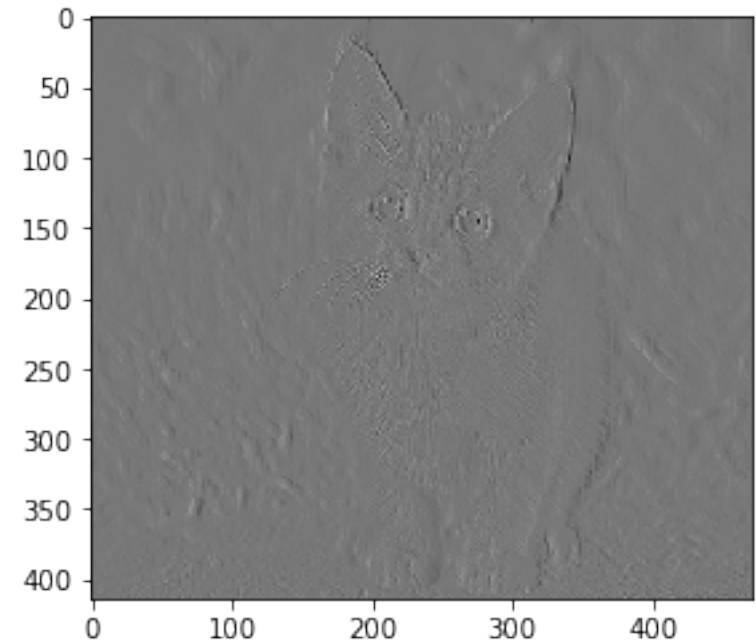
- Multiple kernels (what is a filter -> feature engineering)



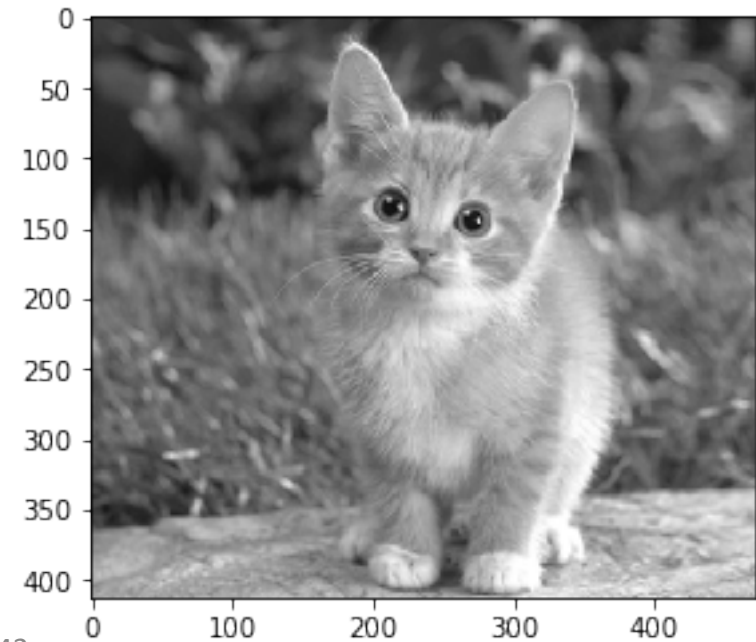
2D convolution



-1	1
-1	1

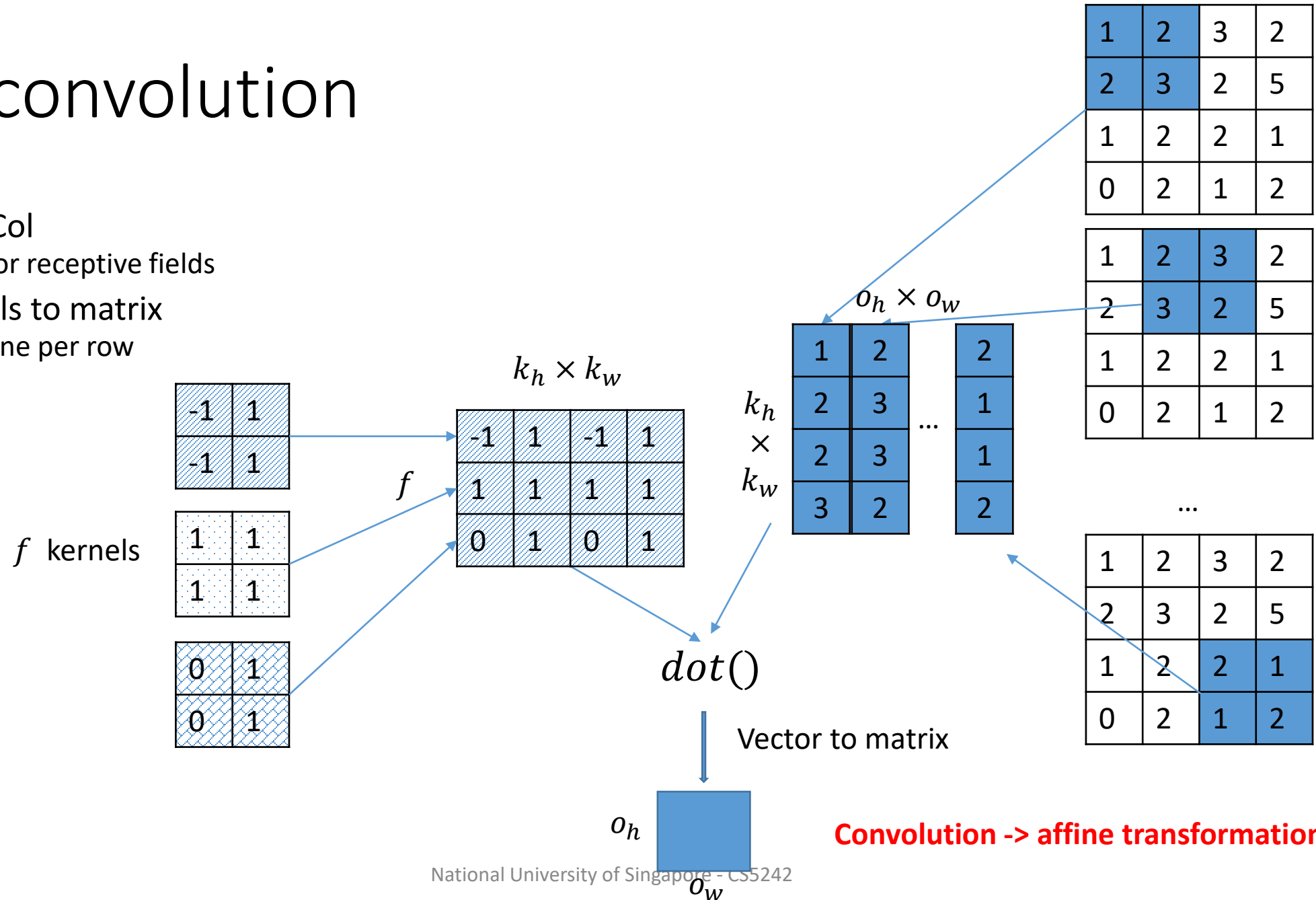


1	1
1	1



2D convolution

- Img2Col
 - For receptive fields
- Kernels to matrix
 - One per row

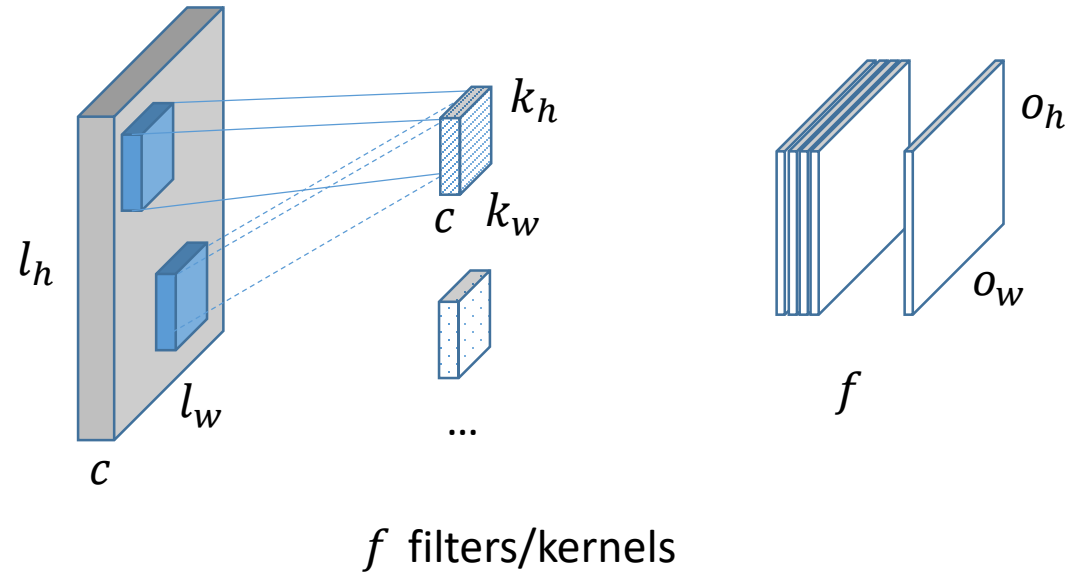


2D convolution

- With multiple kernels (filters)
- Parameter size
 - $f \times k_h \times k_w$
- Output shape
 - $(f, o_h, o_w) = (f, \left\lfloor \frac{l_h + p_h - k_h}{s_h} \right\rfloor + 1, \left\lfloor \frac{l_w + p_w - k_w}{s_w} \right\rfloor + 1)$
- Computation cost
 - $O(f \times k_h \times k_w \times o_h \times o_w)$ (float multiplication ops, FLOP)

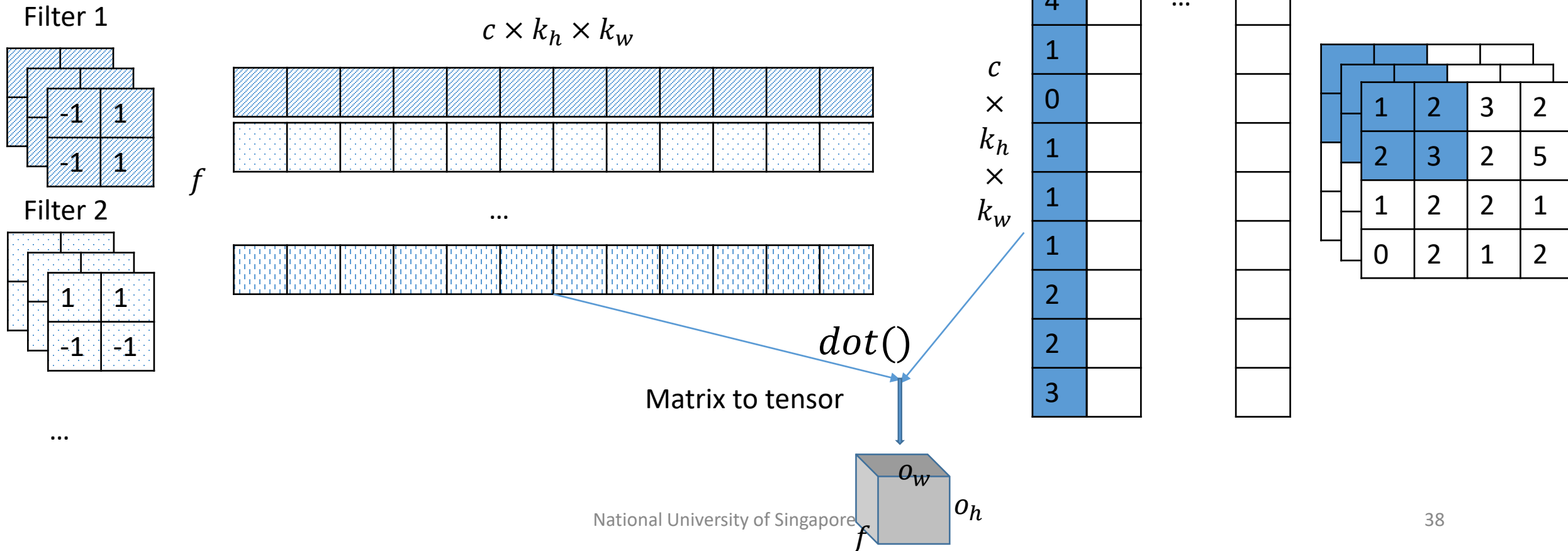
2D Convolution

- With multiple input channels and kernels (filters)



2D convolution

- With multiple channels and multiple kernels (filters)



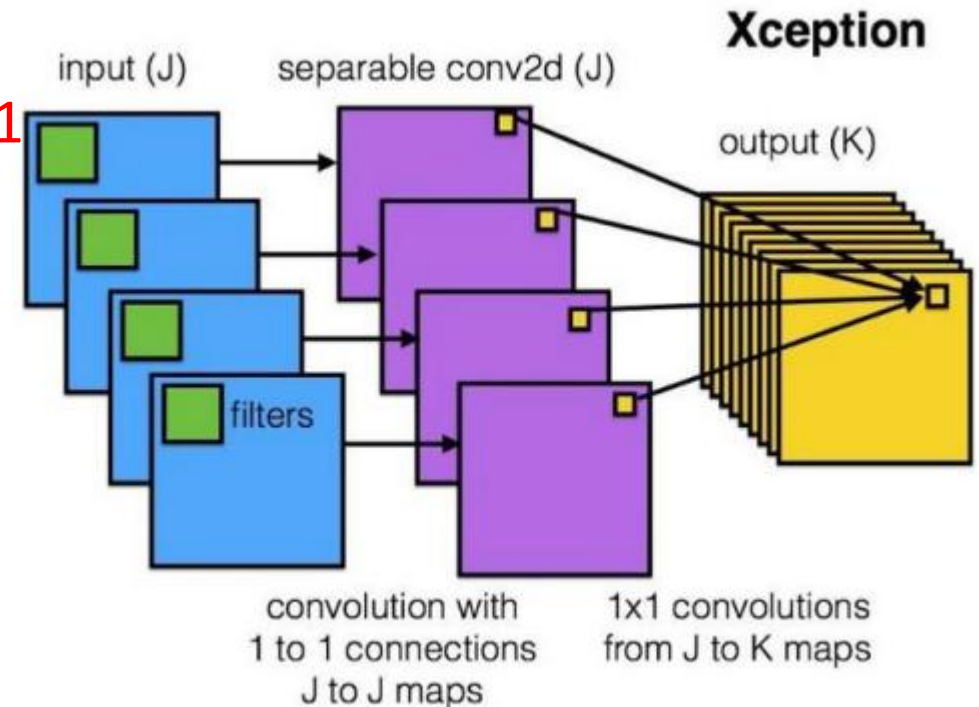
2D convolution

- With multiple channels and kernels (filters)
- Parameter size
 - $f \times (c \times k_h \times k_w)$
- Output shape
 - $(f, o_h, o_w) = (\left\lfloor \frac{l_h + p_h - k_h}{s_h} \right\rfloor + 1, \left\lfloor \frac{l_w + p_w - k_w}{s_w} \right\rfloor + 1)$
- Computation cost
 - $O(f \times c \times k_h \times k_w \times o_h \times o_w)$ (float multiplication ops)

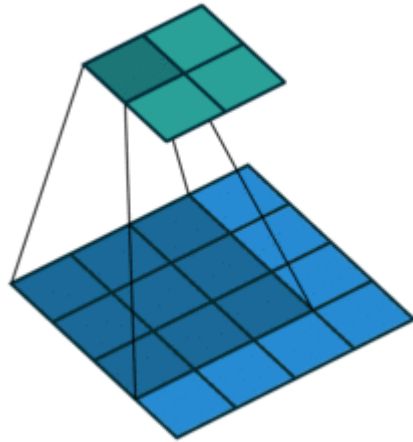
Other convolution operations

Depth-wise Separable Convolution

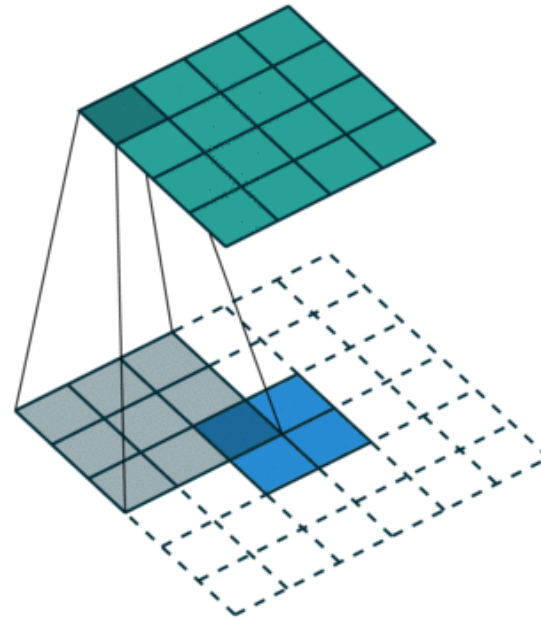
- Depth-wise convolution
 - # input channels (feature mas) = # output channels (feature maps)
 - Parameter size: $c \times k_h \times k_w$
 - Output shape: $c \times o_h \times o_w$
 - Different with normal convolution with $f=1$
 - Output shape $1 \times o_h \times o_w$
 - Computation cost:
 - $O(c \times k_h \times k_w \times o_h \times o_w)$
- Point-wise convolution
 - Normal convolution with 1×1 kernel



Transposed Convolution



Convolution [2]

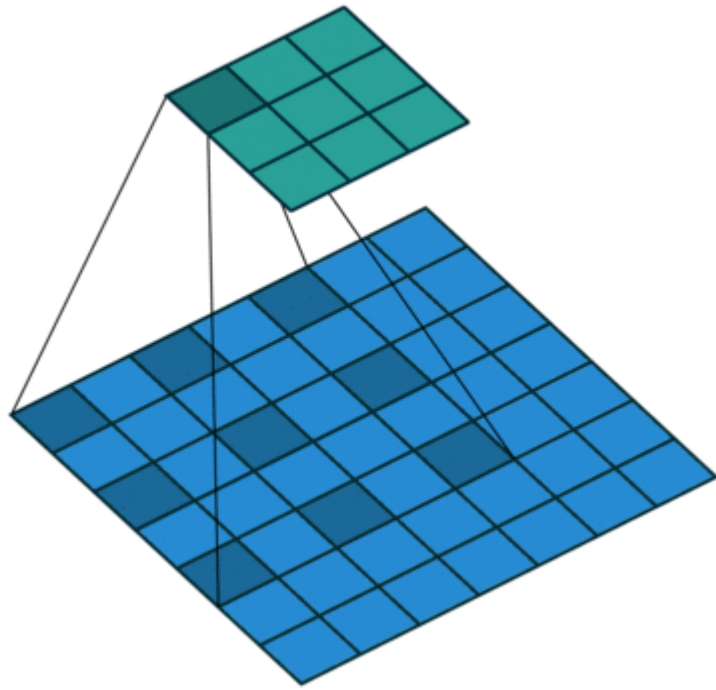


Transposed convolution [2]

Transposed Convolution

- Application
 - Image generation (GAN)
 - Visualization
 - Segmentation [4]
- Implementation
 - Normal convolution, $Y = W * X$
 - Transposed convolution $Y = W^T * X$

Dilated Convolution



Source from [2]

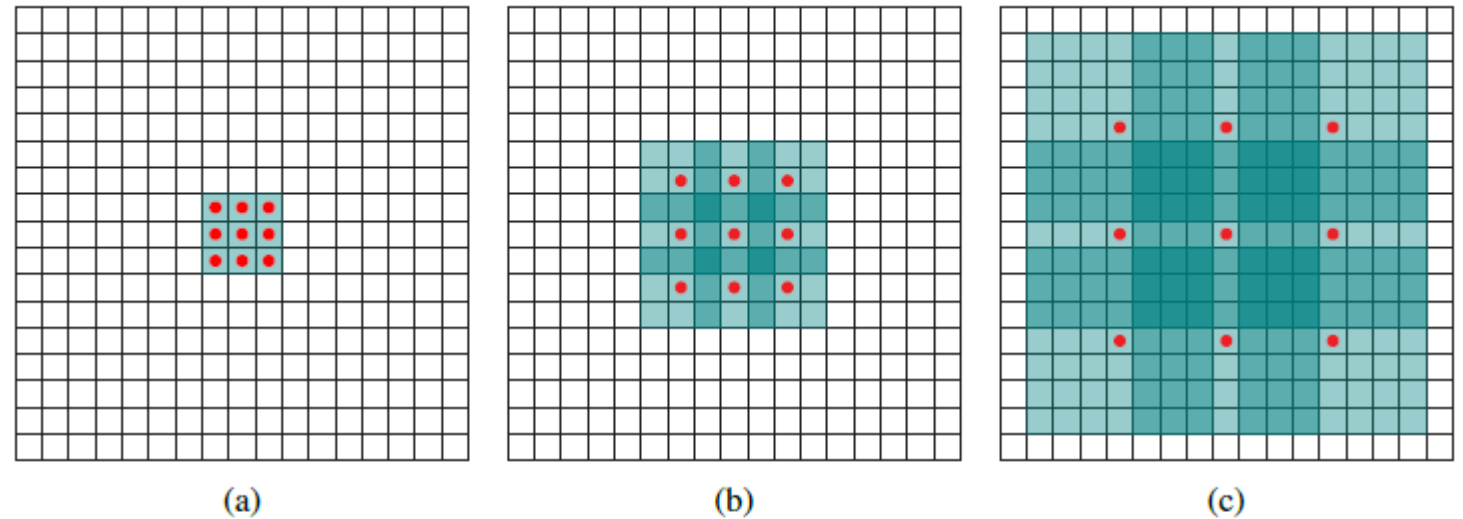


Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3 . (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7 . (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15 . The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

Source from [4]

Dilated convolution

- Increase the receptive field of neurons in top layers fast
- Less parameters compared with normal convolution
- Dense prediction
 - vision applications where the predicted object that has similar size and structure to the input image
 - semantic segmentation with one label per pixel

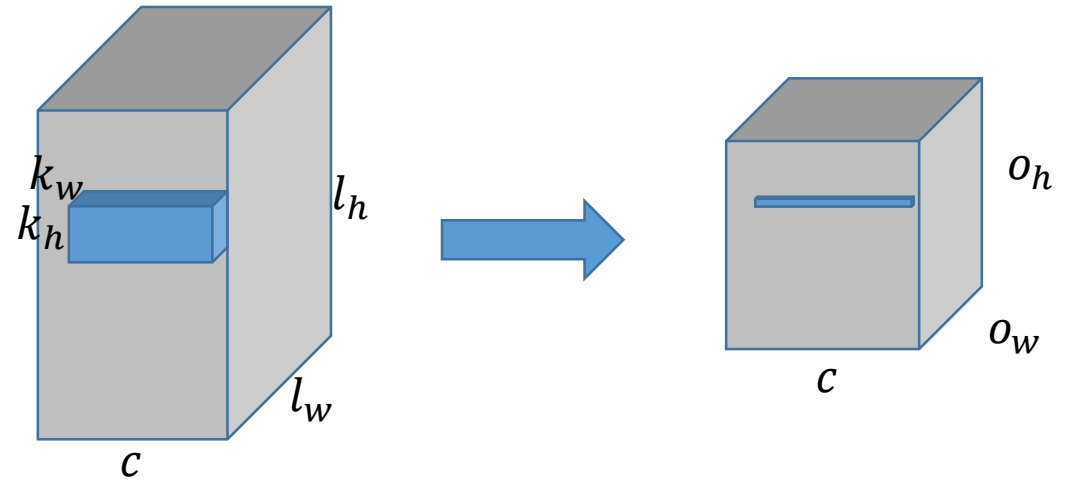
3D convolution

- 2D convolution
 - Input shape (c, h, w)
 - Convolution/slide over spatial dimensions, i.e. height and width
- 3D convolution
 - Input shape (c, h, w, **d**)
 - Convolution/slide over h, w, and d.

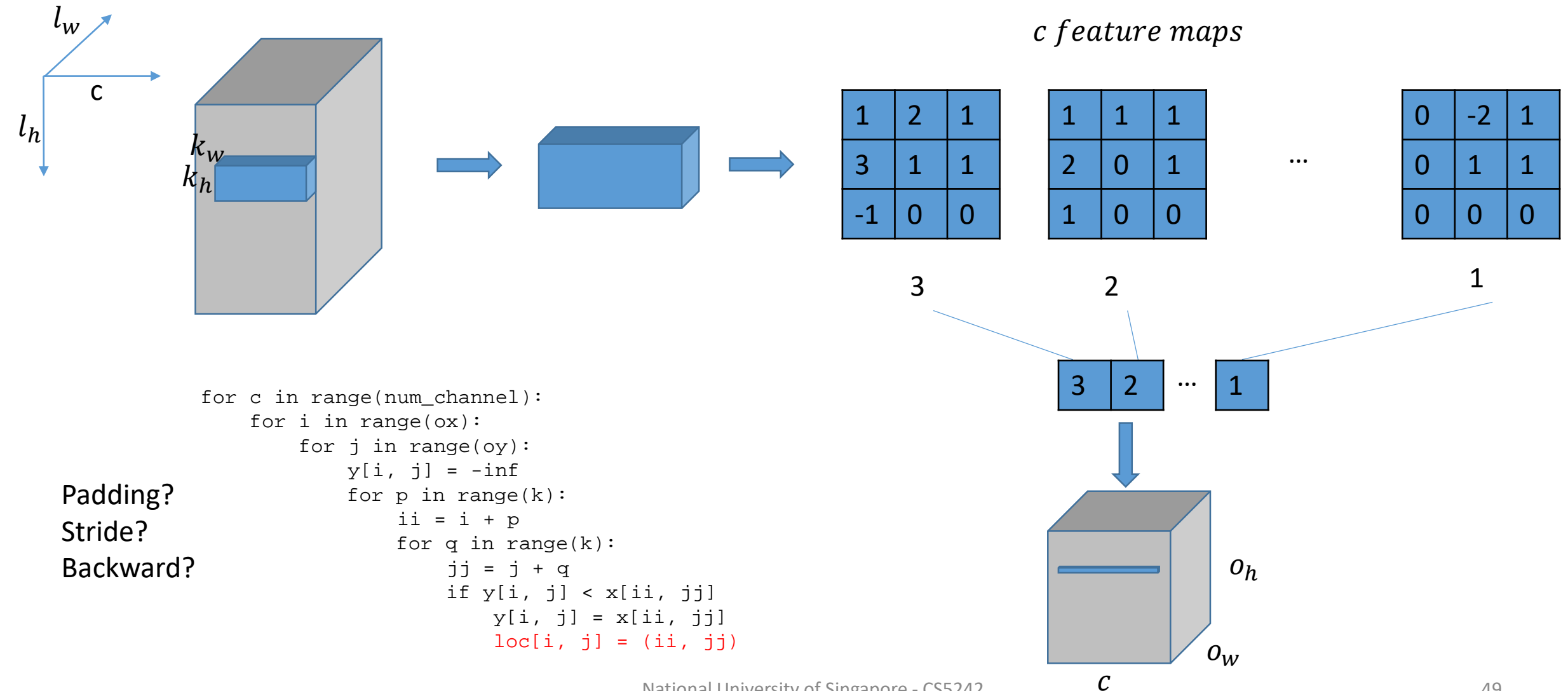
Pooling

Pooling

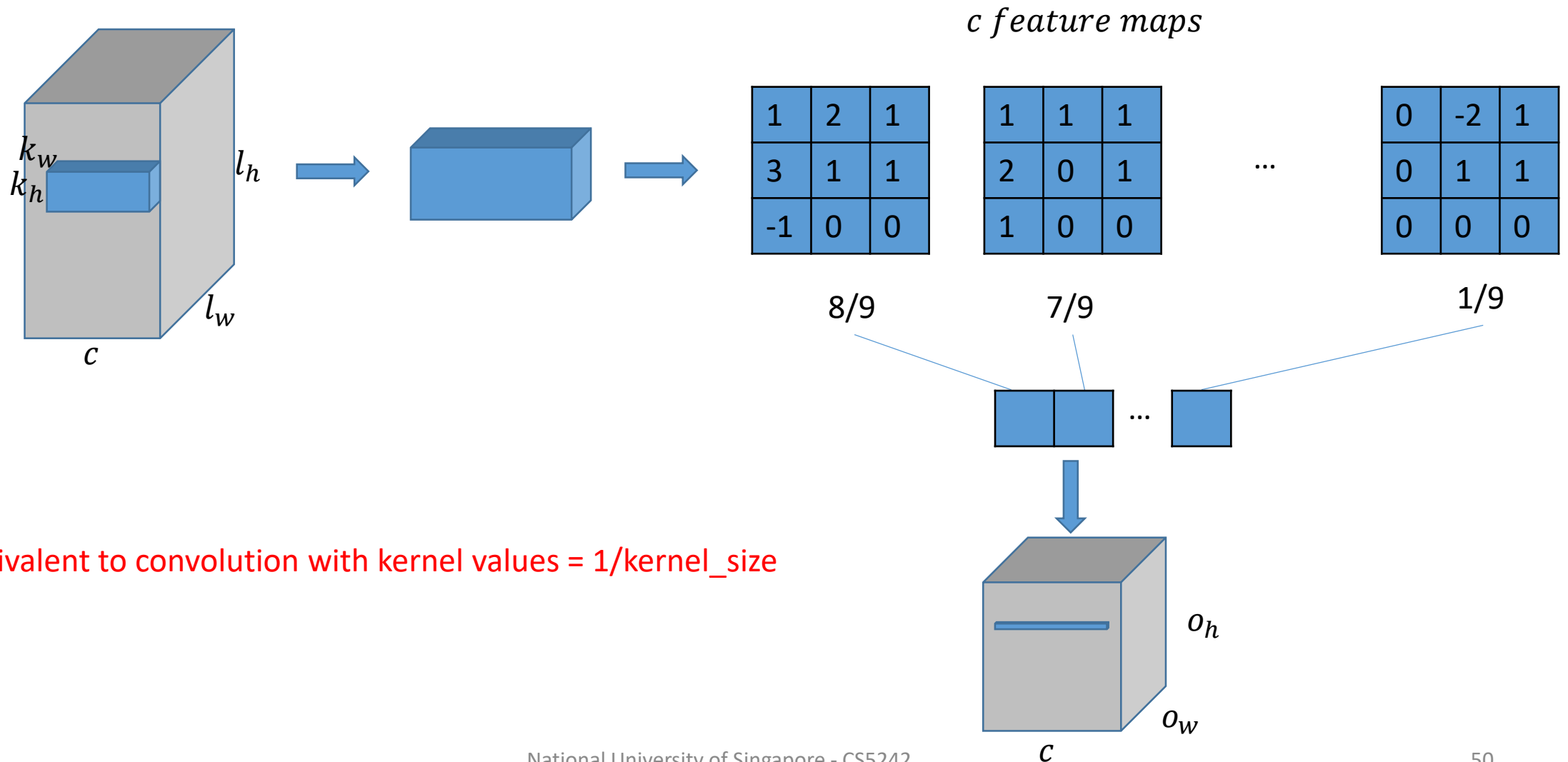
- Input shape
 - (c, l_h, l_w)
- Output shape
 - $(c, o_h, o_w) = (c, \left\lfloor \frac{l_h + p_h - k_h}{s_h} \right\rfloor + 1, \left\lfloor \frac{l_w + p_w - k_w}{s_w} \right\rfloor + 1)$
- Why do pooling?



Max Pooling



Average Pooling



Equivalent to convolution with kernel values = $1/\text{kernel_size}$

References and additional readings

- [1] LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-based learning applied to document recognition" (PDF). Proceedings of the IEEE. 86 (11): 2278–2324. doi:10.1109/5.726791. Retrieved October 7, 2016.
- [2] http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html
- [3] https://zhuanlan.zhihu.com/p/28749411?utm_medium=social&utm_source=wechat_session
- [4] <https://arxiv.org/pdf/1511.07122.pdf>
- [5] <https://arxiv.org/pdf/1412.7062.pdf>
- <http://cs224d.stanford.edu/>
- <http://cs231n.stanford.edu/>
- Goodfellow Ian, Bengio Yoshua, Courville Aaron. Deep learning. MIT Press. <http://www.deeplearningbook.org>. Chapter 9.
- https://www.tensorflow.org/api_guides/python/nn#Notes_on_SAME_Convolution_Padding
- <http://timdettmers.com/2015/03/26/convolution-deep-learning/>
- https://www.tensorflow.org/api_docs/python/tf/nn/separable_conv2d
- http://link.zhihu.com/?target=http%3A//www.di.ens.fr/data/publications/papers/phd_sifre.pdf