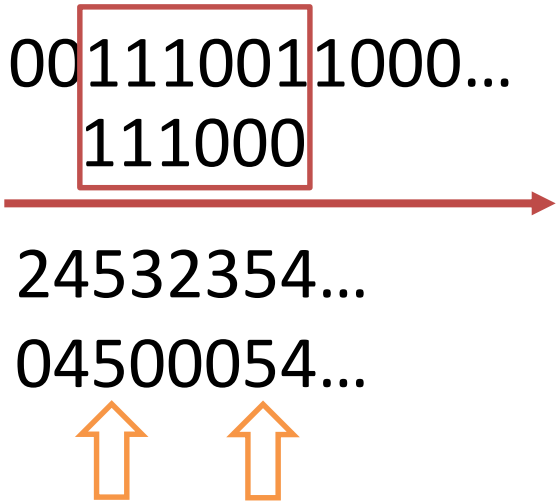
Lecture 7:
Convolutional Neural Network

Topics of this lecture

- A brief review of pattern detection
- Convolutional neural network
  - Basic structure
  - Meaning of a convolutional layer
  - Meaning of a pooling layer
  - Meaning of a full connected layer
  - Other issues related to CNN
  - Learning of CNN
- Example of CNN-based image recognition

A brief review of pattern detection (1/5)

- Given a data sequence, say x="001110011000…", we want to detect if this sequence contains the pattern w="111000", we can compare x and w bit-by-bit (see below).
- That is, we can shift w from left to right, and match x with w. If the similarity is larger than a threshold, we can say that a pattern "similar to w" exists in x.



- Similarity=number of identical bits
- Can be normalized by dividing the length of w
- Can use a threshold (say, 3) to filter out non-similar locations

A brief review of pattern detection (2/5)

- Generally speaking, we may have
  - A long sequence $x(n), n \in [0, N-1]$; and
  - A relatively short sequence $w(n), n \in [0, M-1]$ $(M < N)$.
- We want to know if $w(n)$ exists in $x(n)$.
- We can define the similarly based on the **cross-correlation** given as follows:

$$r(n) = x(n) \otimes w(n)$$
$$= \sum_{m=0}^{M-1} x(n+m)w(m)\,, n = 0, \dots, N-1 \qquad (1)$$

- If $r(\tau) > \theta$, we can say that a pattern similar to $w(n)$ exists in $x(n)$, and the position is τ. Here, θ is a given threshold.

A brief review of pattern detection (3/5)

- In digital signal processing, a *linear, time-invariant* digital filter is uniquely defined by its *impulse response* $w(n), n = 0, \dots, M-1$.
- Given $w(n)$, the output $y(n)$ for any input $x(n)$ is given by

$$y(n) = x(n) \circledast w(n)$$
$$= \sum_{m=0}^{M-1} x(n-m)w(m), n \geq 0 \qquad (2)$$

- This equation is called the *convolution sum* of $x(n)$ and $w(n)$.
- Eq. (1) and Eq. (2) are very similar. In fact, if we define $\widehat{w}(n) = w(M-n-1), n = 0, \dots, M-1$, we can use Eq. (2) to detect the pattern. If w is symmetric, convolution and cross correlation are the same, but in general they are different.
- Both convolution sum and cross correlation can be found efficiently using FFT.

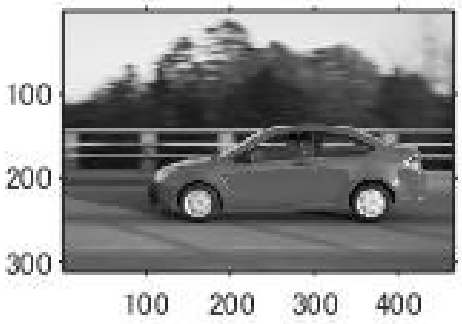
A brief review of pattern detection (4/5)

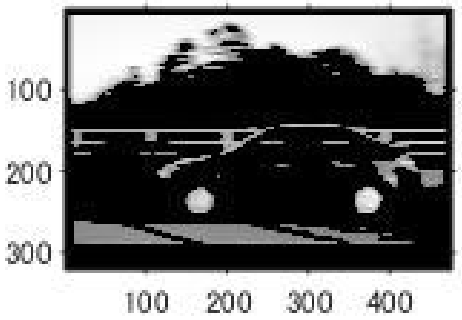
- Similarly, if we have
  - A large image $x(i,j), i,j = 0, \dots, N-1$; and
  - A relatively small image $w(i,j), i,j = 0, \dots, M-1$ ($M < N$).
- We want to know if $w(m,n)$ exists in $x(i,j)$.
- Again, we can define the similarly as follows:

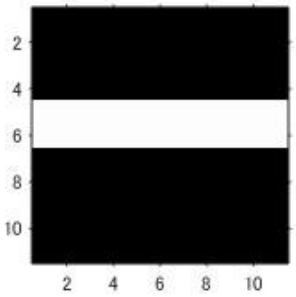
$$r(i,j) = x(i,j) \otimes w(i,j)$$
$$= \sum_{n=0}^{M-1} \sum_{m=0}^{M-1} x(i+m, j+n) w(m,n), \quad i,j = 0,1, \dots, N-1 \qquad (3)$$

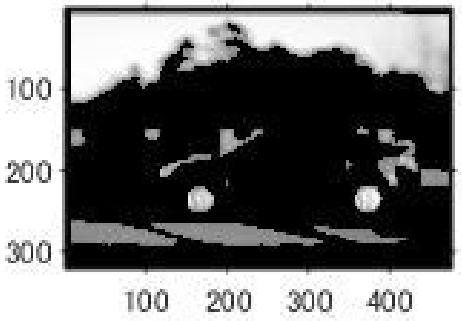
- Note that the most important thing for detecting a given pattern is to find the locations where $r(i,j)$ takes large values.

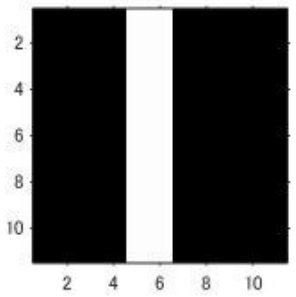
A brief review of pattern detection (5/5)









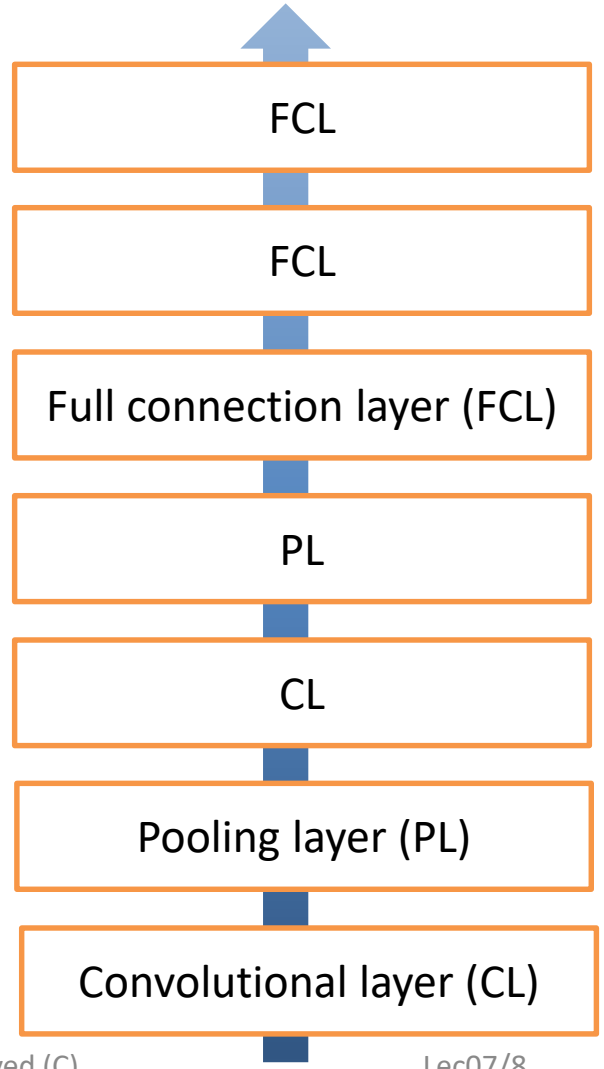


Using different patterns given on the right, we can "match" them with the original image, and see the similarity on each position.

- We can see that the guardrail is more similar to the first pattern because the cross correlation has larger values (brighter).
- The "similarity" has been normalized by the maximum value and then cut by a threshold T=0.5.

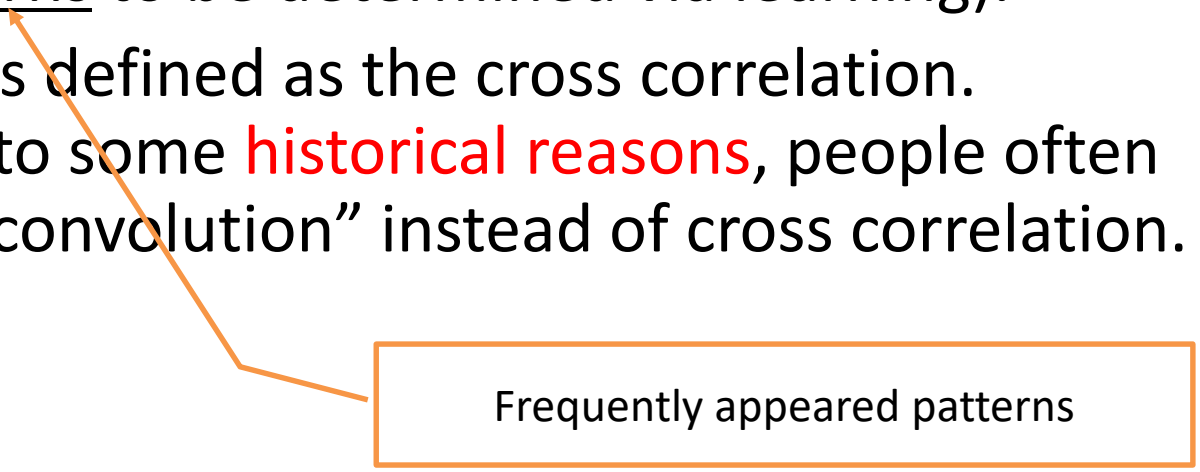
Convolutional neural networks - Basic structure

- Convolutional neural network (CNN) is a special MLP.
- The basic structure of the network is the same as a normal MLP. There is one input layer, one output layer, and many hidden layers.
- The hidden layers can be classified roughly into three categories:
  - **Convolutional layer (CL)**
  - **Pooling layer (PL)**
  - **Fully connected layer (FCL)**
- Usually, CL is used with PL in pair; and several FLs are used together to find the final outputs.
- The right figure is a typical example.

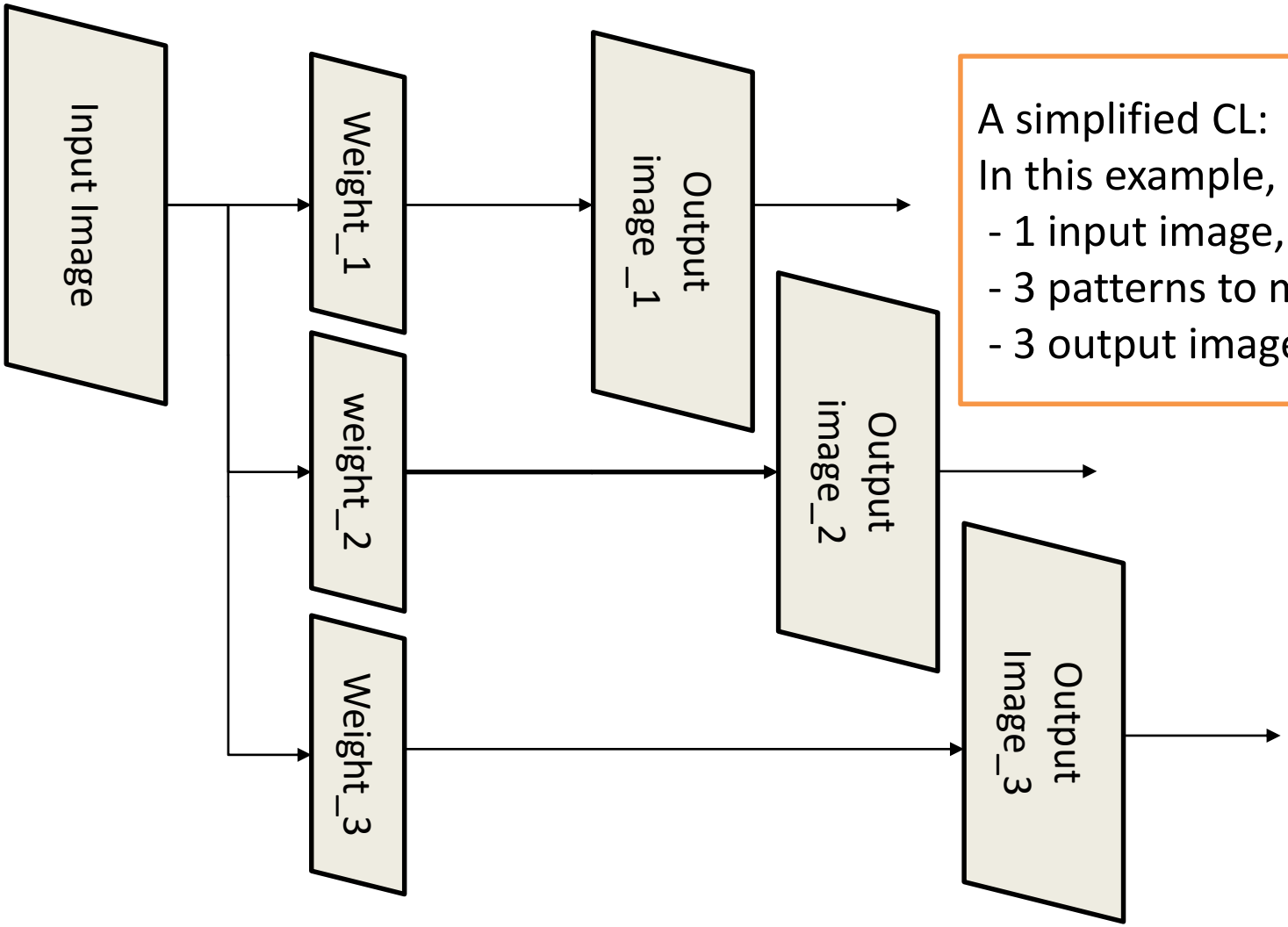


CNN: Meaning of a CL (1/6)

- Since CNN was first developed for image recognition, we usually consider the input of a CNN an "multi-channel image" or "multi-channel map".
- The basic operation of a CL is to find the similarity between a large input image and several small "weight images" (patterns to be determined via learning).
- The similarity is defined as the cross correlation. However, due to some historical reasons, people often use the term "convolution" instead of cross correlation.



CNN: Meaning of a CL (2/6)



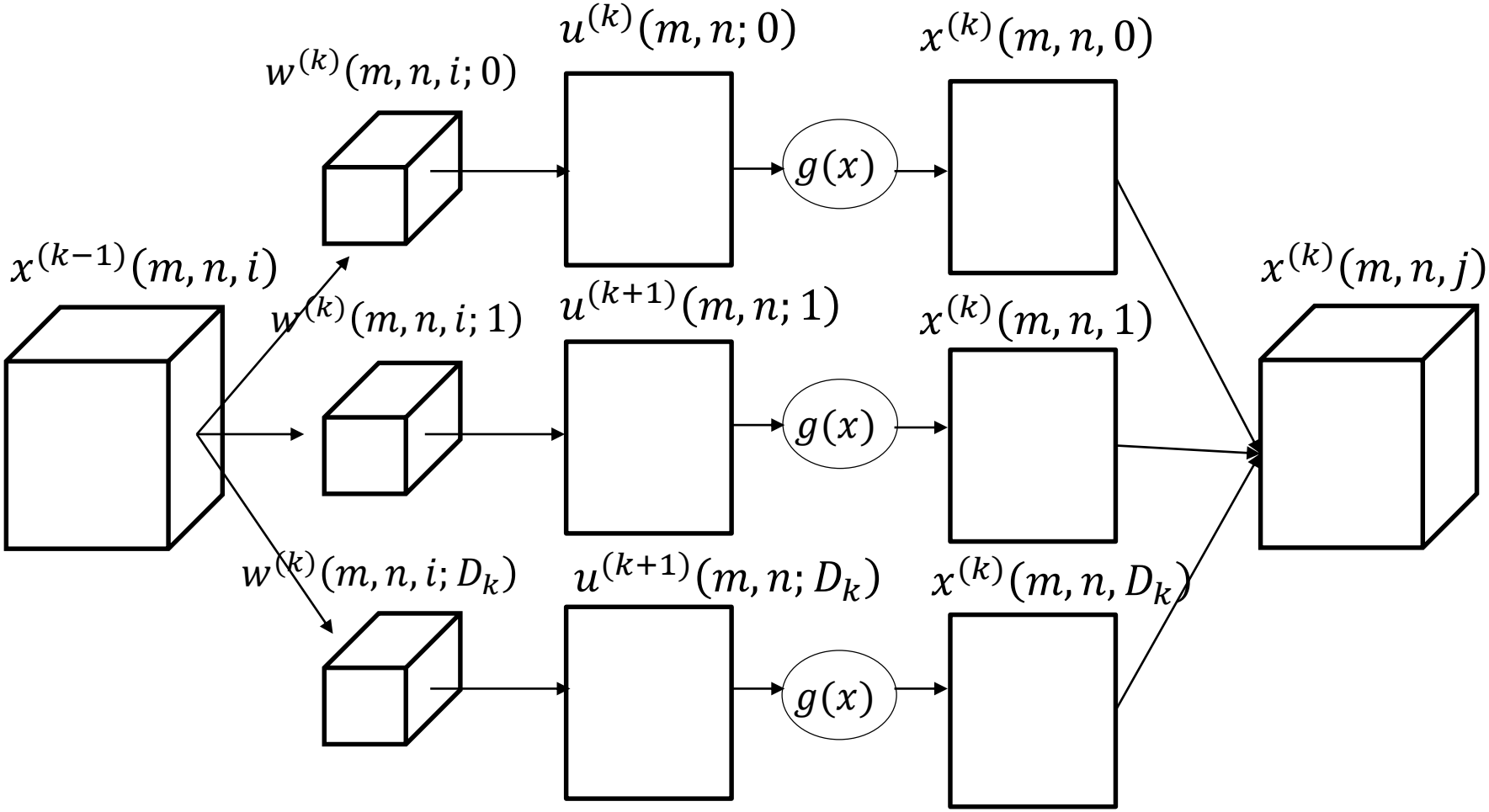
A simplified CL:
In this example, there are:
- 1 input image,
- 3 patterns to match, and
- 3 output images (channels).

CNN: Meaning of a CL (3/6)

- The sizes of the input and output
  - The original input data may contain data (images or maps) captured from D_0 "channels".
    - For a color image, $D_0 = 3$.
    - A satellite data may contain images from several channels (see https://www.weather.gov/satellite#geocolorGOESe).
  - For the k-th ($k \geq 1$) convolutional layer, we obtain D_k output images if we try to detect D_k patterns.
  - The size of each pattern is $M_k \times M_k \times D_{k-1}$, where $M_k \times M_k$ is the size of the 2-D image pattern to detect, and D_{k-1} is the number of channels of the $(k-1)$-th layer.

May not be a square in practice!

CNN: Meaning of a CL (4/6)



CNN: Meaning of a CL (5/6)

- Summary of notations (for the k-th layer):
  - $x^{(k-1)}(m,n,i)$: input image (map) of the i-th channel
    - $m,n=0,\dots,M_{k-1}; i=0,\dots,D_{k-1};$
  - $w^{(k)}(m,n,i;j)$: j-th pattern to match the input data.
    - $m,n=0,\dots,M_k; i=0,\dots,D_{k-1}; j=0,\dots,D_k$
  - $u^{(k)}(m,n;j)$: effective input of the (m,n) −th neuron in the j-th channel of the k-th layer.
    - $m,n=0,\dots,M_k;\ j=0,\dots,D_k$
  - $x^{(k)}(m,n;j)$: image (map) of the j-th channel for the k-th layer.
    - $m,n=0,\dots,M_k;\ j=0,\dots,D_k$
  - $g()$: The activation function used by the k-th layer. For convolutional layers, we usually use ReLU instead of using sigmoid functions.

CNN: Meaning of a CL (6/6)

- The output of the k-th CL is found by

$$x^{(k)}(m,n;j) = g\left(u^{(k)}(m,n;j)\right), j = 0,1,\dots,D_k \quad (4)$$

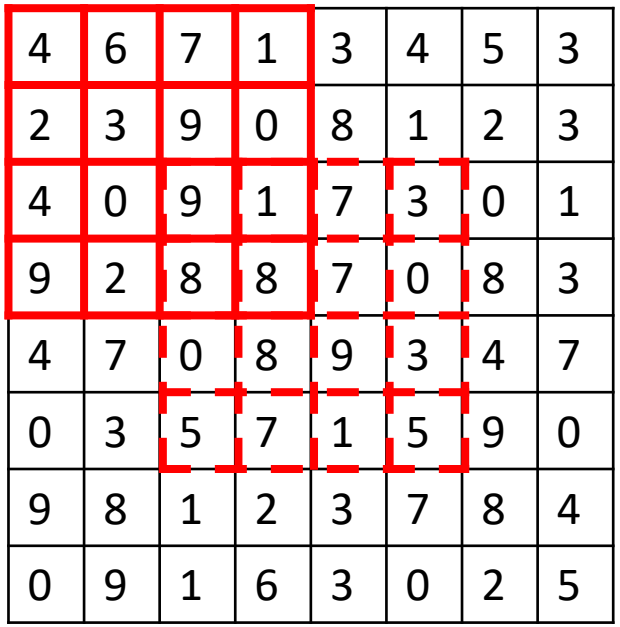
$$u^{(k)}(m,n;j) = \sum_{i=0}^{D_{k-1}} \sum_{p=0}^{M_k-1} \sum_{q=0}^{M_{k-1}} x^{(k-1)}(m+p,n+q,i) w^{(k)}(p,q,i,j) + \theta_{mnj} \quad (5)$$

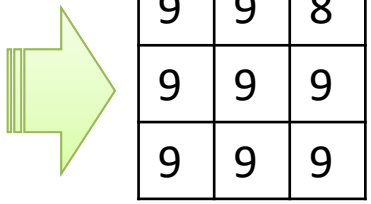
- Note that for each pattern to match, we actually have D_{k-1} input images (2-D maps), one for each channel. The results are added together in Eq. (5).
- The number of channels of the next layer equals to the number of patterns used to match in the input data.
- The threshold can be the same for all neurons in all channels.

CNN: Meaning of a PL (1/3)

- The main purposes of a pooling layer are
  - To reduce the computational cost;
  - To reduce the sensitivity of the network to various changes (shift, rotation, noise, etc.);
  - To make it easier for the next layer to detect meaningful patterns.
- The basic operation in the pooling layer is to replace a block of data (e.g. a 4x4 segment of the input image) with one value. This value can be
  - The maximum value contained in the block: max pooling;
  - The average value of all values in the block: average pooling.
- Usually, what we want is to detect some meaningful patterns. In this sense, max pooling should be used.

CNN: Meaning of a PL (2/3)





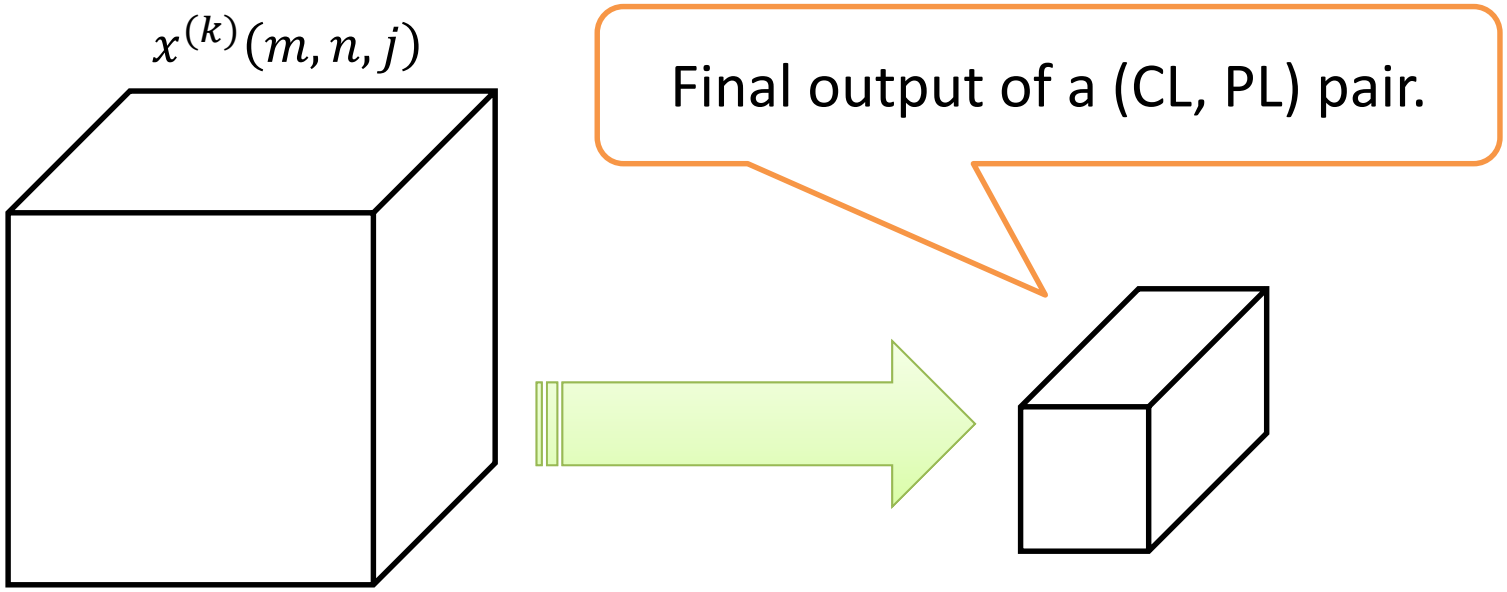
The given pattern has been detected ALMOST everywhere!

Eq. (6): Max pooling
Eq. (7): Average pooling

$$\hat{x}^{(k)}(m,n,j) = \max_{(m,n)\in Pool} x^{(k)}(m,n,j) \qquad (6)$$

$$\hat{x}^{(k)}(m,n,j) = \frac{1}{Pooling\_size}\sum_{(m,n)\in Pool} x^{(k)}(m,n,j) \qquad (7)$$

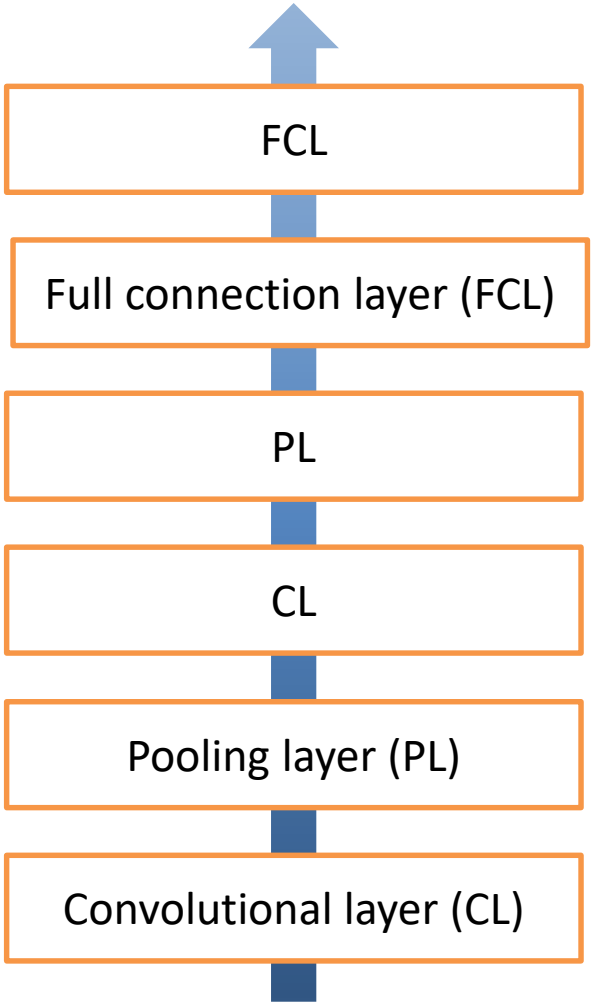
CNN: Meaning of a PL (3/3)



- The pooling size: In the previous example, the size is (4, 4). A larger size can make the system less sensitive to various changes (may not be good to detect the position accurately).
- The stride: In the above example, the stride is (2, 2). A larger stride can produce a smaller output image.

CNN: meaning of an FCL

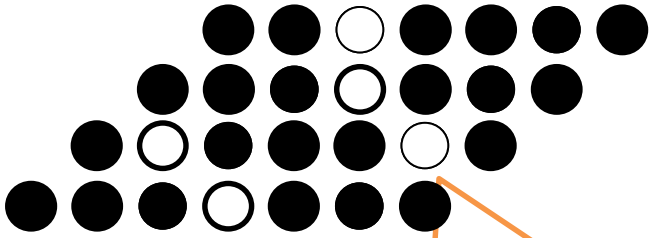
- Calculation of the fully connected layers is the same as we have learned for general MLPs.
- This part functions as an integrator that summarizes all information obtained so far and makes the final decision.
- Given the information extracted by the lower layers, the last FCL usually can make good decisions even if they are relatively "weak" (e.g. linear).



CNN: other issues (1/2)

- Stride can also be used in the convolutional layers to reduce the computational cost.
- Zero-padding can be used to extend the "border" of an image to detect patterns close to the borders.
- Dropout is often used to reduce the chance of over-fitting. That is, some (randomly selected) neurons can be discarded (ignored) in each learning cycle to effectively simplify the structure of the network.

Proper zero-padding can make it easier to detect patterns close to the border.



Dropout can reduce the number of parameters for training, and reduce over-fitting.

CNN: other issues (2/2)

- Local contrast normalization (LCN) is an operation useful for image recognition.
- This can be considered an additional layer between a pooling layer and the next convolutional layer.
- The simplest method is to remove the average as follows:

$$\hat{x}(i,j) = x(i,j) - \bar{x}, for\ i,j \in W \quad (8)$$

- where $\bar{x}$ is the average value of $x(i,j)$ in a sliding window W. We may also normalize the data by dividing them with the standard deviation.
- In any case, the activation function of an LCN layer is a differentiable function, and its first order derivative can be used for updating the weights in the CL.

Learning of CNN (1/3)

- Learning of a CNN can be conducted in the same way as an MLP, using the well-known BP algorithm (or improved versions).
  - For fully connected layers, learning is exactly the same.
  - For pooling layers and local contrast normalization layers, there is no parameters to learn (although we need to pre-specify the window size for pooling, the stride size, and the window size for normalization).
  - For convolutional layers, we can convert the cross correlation (or convolution sum) into a form of linear transform, and then conduct BP-based learning.

Learning of CNN (2/3)

- Take 1-D case as an example. For $x(n), n = 0,1, \ldots, 7$; and $w(n), n = 0,1,2$, we have

$$\begin{bmatrix} r(0) \\ r(1) \\ \vdots \\ r(7) \end{bmatrix} = \begin{bmatrix} w(0) & w(1) & w(2) & 0 & 0 & 0 & 0 & 0 \\ 0 & w(0) & w(1) & w(2) & 0 & 0 & 0 & 0 \\ 0 & 0 & w(0) & w(1) & w(2) & 0 & 0 & 0 \\ 0 & 0 & 0 & w(0) & w(1) & w(2) & 0 & 0 \\ 0 & 0 & 0 & 0 & w(0) & w(1) & w(2) & 0 \\ 0 & 0 & 0 & 0 & 0 & w(0) & w(1) & w(2) \\ 0 & 0 & 0 & 0 & 0 & 0 & w(0) & w(1) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w(0) \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(7) \end{bmatrix}$$

OR $R = WX$

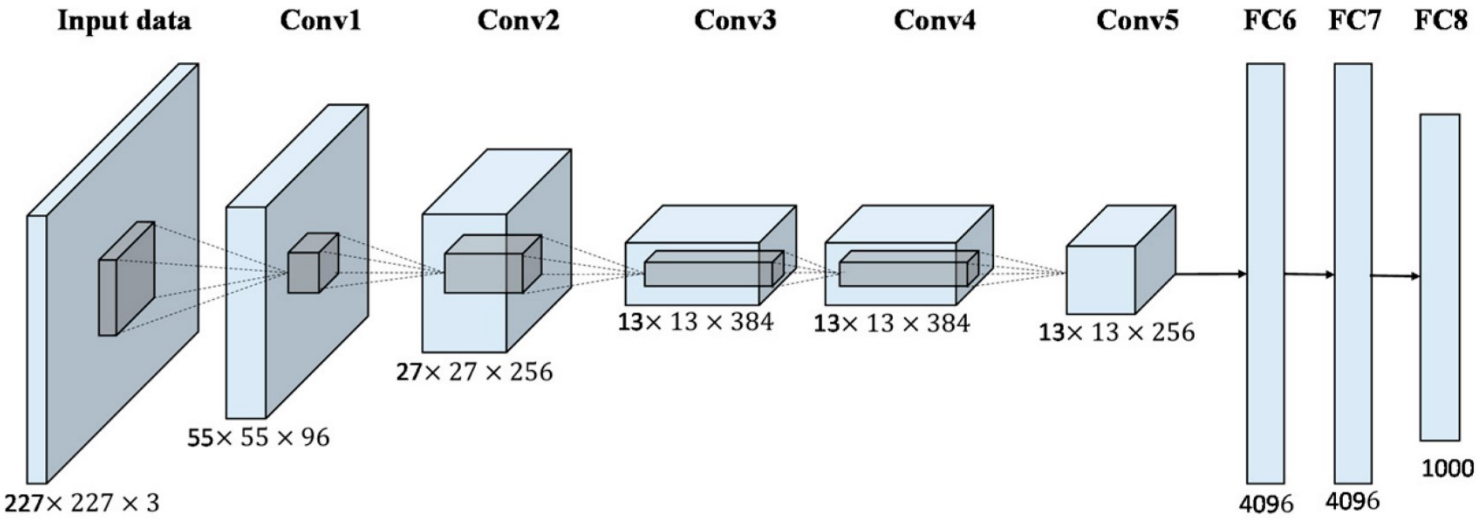
- Thus, although the weight matrix is very sparse, learning of a CL is the same as that of an FL.

Learning of CNN (3/3)

- However, if we use BP as is, based on the equation given in the previous page, the weight matrix may become non-sparse after learning, and the weights may take many different values.
- Since a small weight matrix (the pattern to detect) is shared by all pixels (neurons) in the same channel, it is better to define each element in the big weight matrix as a “pointer” (of a certain weight).
- During learning, we should update the weights based on these pointers. By so doing, we can keep the physical meanings of the weights unchanged.
- Zero elements in the big matrix will not be updated at all.

CNN for image recognition (1/4)

- Let us take AlexNet as an example.



AlexNet is a convolutional neural network which competed in the ImageNet Large Scale Visual Recognition Challenge in 2012. AlexNet was designed by the SuperVision group, consisting of Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever (*from Wikipedia*).

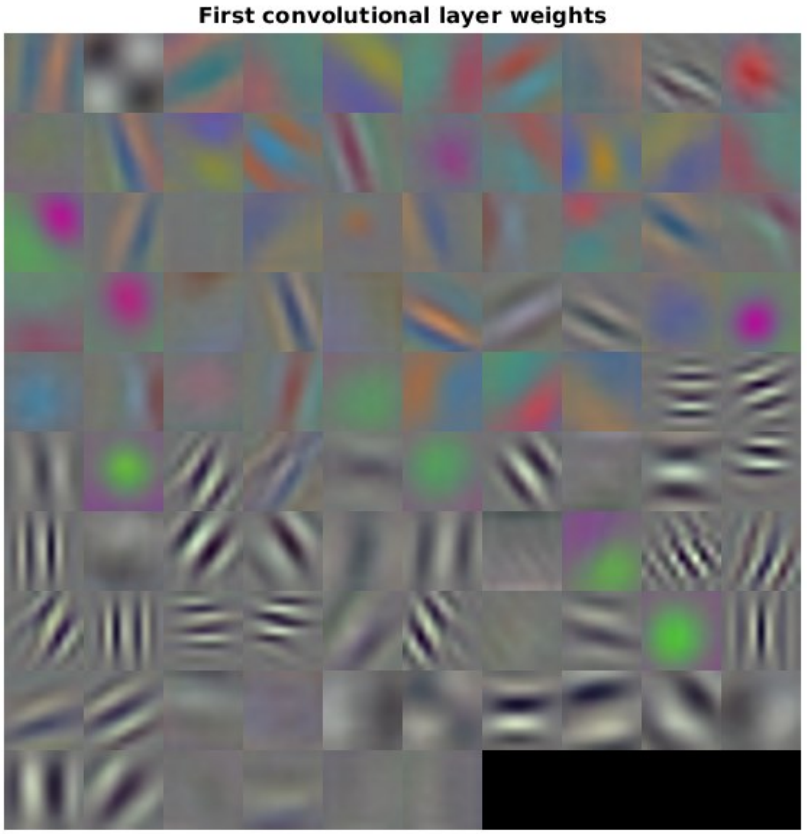
CNN for image recognition (2/4)

AlexNet implemented in Matlab

1	'data'	Image Input	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0]
6	'conv2'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1]
11	'relu3'	ReLU	ReLU
12	'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'fc8'	Fully Connected	1000 fully connected layer
24	'prob'	Softmax	softmax
25	'output'	Classification Output	crossentropyex with 'tench', 'goldfish', and 998 other classes

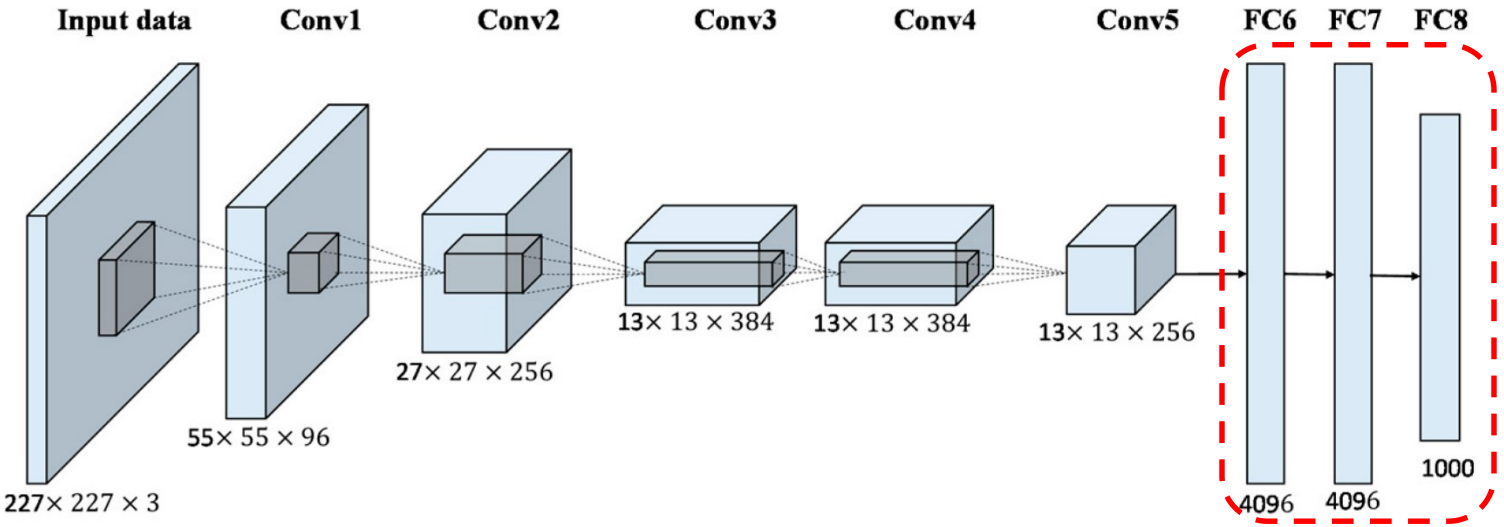
CNN for image recognition (3/4) visualize the first CL

```
% Load pre-trained AlexNet
net = alexnet()

w1 = net.Layers(2).Weights;

% Scale and resize the weights for
% visualization
w1 = mat2gray(w1);  w1 = imresize(w1,5);

% Display a montage of network weights.
% There are 96 individual sets of
% weights in the first layer.
figure; montage(w1)
title('First convolutional layer weights')
```



https://jp.mathworks.com/help/vision/examples/image-category-classification-using-deep-learning.html

Transfer learning based on CNN



- We can replace the fully connected layers with properly defined sizes, and train the network using a new data set.
- This way, we can re-use a well-trained CNN for solving different (image recognition related) problems.
- This is called transfer learning.

Homework of today

- Try to investigate the output of each layer of the AlexNet when an image (e.g. a cat or a dog) is given as the input, and try to understand the physical meaning of these outputs.
- You can use Matlab, Python, or any tool you like.