Deep Learning Seminar
Chapter 9-2. Convolutional Networks

Heechul Lim

Department of Information and Communication Engineering

DGIST

2017.10.11

Chapter 9. Convolutional Networks

Chapter 9. Convolutional Networks

- **Part 2**

  9.6 Structured Outputs

  9.7 Data Types

  9.8 Efficient Convolution Algorithms

  9.9 Random or Unsupervised Features

  9.10 The Neuroscientific Basis for Convolutional Networks

  9.11 Convolutional Networks and the History of Deep Learning

Chapter 9. Convolutional Networks

- **Part 2**

  9.6 Structured Outputs

  9.7 Data Types

  9.8 Efficient Convolution Algorithms

  9.9 Random or Unsupervised Features

  9.10 The Neuroscientific Basis for Convolutional Networks

  9.11 Convolutional Networks and the History of Deep Learning

Efficient Convolution Algorithms

- Modern CNN compute more than one million units
- Exploiting parallel computation resources are essential
- Selecting an appropriate convolution algorithm is important
- If the kernel is separable, naive convolution is inefficient
- Naive d-dimensional convolution requires $O(\boldsymbol{W}^{\boldsymbol{d}})$ runtime
  - W is wide element's number in each dimension
- Separable convolution requires $O(\boldsymbol{W} \times \boldsymbol{d})$ runtime

Separable kernel convolution definition

- **Definition of convolution 2D**

$$y[m,n] = h[m,n] * x[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} h[i,j] \cdot x[m-i, n-j]$$

- $h[m,n]$ **is separable**

$$h[m,n] = h_1[m] \cdot h_2[n]$$

y : input image tensor

m : column index of y

n : row index of y

h : tensor of convolution kernels

source : http://www.songho.ca/dsp/convolution/convolution2d_separable.html

Separable kernel convolution definition

- **Definition of convolution 2D**

$$y[m,n] = h[m,n] * x[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} h[i,j] \cdot x[m-i, n-j]$$

- $h[m,n]$ **is separable**

$$h[m,n] = h_1[m] \cdot h_2[n]$$

- **Substitute** $h[m,n]$ **into the equation**

$$\begin{aligned} y[m,n] = h[m,n] * x[m,n] &= \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} h[i,j] \cdot x[m-i, n-j] \\ &= \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} h_1[i] \cdot h_2[j] \cdot x[m-i, n-j] \\ &= \sum_{j=-\infty}^{\infty} h_2[j] \cdot \left[\sum_{i=-\infty}^{\infty} h_1[i] \cdot x[m-i, n-j] \right] \end{aligned}$$

source : https://www.slideshare.net/viisonartificial2012/grupo-2-convolution-separable

Separable kernel definition

- **Definition of convolution 1D**

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$

- **Separable 2D convolution: twice of 1D convolution**

$$\begin{aligned} y[m,n] = \left(h_1[m] \cdot h_2[n]\right) * x[m,n] &= h_2[n] * \left(h_1[m] * x[m,n]\right) \\ &= h_1[m] * \left(h_2[n] * x[m,n]\right) \end{aligned}$$

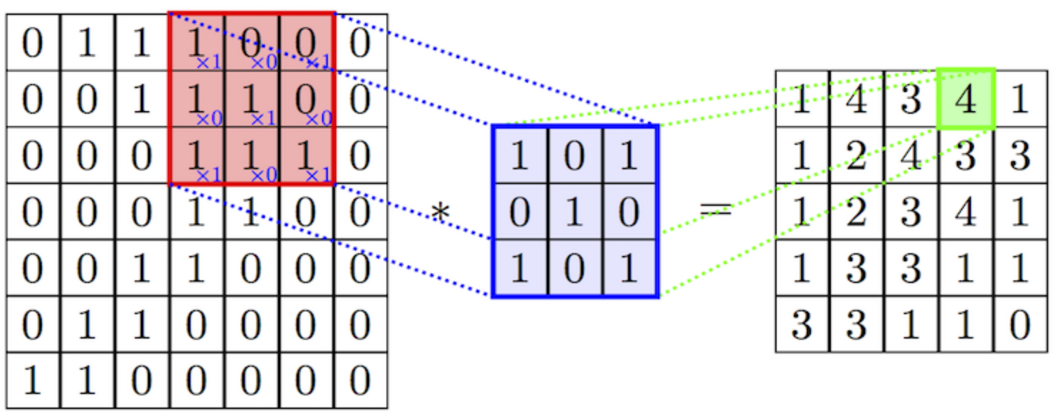
Separable kernel

- **Concept**

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix} \qquad \begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix} = \begin{bmatrix}1\\2\\1\end{bmatrix} \cdot \begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

Input Separable Kernel

- **Convolution process image**



source : https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html

Computation costs

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix} \qquad \begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix} = \begin{bmatrix}1\\2\\1\end{bmatrix}\cdot\begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

Input Separable Kernel

- **A: Convolution 2D**

$$\boxed{1\cdot 1}+\boxed{2\cdot 2}+\boxed{3\cdot 1}+\boxed{4\cdot 2}+\boxed{5\cdot 4}+\boxed{6\cdot 2}+\boxed{7\cdot 1}+\boxed{8\cdot 2}+\boxed{9\cdot 1}=80$$

- **B: Separable convolution 2D**

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix} * \begin{bmatrix}1\\2\\1\end{bmatrix} = \begin{bmatrix}\boxed{1\cdot 1}+\boxed{4\cdot 2}+\boxed{7\cdot 1} & \boxed{2\cdot 1}+\boxed{5\cdot 2}+\boxed{8\cdot 1} & \boxed{3\cdot 1}+\boxed{6\cdot 2}+\boxed{9\cdot 1}\end{bmatrix} = \begin{bmatrix}16 & 20 & 24\end{bmatrix}$$

$$\begin{bmatrix}16 & 20 & 24\end{bmatrix} * \begin{bmatrix}1 & 2 & 1\end{bmatrix} = \boxed{16\cdot 1}+\boxed{20\cdot 2}+\boxed{24\cdot 1}=80$$

- **Computation cost**
  - A: $O(W^d)$ vs B: $O(W \times d)$

Strategy for pixel-wise labeling of images

Convolution in 1D

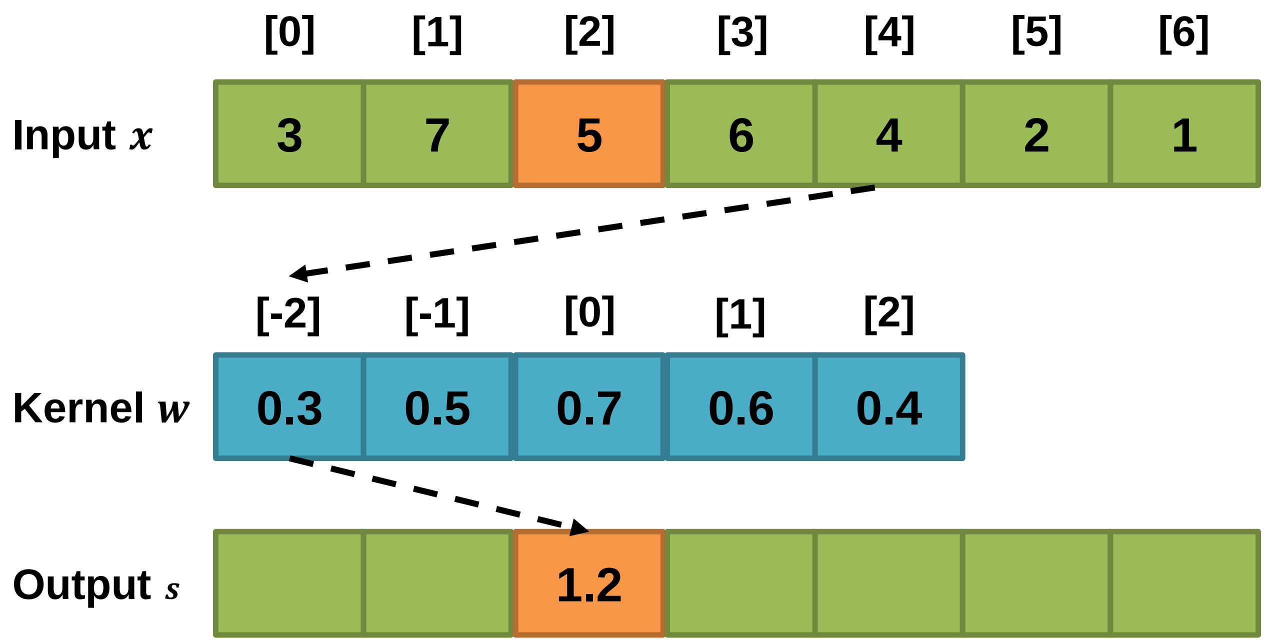
- **Definition**

$$y[k] = h[n] * x[n] = \sum_{i=-\infty} x[i]h[k-i] \quad k = 0, 1, \ldots, 6$$

$$\begin{aligned}
y[0] &= \textstyle\sum_{i=-\infty}^{\infty} x[i]h[-i] &&= x[0]h[0] + 0 + 0 \\
y[1] &= \textstyle\sum_{i=-\infty}^{\infty} x[i]h[1-i] &&= x[0]h[1] + x[1]h[0] + 0 \\
y[2] &= \textstyle\sum_{i=-\infty}^{\infty} x[i]h[2-i] &&= x[0]h[2] + x[1]h[1] + x[2]h[0] \\
y[3] &= \textstyle\sum_{i=-\infty}^{\infty} x[i]h[3-i] &&= x[0]h[3] + x[1]h[2] + x[2]h[1] \\
y[4] &= \textstyle\sum_{i=-\infty}^{\infty} x[i]h[4-i] &&= x[1]h[3] + x[2]h[1] + 0 \\
y[5] &= \textstyle\sum_{i=-\infty}^{\infty} x[i]h[5-i] &&= x[2]h[3] + 0 + 0
\end{aligned}$$

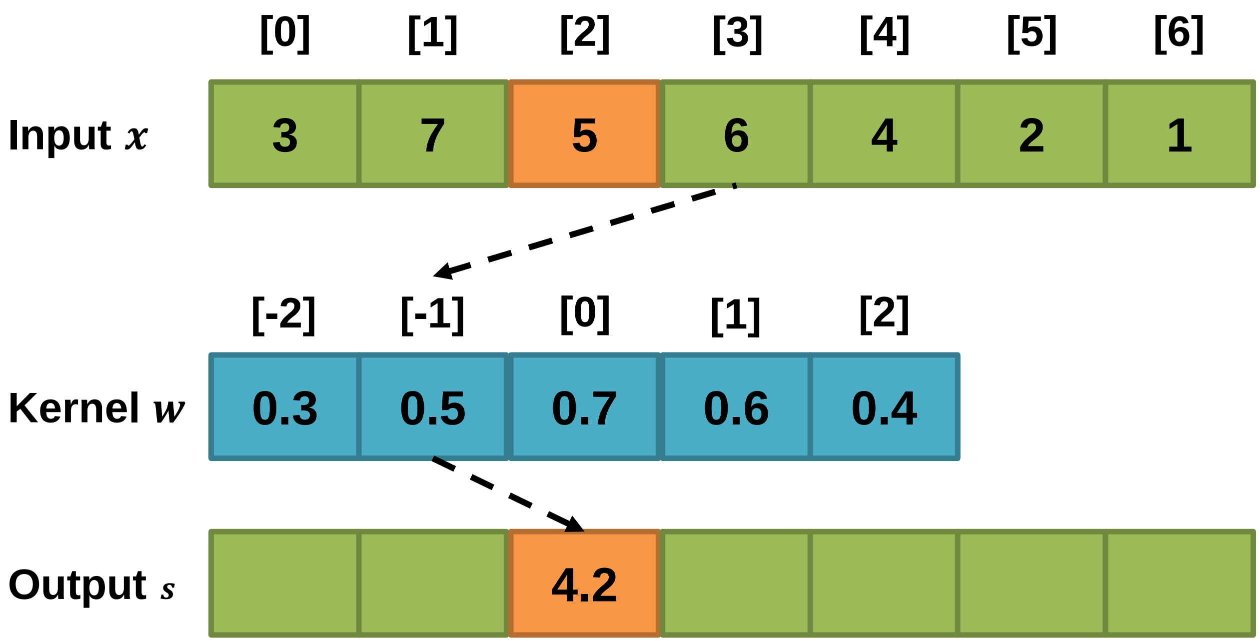
Convolution in 1D

- Discrete 1D convolution in computer



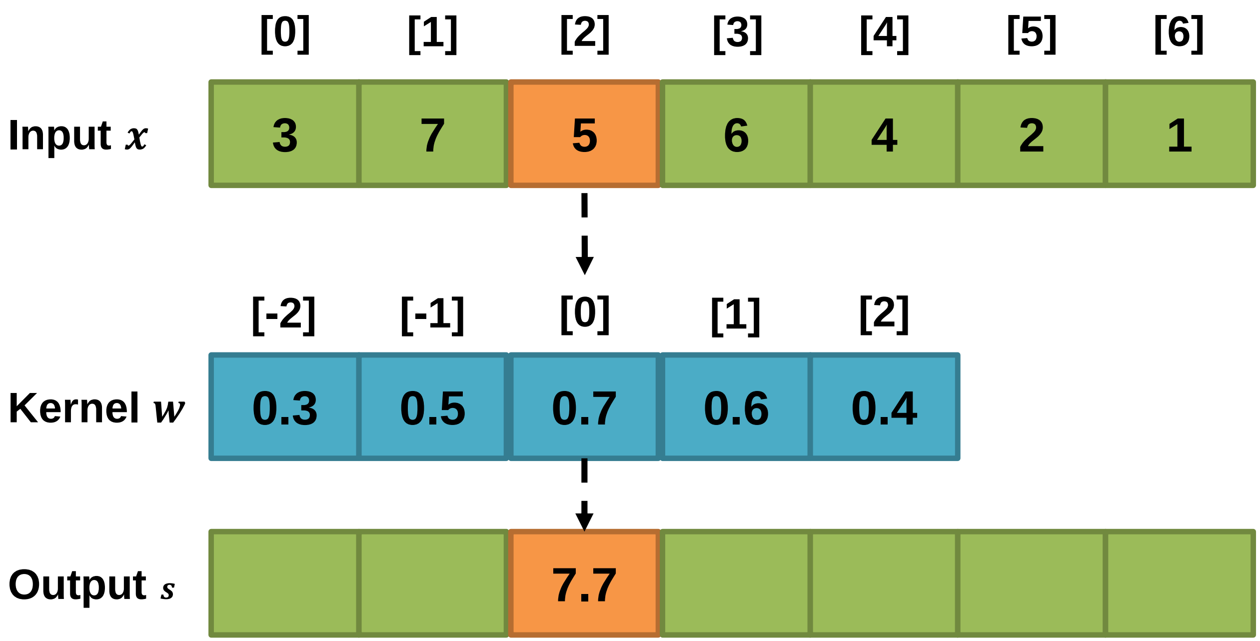
Convolution in 1D

- **Discrete 1D convolution in computer**



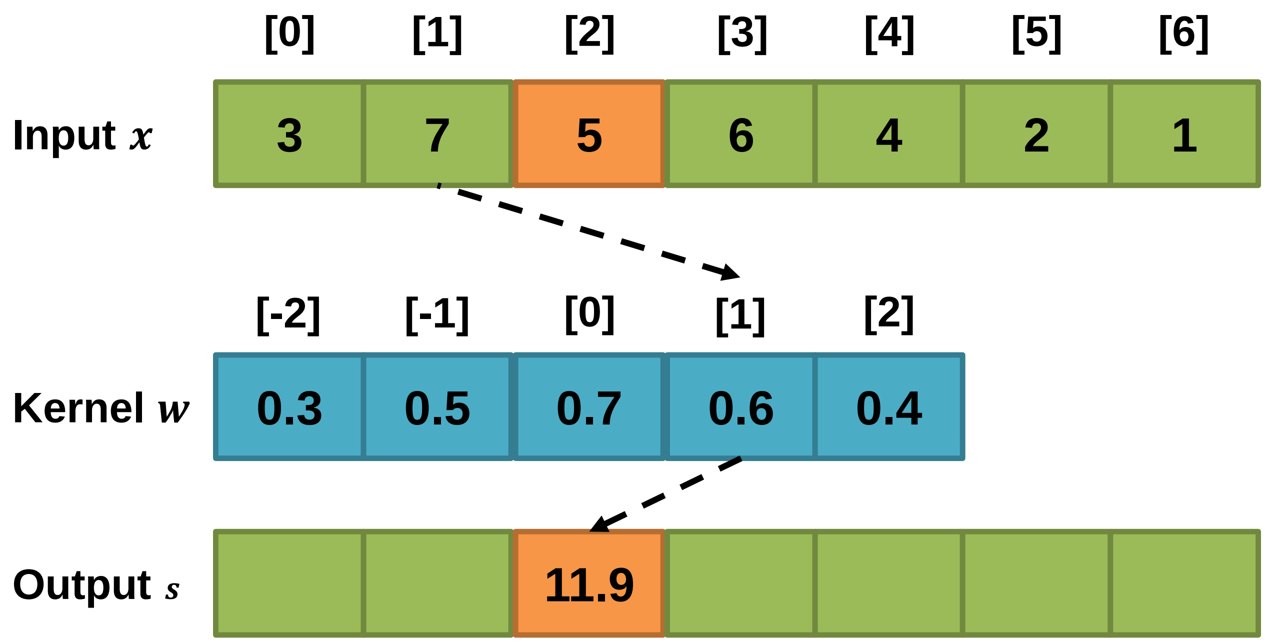
Convolution in 1D

- **Discrete 1D convolution in computer**



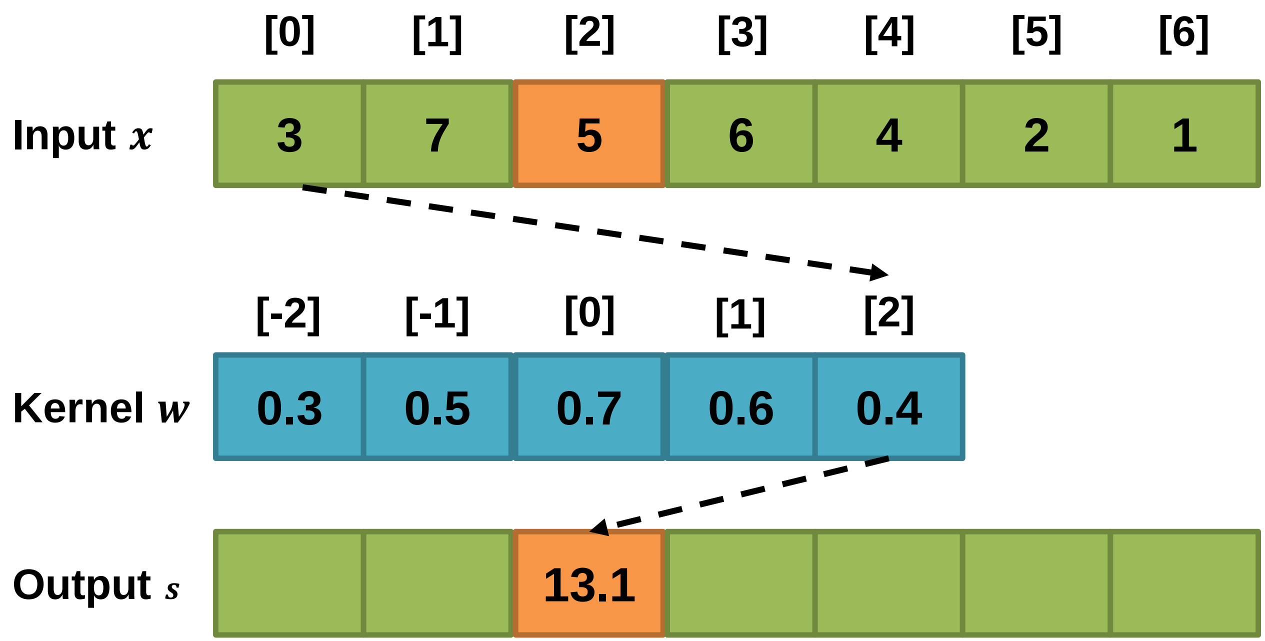
Convolution in 1D

- **Discrete 1D convolution in computer**



Convolution in 1D

- **Discrete 1D convolution in computer**



Convolution code in 2D

```
// find center position of kernel (half of kernel size)
kCenterX = kCols / 2;
kCenterY = kRows / 2;

for(i=0; i < rows; ++i)              // rows
{
    for(j=0; j < cols; ++j)          // columns
    {
        for(m=0; m < kRows; ++m)     // kernel rows
        {
            mm = kRows - 1 - m;      // row index of flipped kernel

            for(n=0; n < kCols; ++n) // kernel columns
            {
                nn = kCols - 1 - n;  // column index of flipped kernel

                // index of input signal, used for checking boundary
                ii = i + (m - kCenterY);
                jj = j + (n - kCenterX);

                // ignore input samples which are out of bound
                if( ii >= 0 && ii < rows && jj >= 0 && jj < cols )
                    out[i][j] += in[ii][jj] * kernel[mm][nn];
            }
        }
    }
}
```

Computation costs

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Input

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Separable Kernel

- **A: Process using convolution 2D**

$$1 \cdot 1 + 2 \cdot 2 + 3 \cdot 1 + 4 \cdot 2 + 5 \cdot 4 + 6 \cdot 2 + 7 \cdot 1 + 8 \cdot 2 + 9 \cdot 1 = 80$$

- **Computation cost ($W = 3, d = 2$)**
  - $O(\boldsymbol{W^d})$
  - $\mathbf{9 \times 9 = 81,\ 81 \div 9 = 9}$

Computation costs

y : input image tensor

m : column index of y (0 ~ 2)

n : row index of y (0 ~ 2)

h : tensor of convolution kernels

I, j : (-1 ~ 1)

Definition of B

$$y[m,n] = \sum_{j=-\infty}^{\infty} h_2[j] \cdot \left[\sum_{i=-\infty}^{\infty} h_1[i] \cdot x[m-i, n-j] \right]$$

B: Process using separable convolution 2D

m: 0~2, n: 1

i: -1~1

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1\cdot1+4\cdot2+7\cdot1 & 2\cdot1+5\cdot2+8\cdot1 & 3\cdot1+6\cdot2+9\cdot1 \end{bmatrix} = \boxed{\begin{bmatrix} 16 & 20 & 24 \end{bmatrix}}$$

j: -1~1

$$\begin{bmatrix} 16 & 20 & 24 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

reuse

Computation cost ($W = 3, d = 2$)

i: -1~1

i: -1~1

n: 0~2

- $3 \times 3 + 3 \times 3 = 18$ in 1 row and 3 row $18 \times 3 = 54$, $54 \div 9 = 6$

m: 0~2

j: -1~1

Computation costs

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Input — Separable Kernel

- **A: Process using convolution 2D**
- **B: Process using separable convolution 2D**
- **Computation cost ($W = 3, d = 2$)**
  - **A** $\rightarrow O(W^d) \rightarrow 9 \times 9 = 81,\ 81 \div 9 = 9$
  - **B** $\rightarrow O(W \times d) \rightarrow$
    $3 \times 3 + 3 \times 3 = 18$ **in** 1 **row and** 3 **row** $18 \times 3 = 54,\ 54 \div 9 = 6$

Chapter 9. Convolutional Networks

- **Part 2**

  9.6 Structured Outputs

  9.7 Data Types

  9.8 Efficient Convolution Algorithms

  9.9 Random or Unsupervised Features

  9.10 The Neuroscientific Basis for Convolutional Networks

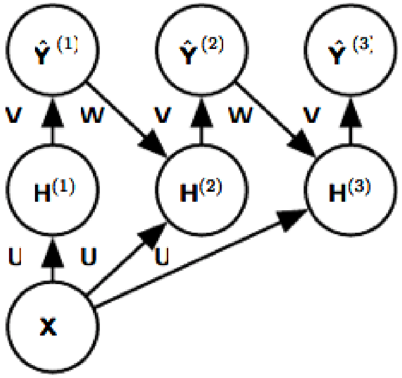
  9.11 Convolutional Networks and the History of Deep Learning

Structured outputs

- **Pixel-wise labeling of images**
- **Tensor $S_{i,j,k}$**
  - The **probability** that pixel (i, j) of the input belongs to class k
  - Allows the model to **label every pixel**

Strategy for pixel-wise labeling of images

- Produce an **initial guess** of the image labels
- **Refine** this using the interactions between neighboring pixels
- **Repeating** this refinement step several times
- **Sharing** weights between the last layers of the deep net



X : Input image tensor

Y : Probability distribution over labels for each pixel

H : Hidden representation

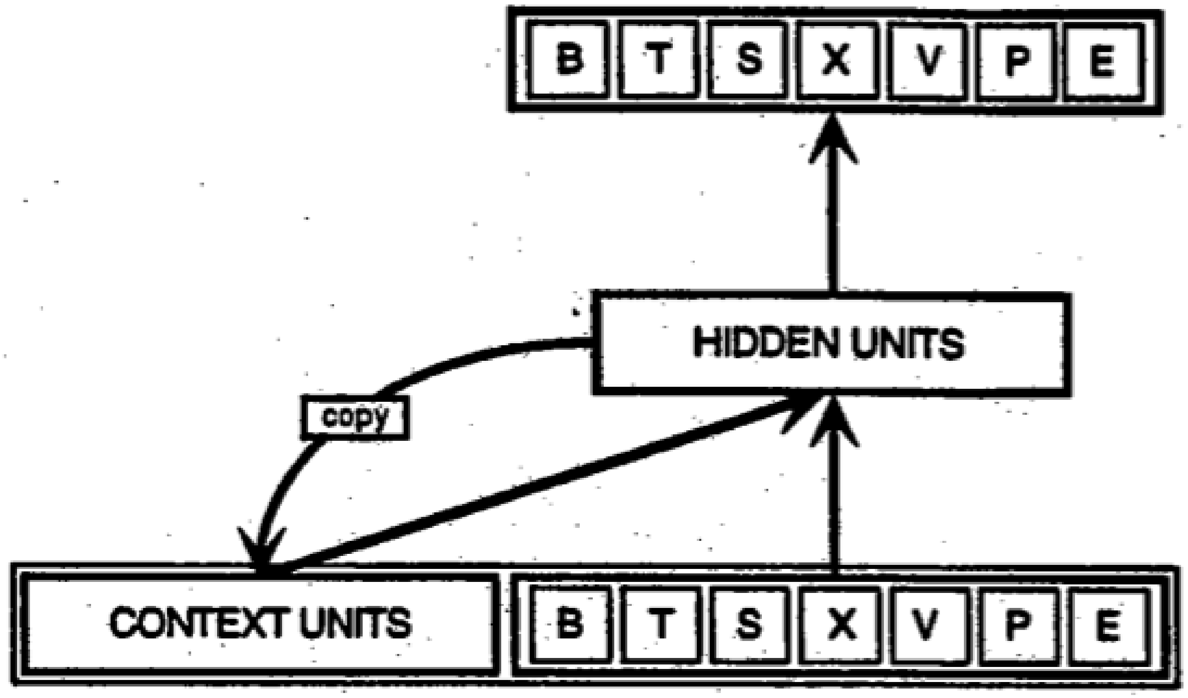
U : Tensor of convolution kernels

V : Tensor of kernels to produce an estimate of the labels

W : Kernel tensor to convolve over Y to provide input to H

Strategy for pixel-wise labeling of images

- **Basic RNN structure introduced by Elman**
- **Receiving input data and context unit in hidden layer**
- **Feedback structure that seems like memory**

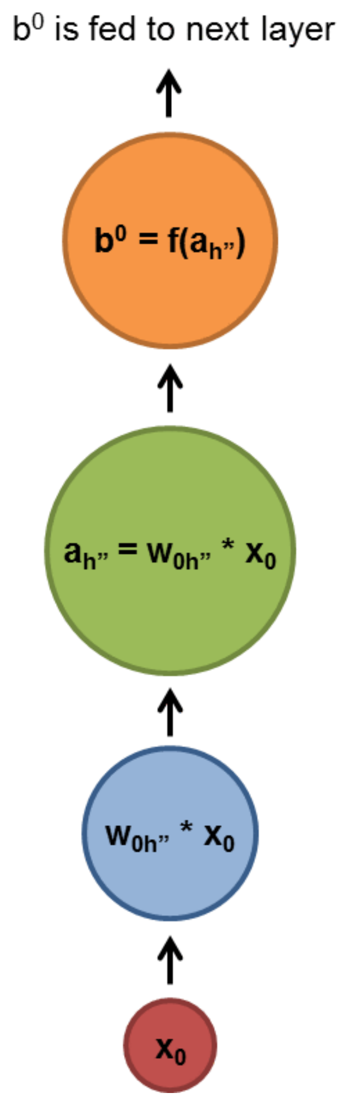


source : Elman, J. L. Finding structure in time. In *Cognitive Sciences*, 1990

Strategy for pixel-wise labeling of images

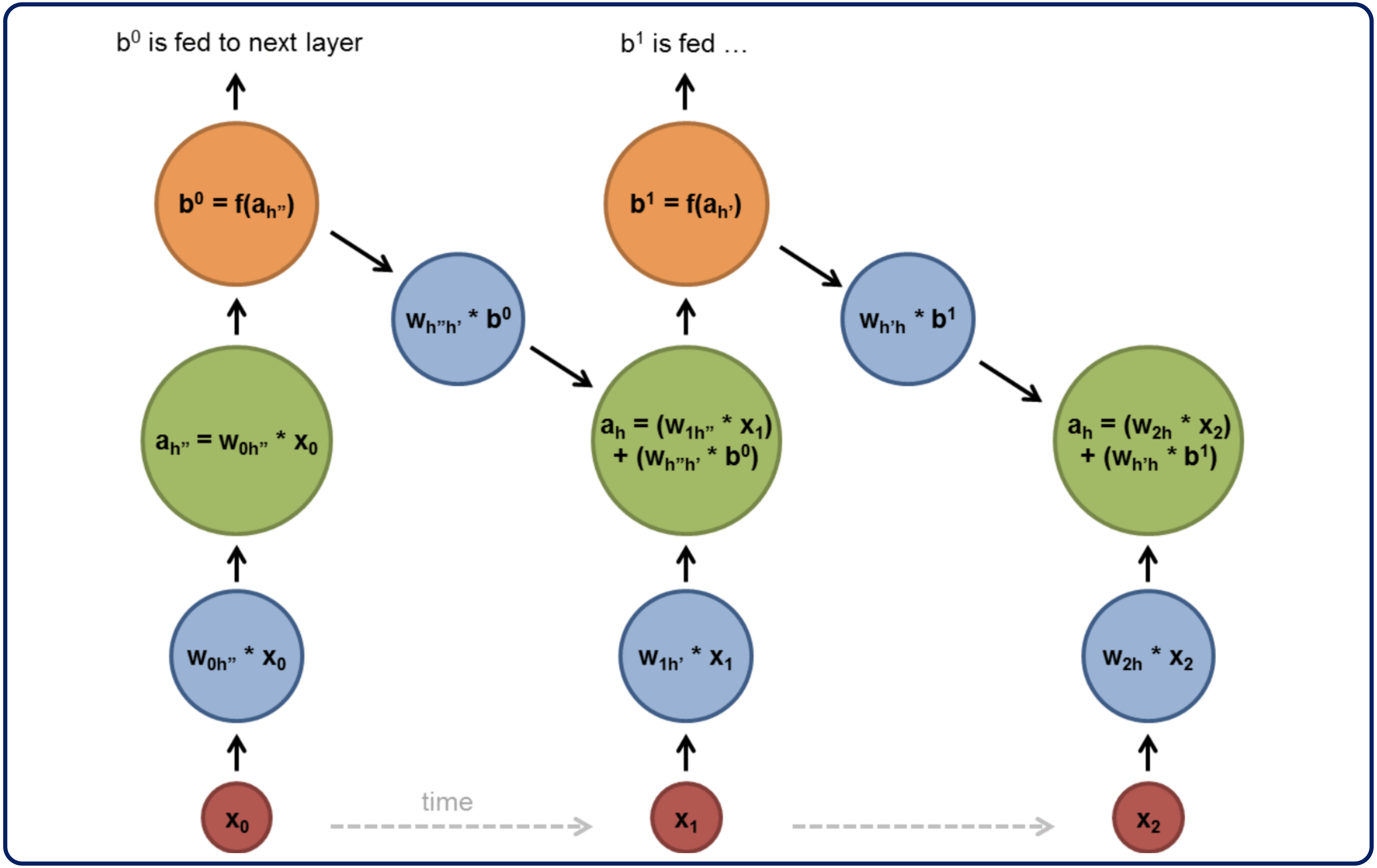
?

Strategy for pixel-wise labeling of images



source : https://imgur.com/kpZBDfV

Strategy for pixel-wise labeling of images



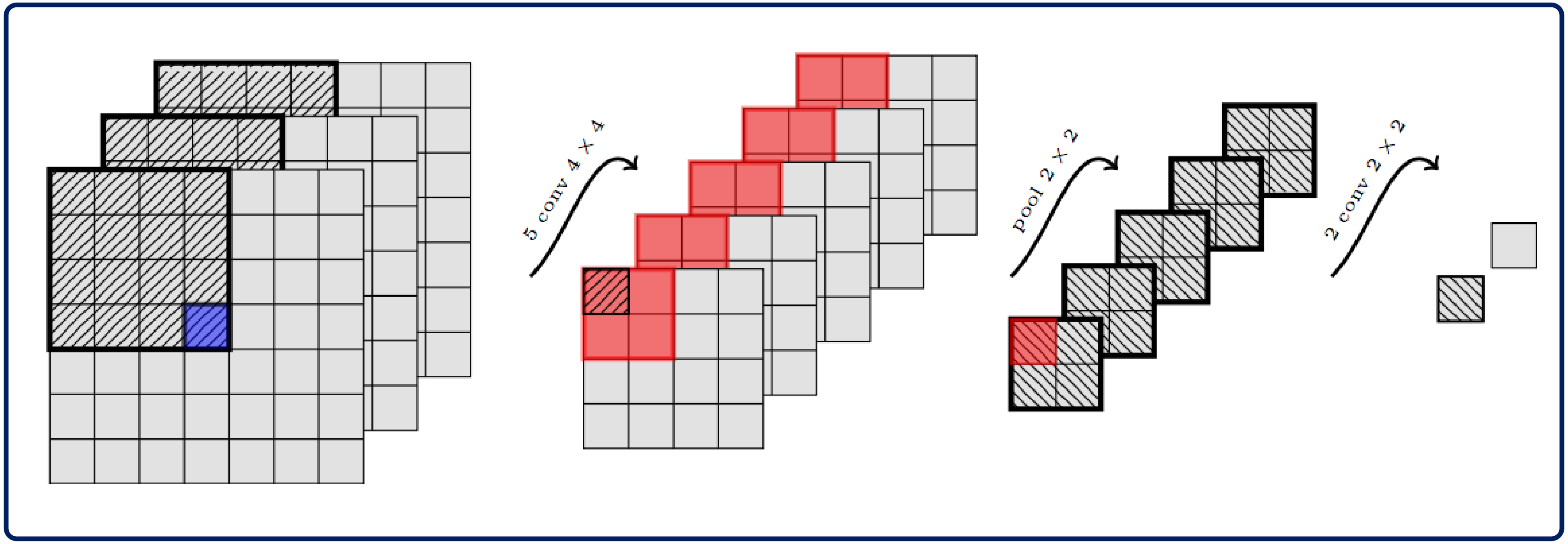
source : https://imgur.com/kpZBDfV

Recurrent Convolutional Neural Networks for Scene Labeling

- The goal is to assign a class label to each pixel
- To ensure a high class accuracy,
  it is essential to capture long range (pixel) label dependencies
- We propose a recurrent convolutional neural network

Recurrent Convolutional Neural Networks for Scene Labeling

- A simple convolutional network
- 4 × 4 convolutions, followed by one 2 × 2 pooling, followed by two 2 × 2 convolutions
- Each 1 × 1 output plane : A score for a given class



source : 2014 ICML--Pinheiro--Recurrent convolutional neural networks for scene labeling

Recurrent Network Approach

- $\mathbf{F}^p = [f(\mathbf{F}^{p-1}), I^p_{i,j,k}], \qquad \mathbf{F}^1 = [\mathbf{0}, I_{i,j,k}]$
- $L(f) + L(f \circ f) + \ldots + L(f \boxed{\circ^P} f)$ $\quad \boxed{(f \circ g)(x) = f(g(x))}$
- $f(I_{i,j,k}; (\mathbf{W}, \mathbf{b})) = \mathbf{W}_M \mathbf{H}_{M-1}$
- $\mathbf{H}_m = \tanh(\mathrm{pool}(\mathbf{W}_m \mathbf{H}_{m-1} + \mathbf{b}_m))$
- $$p(c|I_{i,j,k}; (\mathbf{W}, \mathbf{b})) = \frac{e^{f_c(I_{i,j,k};(\mathbf{W},\mathbf{b}))}}{\sum_{d \in \{1,\ldots,N\}} e^{f_d(I_{i,j,k};(\mathbf{W},\mathbf{b}))}}$$

I : Input image data
(i, j) : Location (i, j) of the training image k
k : Training image channel number
p : Instance number of the network (1 ≤ p ≤ P)
L : Maximum likelihood

W : weight parameter (toeplitz matrix)
b : bias vector
tanh : point-wise hyperbolic tangent function
pool : max-pooling function
m : {1,…,M} and M is number of stages

source : 2014 ICML--Pinheiro--Recurrent convolutional neural networks for scene labeling

Toeplitz matrix

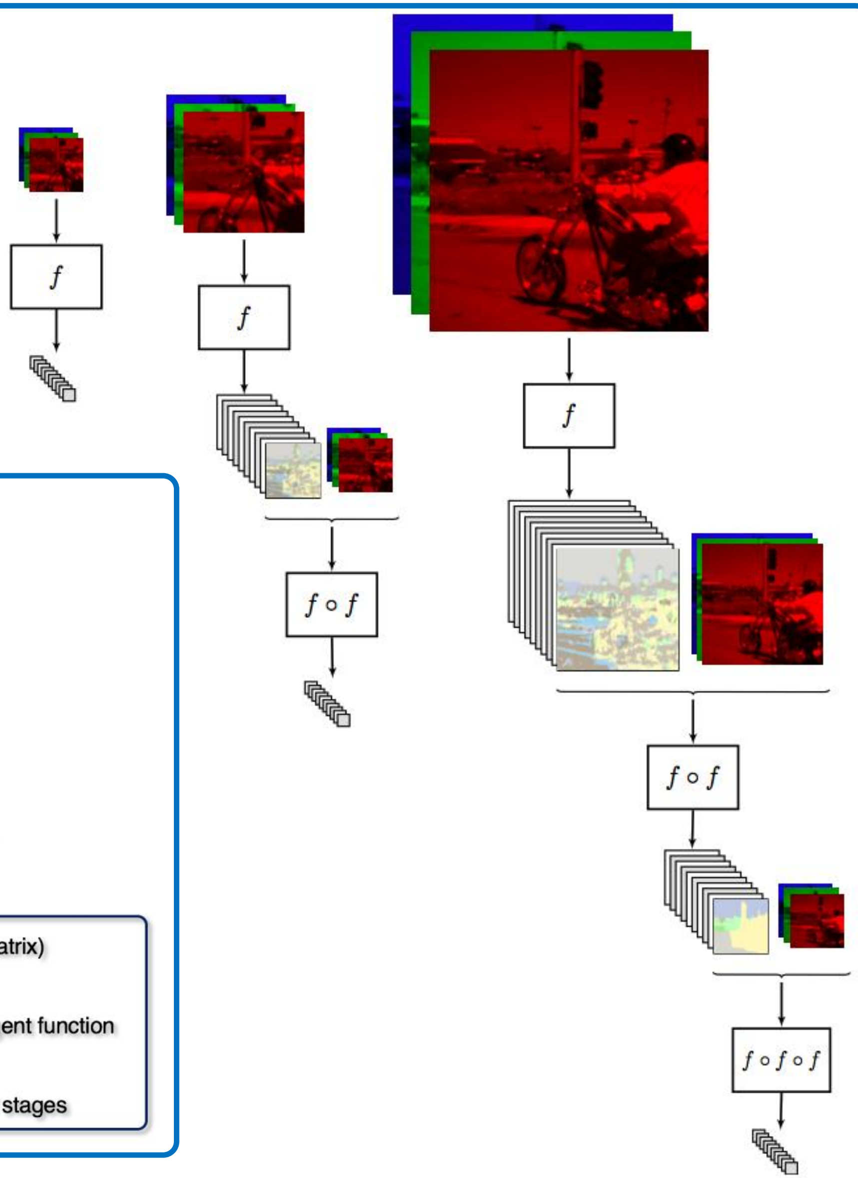
- $y[k] = h[n] * x[n] = \sum_{i=-\infty}^{} x[i]h[k-i] \quad k = 0, 1, \ldots, 6$

- $$\begin{aligned} y[0] &= \textstyle\sum_{i=-\infty}^{\infty} x[i]h[-i] &&= x[0]h[0] + 0 + 0 \\ y[1] &= \textstyle\sum_{i=-\infty}^{\infty} x[i]h[1-i] &&= x[0]h[1] + x[1]h[0] + 0 \\ y[2] &= \textstyle\sum_{i=-\infty}^{\infty} x[i]h[2-i] &&= x[0]h[2] + x[1]h[1] + x[2]h[0] \\ y[3] &= \textstyle\sum_{i=-\infty}^{\infty} x[i]h[3-i] &&= x[0]h[3] + x[1]h[2] + x[2]h[1] \\ y[4] &= \textstyle\sum_{i=-\infty}^{\infty} x[i]h[4-i] &&= x[1]h[3] + x[2]h[1] + 0 \\ y[5] &= \textstyle\sum_{i=-\infty}^{\infty} x[i]h[5-i] &&= x[2]h[3] + 0 + 0 \end{aligned}$$

- $$\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \\ y[4] \\ y[5] \end{bmatrix} = \boxed{\begin{bmatrix} h[0] & 0 & 0 \\ h[1] & h[0] & 0 \\ h[2] & h[1] & h[0] \\ h[3] & h[2] & h[1] \\ 0 & h[3] & h[2] \\ 0 & 0 & h[3] \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \end{bmatrix}}$$

source : http://www.gaussianwaves.com/2014/02/polynomials-convolution-and-toeplitz-matrices-connecting-the-dots/
http://www.purplemath.com/modules/fcncomp4.htm

Recurrent Network Approach



- $\mathbf{F}^p = [f(\mathbf{F}^{p-1}), I^p_{i,j,k}], \qquad \mathbf{F}^1 = [\mathbf{0}, I_{i,j,k}]$
- $L(f) + L(f \circ f) + \ldots + L(f \circ^P f)$
- $f(I_{i,j,k}; (\mathbf{W}, \mathbf{b})) = \mathbf{W}_M \mathbf{H}_{M-1}$
- $\mathbf{H}_m = \tanh(\text{pool}(\mathbf{W}_m \mathbf{H}_{m-1} + \mathbf{b}_m))$
- $p(c|I_{i,j,k}; (\mathbf{W}, \mathbf{b})) = \dfrac{e^{f_c(I_{i,j,k};(\mathbf{W},\mathbf{b}))}}{\sum_{d \in \{1,\ldots,N\}} e^{f_d(I_{i,j,k};(\mathbf{W},\mathbf{b}))}}$

I : Input image data
(I, j) : Location (i, j) of the training image k
k : Training image channel number
p : Instance number of the network (1 ≤ p ≤ P)
L : Maximum likelihood

W : weight parameter(toeplitz matrix)
b : bias vector
tanh : point-wise hyperbolic tangent function
pool : max-pooling function
m : {1,…,M} and M is number of stages

Strategy for pixel-wise labeling of images

- Once a prediction for each pixel is made, various methods can be used to further process
- Various graphical models can describe that
- Refer to
  - 2005 IEEE, F Ning:
    Toward automatic phenotyping of developing embryos from videos
  - 2014 NIPS, Jonathan Tompson:
    Joint training of a convolutional network and a graphical model for human pose estimation

Chapter 9. Convolutional Networks

- **Part 2**

  9.6 Structured Outputs

  9.7 Data Types

  9.8 Efficient Convolution Algorithms

  9.9 Random or Unsupervised Features

  9.10 The Neuroscientific Basis for Convolutional Networks

  9.11 Convolutional Networks and the History of Deep Learning

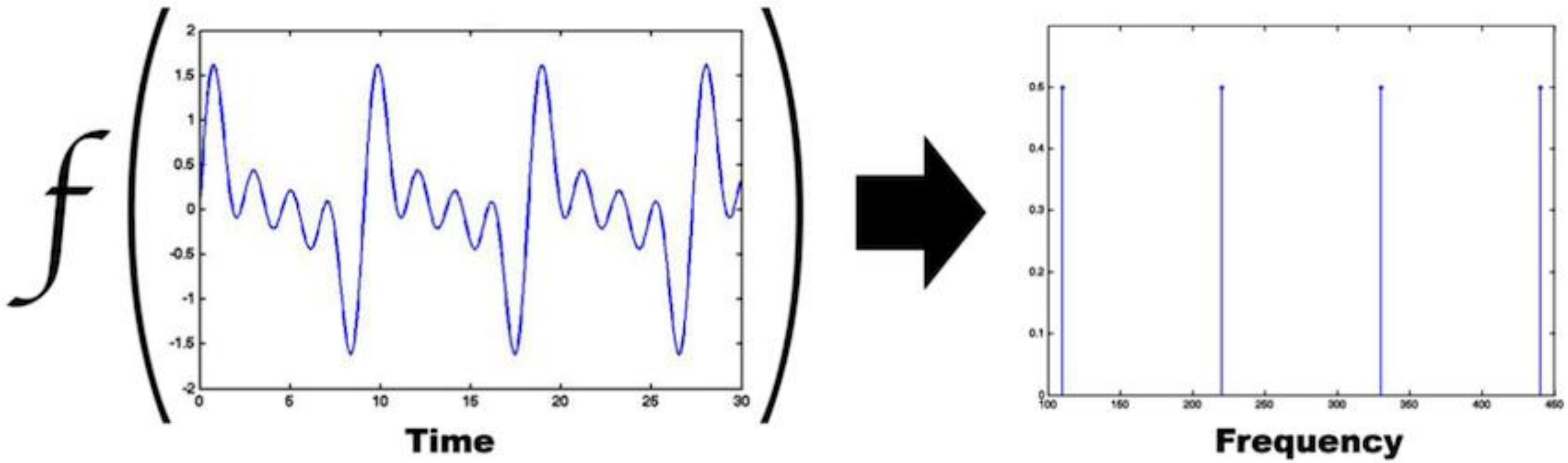
Data types

- The output of the network is allowed to have **variable size** as well as the input
- The data consists of several channels
  - Dimension: 1, 2, 3, ...
  - Channel: 1, 2, 3, …

Dimension	Single channel	Multi channel
1 D	Audio waveform	Skeleton animation data
2 D	Audio data that has been preprocessed with a Fourier transform	Color image data
3 D	Volumetric data	Color video data

Single channel

- **Fourier transform**



source : http://devonbryant.github.io/blog/2013/03/02/fourier-transforms/

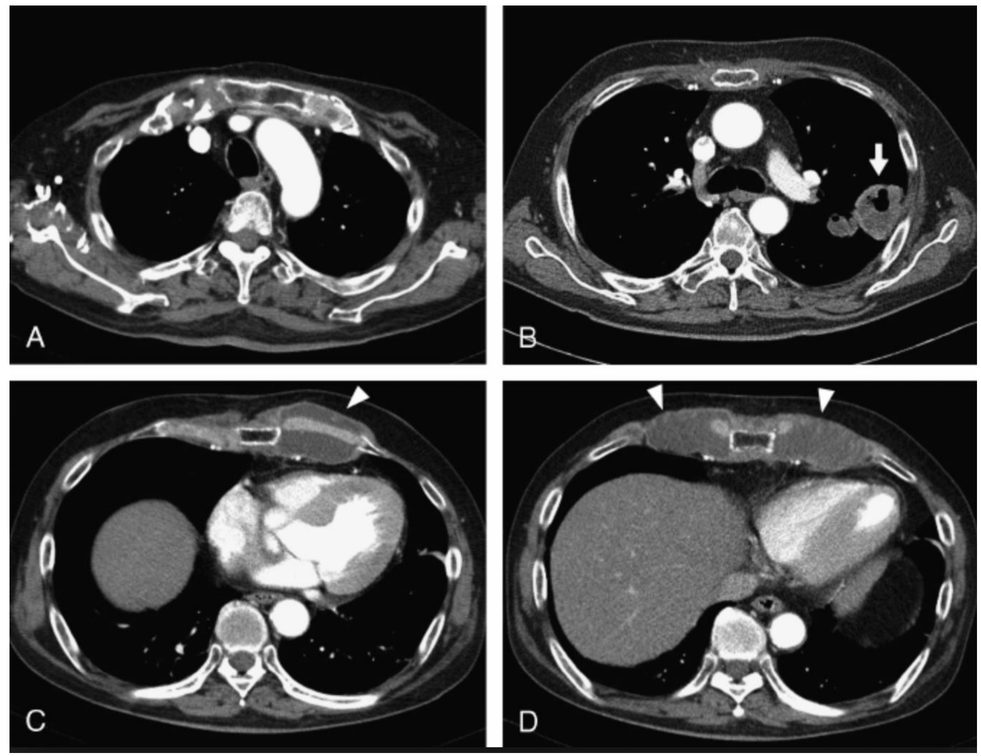
Data types

- The output of the network is allowed to have **variable size** as well as the input
- The data consists of several channels
  - Dimension : 1, 2, 3, ...
  - Channel : 1, 2, 3, …

Dimension	Single channel	Multi channel
1 D	Audio waveform	Skeleton animation data
2 D	Audio data that has been preprocessed with a Fourier transform	Color image data
3 D	$\boxed{\text{Volumetric data}}$	Color video data

Single channel

- **Volumetric data**



source : google image

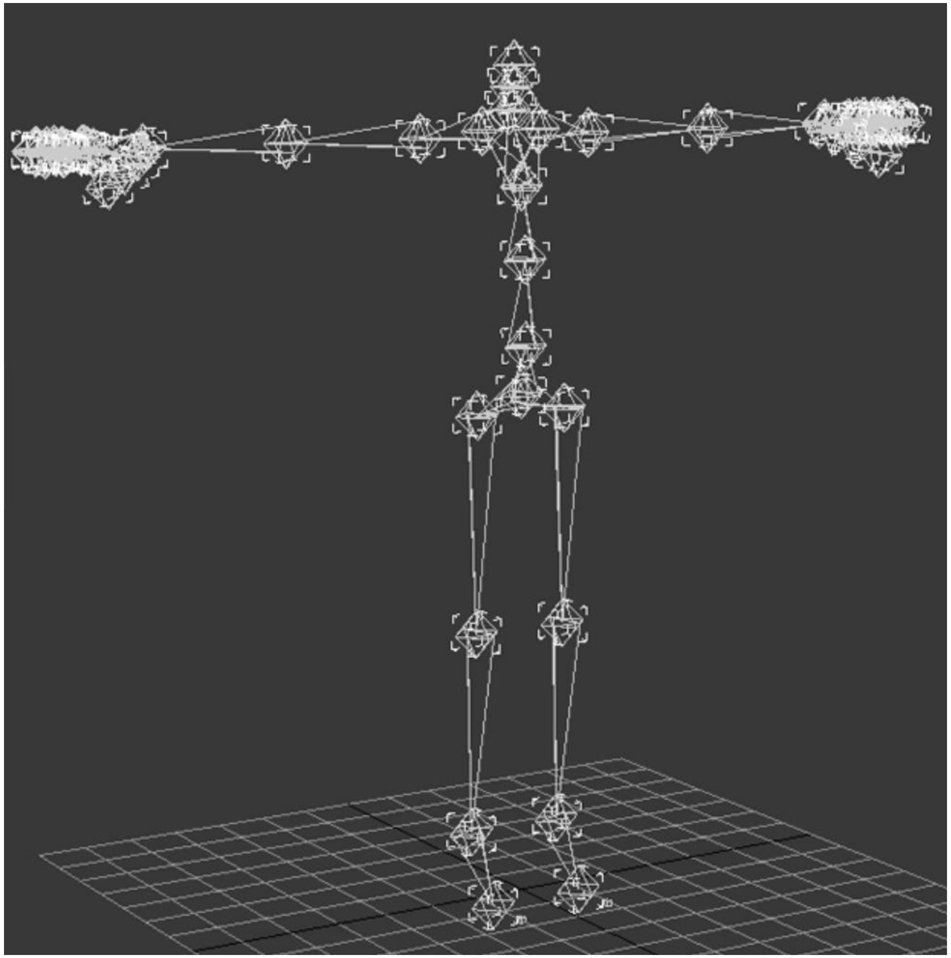
Data types

- **The output of the network is allowed to have variable size as well as the input**
- **The data consists of several channels**
  - Dimension : 1, 2, 3, ...
  - Channel : 1, 2, 3, …

Dimension	Single channel	Multi channel
1 D	Audio waveform	Skeleton animation data
2 D	Audio data that has been preprocessed with a Fourier transform	Color image data
3 D	Volumetric data	Color video data

Multi-channel

- **Skeleton animation data**



source : google image

Data types

- **The output of the network is allowed to have variable size as well as the input**
- **The data consists of several channels**
  - **Dimension : 1, 2, 3, ...**
  - **Channel : 1, 2, 3, …**

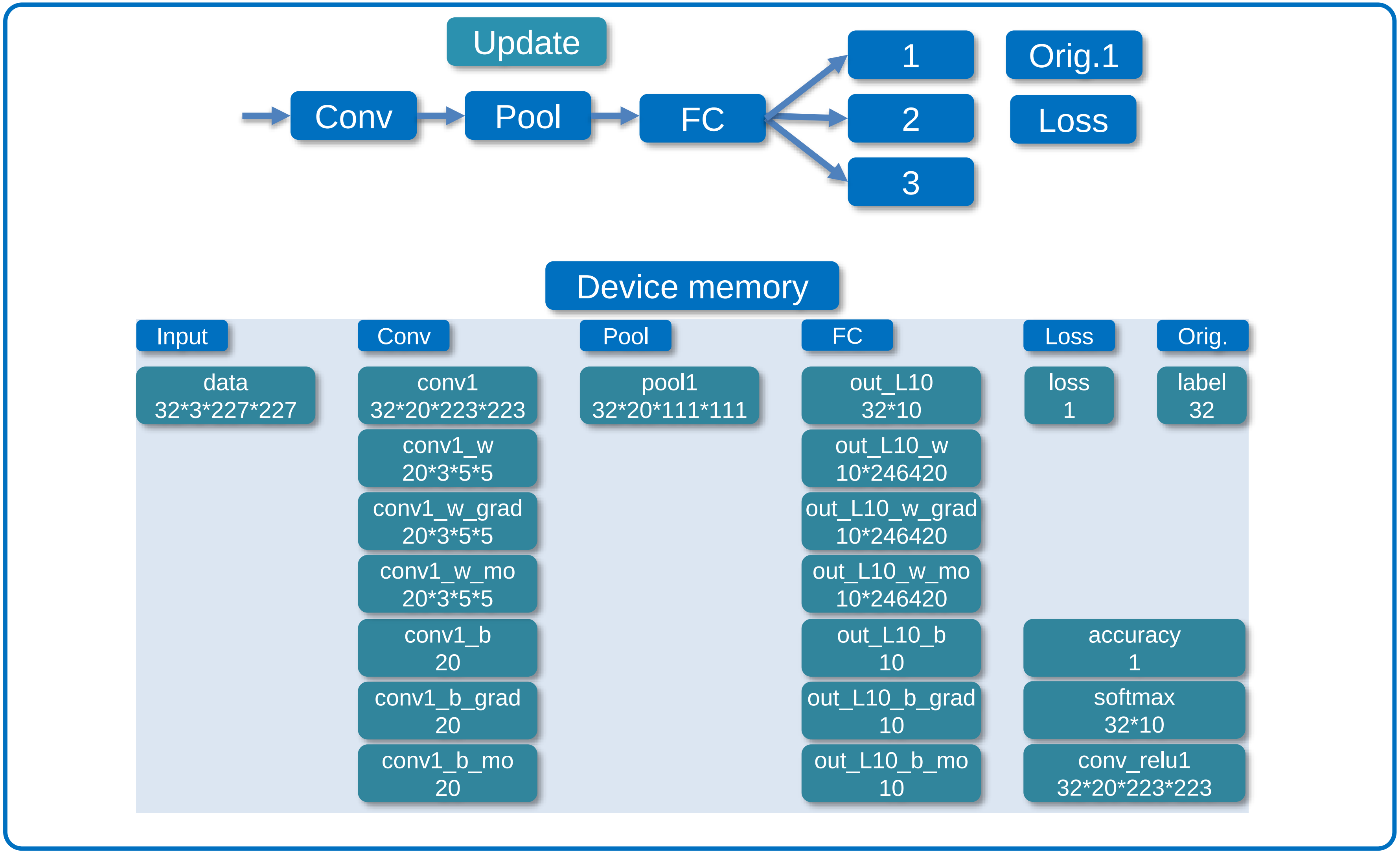
Dimension	Single channel	Multi channel
1 D	Audio waveform	Skeleton animation data
2 D	Audio data that has been preprocessed with a Fourier transform	Color image data
3 D	Volumetric data	Color video data

Data types

- CNN can also process inputs with varying spatial extents
- Variable size of input
  - To assign a class label to each pixel of the input
- Fixed-size of input
  - To assign a single class label to the entire image

Dimension	Single channel	Multi channel
1 D	Audio waveform	Skeleton animation data
2 D	Audio data that has been preprocessed with a Fourier transform	Color image data
3 D	Volumetric data	Color video data

Produce some fixed-size output



Chapter 9. Convolutional Networks

- **Part 2**

  9.6 Structured Outputs

  9.7 Data Types

  9.8 Efficient Convolution Algorithms

  9.9 **Random or Unsupervised Features**

  9.10 The Neuroscientific Basis for Convolutional Networks

  9.11 Convolutional Networks and the History of Deep Learning

Random or Unsupervised Features

- This approach was popular from roughly 2007–2013
- Today, CNN is trained in a purely supervised fashion
- Initializing kernel
  - Simply initialize them randomly.
  - Design them by hand
  - Learn the kernels with an unsupervised criterion
- Refer to
  - 2011 AISTATS, Coates Adam:
    An analysis of single-layer networks in unsupervised feature learning
  - 2009 ICML, Honglak Lee Andrew y ng:
    Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations
  - 2009 ICCV, Kevin Jarrett:
    What is the best multi-stage architecture for object recognition

Chapter 9. Convolutional Networks

- **Part 2**

  9.6 Structured Outputs

  9.7 Data Types

  9.8 Efficient Convolution Algorithms

  9.9 Random or Unsupervised Features

  9.10 The Neuroscientific Basis for Convolutional Networks

  9.11 Convolutional Networks and the History of Deep Learning

The Neuroscientific Basis for Convolutional Networks

- Neural networks that were drawn from neuroscience
- Neurophysiologists David Hubel and Torsten Wiesel
- Greatest influence on deep learning models
- Brain function that are beyond the scope of this book

TDNN and CNN

- **Lang and Hinton (1988)**
  - Time delay neural networks (TDNNs)
- **LeCun (1989)**
  - Developing the modern convolutional network

The Neuroscientific Basis for Convolutional Networks

- On a simplified, cartoon view of brain function
- A part of the brain called V1 (primary visual cortex)
- V1: First area of the brain processing of visual input
- Process
  - Light -> retina -> neuron in retina -> optic nerve -> lateral geniculate nucleus -> V1 at the back of the head

Three properties of V1

- 2-D spatial map
- Many simple cells
- Many complex cells
  - This inspires the pooling units of convolutional networks

Grandmother cells in medial temporal lobe

- Cells that respond to some specific concept
- Regardless of
  - Whether she appears in the left or right side of the image
  - Whether the image is a close-up or zoomed out shot of her entire body
  - Whether she is brightly lit, or in shadow, etc.
- An individual neuron that is activated by certain human
- More general than modern convolutional networks

The inferotemporal cortex

- When viewing an object, information flows
  - Retina -> LGN -> V1 -> V2 -> V4 -> IT
- This happens within the first 100ms of glimpsing an object
- If we interrupt the person's gaze only the firing rates, then IT proves to be similar to a CNN

Differences between CNN and the mammalian vision system

- **The human eye**
  - Very low resolution as input
  - Integrating many other senses
  - Understanding entire scenes including many objects and relationships between objects
  - Processing rich 3-D geometric information

- **CNN**
  - Large full resolution photographs as input
  - Purely visual

Gabor function

- **Response of a simple cell formula**

$$s(I) = \sum_{x \in \mathbb{X}} \sum_{y \in \mathbb{Y}} w(x,y) I(x,y)$$

- **Definition of gabor function about $w(x,y)$ in book**

$$w(x,y;\alpha,\beta_x,\beta_y,f,\phi,x_0,y_0,\tau) = \alpha \exp\left(-\beta_x x'^2 - \beta_y y'^2\right) \boxed{\cos(fx' + \phi)}$$

$$x' = (x - x_0)\cos(\tau) + (y - y_0)\sin(\tau)$$

$$y' = -(x - x_0)\sin(\tau) + (y - y_0)\cos(\tau)$$

Gabor function

- **Gerneral definition**

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\boxed{\exp\left(i\left(2\pi\frac{x'}{\lambda}+\psi\right)\right)}$$

- **Euler's formula**

$$e^{ix} = \cos x + i \sin x$$

- **Gabor function in real space**

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\boxed{\cos\left(2\pi\frac{x'}{\lambda}+\psi\right)}$$

source : https://en.wikipedia.org/wiki/Gabor_filter

Gabor function

- **Definition of gabor function about $w(x, y)$ in book**

$$w(x, y; \alpha, \beta_x, \beta_y, f, \phi, x_0, y_0, \tau) = \alpha \boxed{\exp\left(-\beta_x x'^2 - \beta_y y'^2\right)} \cos(f x' + \phi)$$

$$x' = (x - x_0)\cos(\tau) + (y - y_0)\sin(\tau)$$

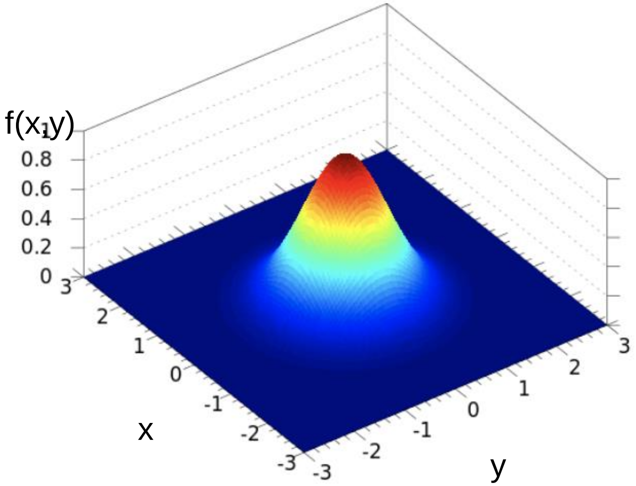
$$y' = -(x - x_0)\sin(\tau) + (y - y_0)\cos(\tau)$$

- **Gaussian function**
  - **1-demension**

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$

  - **2-demension**

$$f(x, y) = A\exp\left(-\left(\frac{(x - x_o)^2}{2\sigma_x^2} + \frac{(y - y_o)^2}{2\sigma_y^2}\right)\right)$$



source : https://en.wikipedia.org/wiki/Gaussian_function

Gabor function

- **Definition of gabor function about $w(x, y)$ in book**

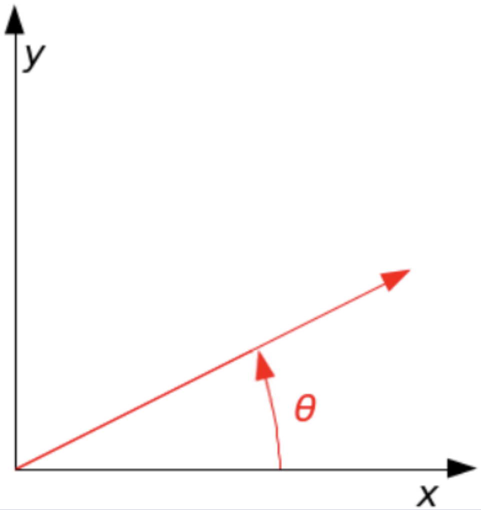
$$w(x, y; \alpha, \beta_x, \beta_y, f, \phi, x_0, y_0, \tau) = \alpha \exp\left(-\beta_x x'^2 - \beta_y y'^2\right) \cos(f x' + \phi)$$

$$\boxed{\begin{aligned} x' &= (x - x_0)\cos(\tau) + (y - y_0)\sin(\tau) \\ y' &= -(x - x_0)\sin(\tau) + (y - y_0)\cos(\tau) \end{aligned}}$$

- **Rotation maxrix**

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad R(-\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



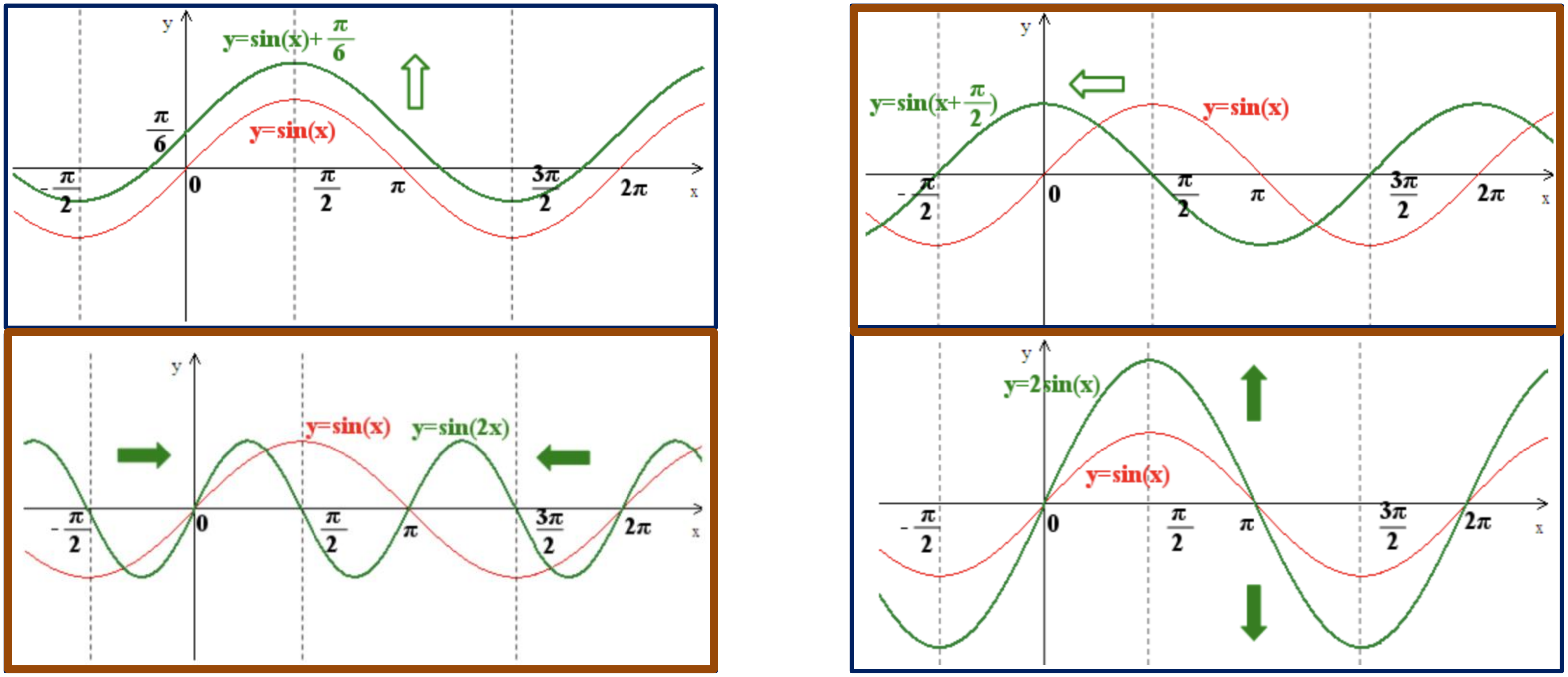
source : https://en.wikipedia.org/wiki/Rotation_matrix

Gabor function

- **Definition of gabor function about $w(x, y)$ in book**

$$w(x, y; \alpha, \beta_x, \beta_y, f, \phi, x_0, y_0, \tau) = \alpha \exp\left(-\beta_x x'^2 - \beta_y y'^2\right) \boxed{\cos(f x' + \phi)}$$

- **Trigonometrical function**



source : http://mathbang.net/529

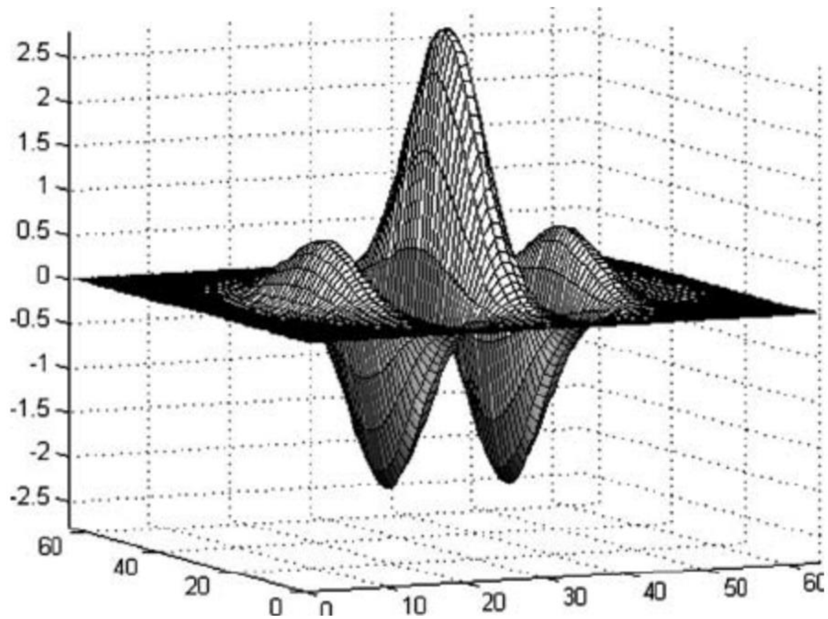
Gabor function

- **Definition of gabor function about $w(x, y)$ in book**

$$w(x, y; \alpha, \beta_x, \beta_y, f, \phi, x_0, y_0, \tau) = \alpha \exp\left(-\beta_x x'^2 - \beta_y y'^2\right) \cos(f x' + \phi)$$

$$x' = (x - x_0)\cos(\tau) + (y - y_0)\sin(\tau)$$

$$y' = -(x - x_0)\sin(\tau) + (y - y_0)\cos(\tau)$$



source : https://www.researchgate.net/figure/250147548_fig1_Fig-1-Perspective-view-of-real-Gabor-function-in-spatial-domain

Thank you