Appendix
Artificial Neural Network & Deep Learning

Prof. Chang-Chieh Cheng

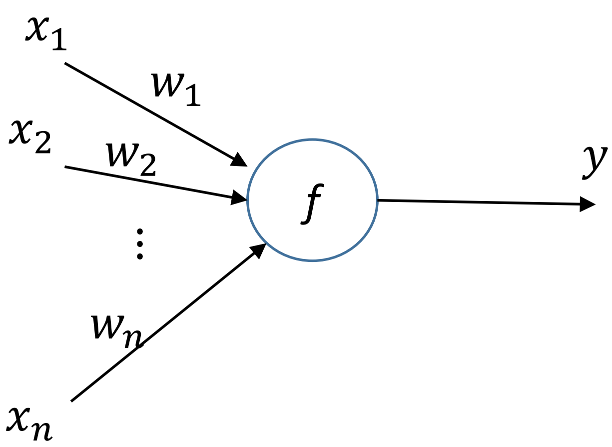
Dept. Computer Science

National Chiao Tung University, Taiwan

Artificial Neural Network

- Artificial neuron
  - Also called "Perceptron"
  - A set of inputs $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]$ and weights $\mathbf{w} = [w_1 \quad w_2 \quad \cdots \quad w_n]$
  - output $y = f(\mathbf{x}, \mathbf{w})$
  - For example:
    -

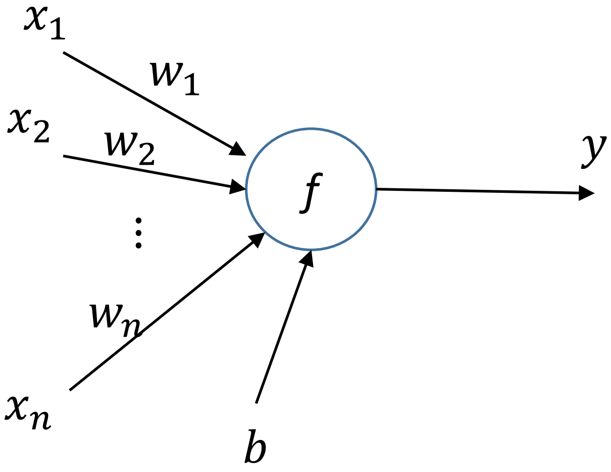
$$f(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{n} x_i w_i$$



Artificial Neural Network

- Artificial neuron
  - Biased perceptron

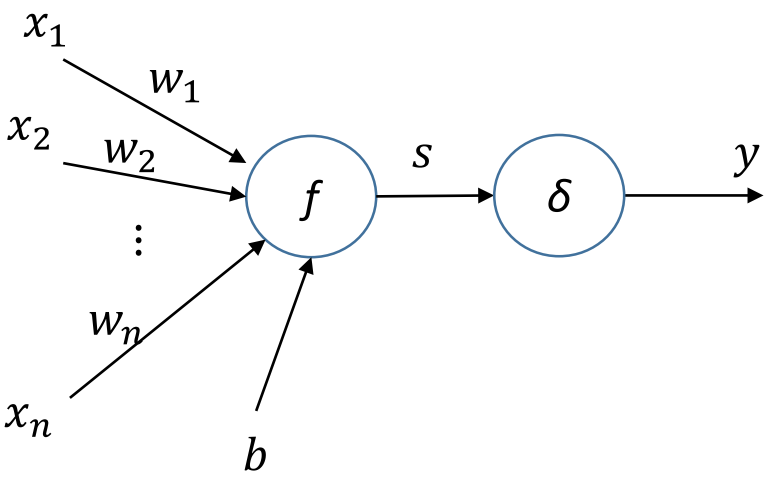
$$f(\mathbf{x}, \mathbf{w}, b) = \sum_{i=1}^{n} x_i w_i + b$$

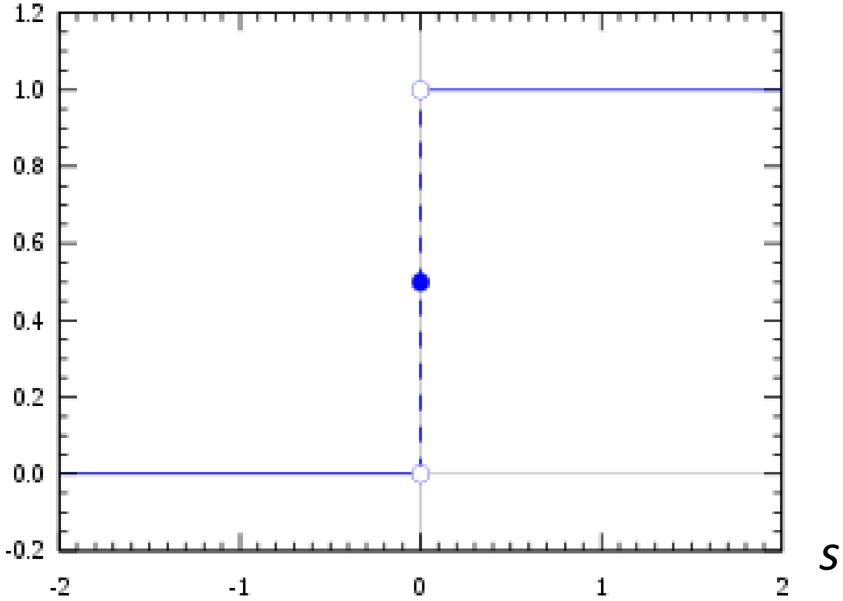


Artificial Neural Network

- Activation functions
  - Step activation

$$y = \delta(s) = \begin{cases} 0 & \text{if } s \leq 0 \\ 1 & \text{if } s > 0 \end{cases}$$



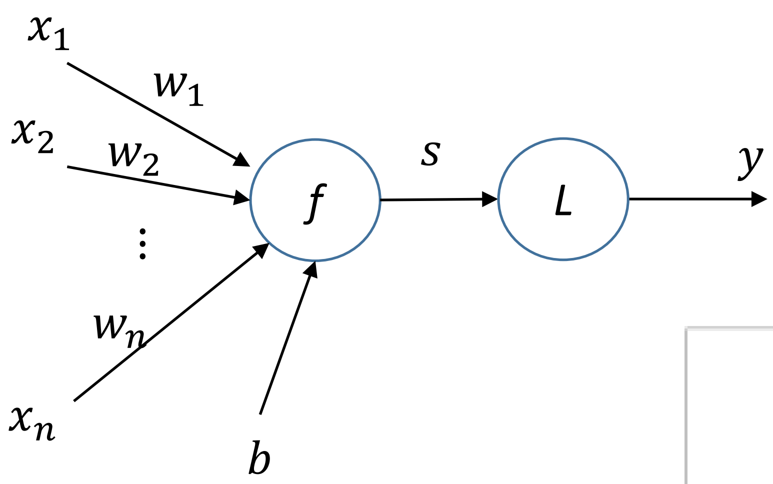


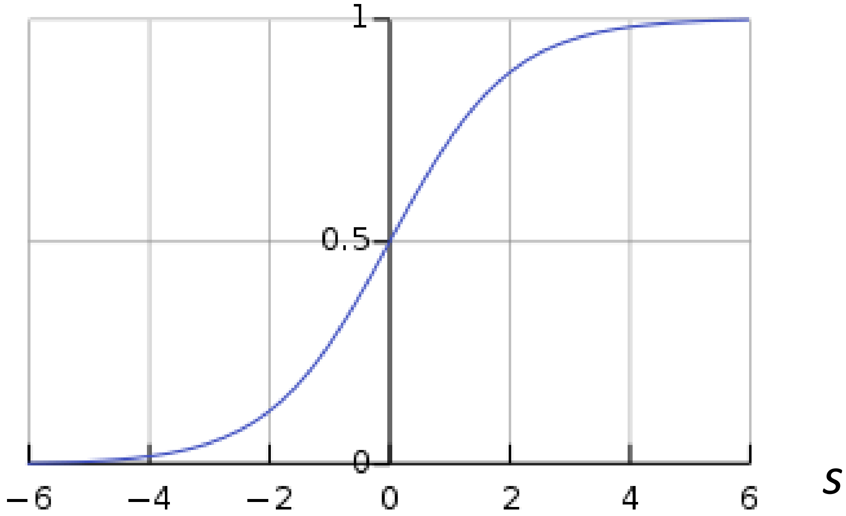
- where $\delta(y)$ is the step function

Artificial Neural Network

- Activation functions
  - Sigmoid activation

$$y = L(s) = \frac{1}{1 + e^{-s}}$$





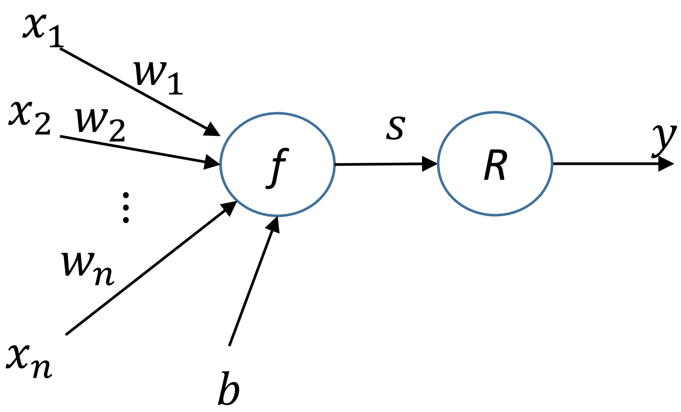
Artificial Neural Network

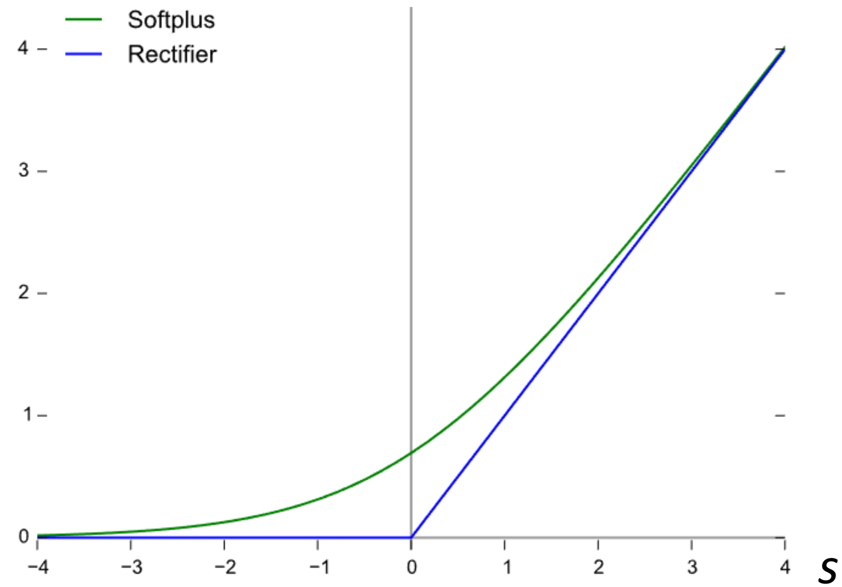
- Activation functions
  - Rectified linear unit, ReLU (or rectifier):

$$y = R(s) = \max(0, s)$$

  - Softplus, a smooth approximation to the ReLU

$$y = SR(s) = \ln(1 + e^{s})$$



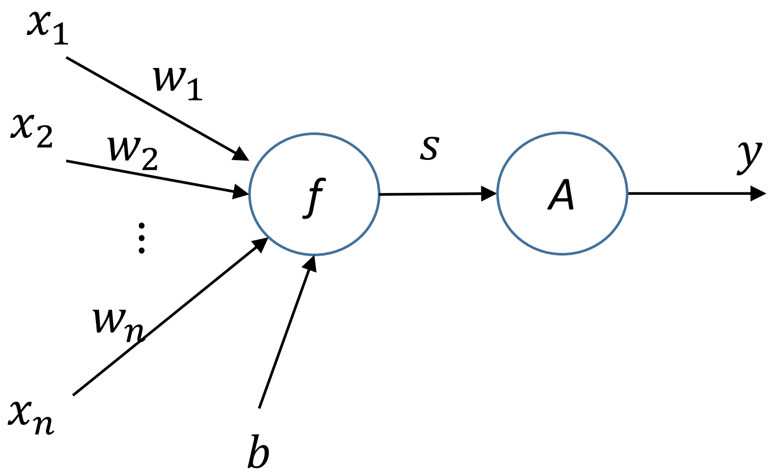


Artificial Neural Network

- Finding the best weights and bias so that the error of output is minimum.
- Error-based machine learning model (logistic model or SVM)
- $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, each $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ has a target result *t*
- $\mathbf{w} = \{w_1, w_2, \ldots, w_n\}$

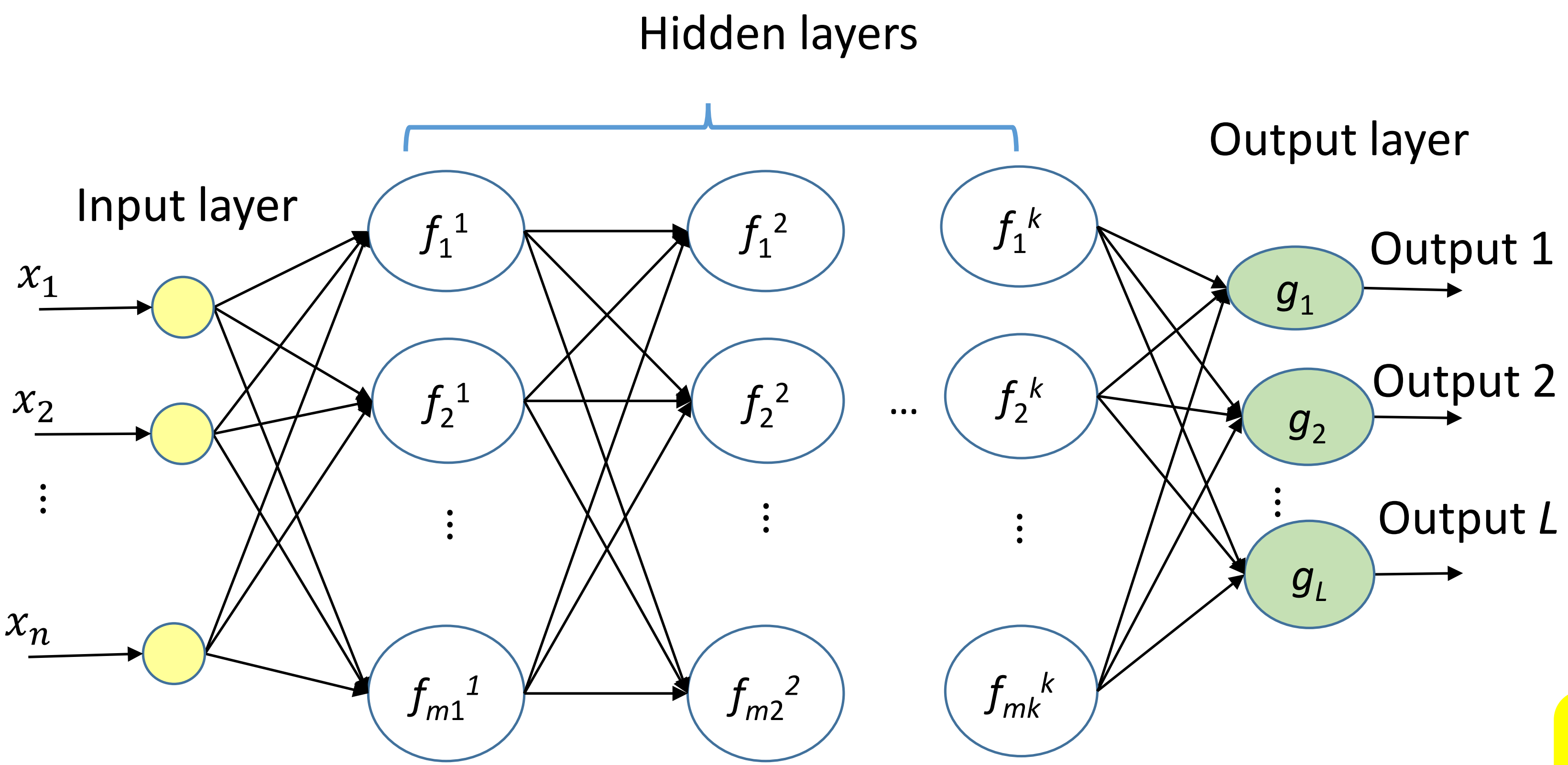
$$\mathbf{w} = \arg\min_{\mathbf{w}'} \sum_{i=1}^{m} E(t_i, y_i)$$

where $E(t, y)$ is the error of t with y.



Artificial Neural Network

- Artificial Neural network, ANN
  - Each circle in the hidden layers and output layer is a neuron
  - Each edge that goes to a neuron represents an input with a weight
  - W. McCulloch and P. Walter, "**A Logical Calculus of Ideas Immanent in Nervous Activity**". *Bulletin of Mathematical Biophysics*. 5 (4): 115–133, 1943.



Artificial Neural Network

- Softmax function
  - To deal with the classification of multicategory

$$P(y_i|\mathbf{x}) = \frac{e^{\mathbf{x}^{\mathrm{T}}\mathbf{w}}}{\sum_{l=1}^{L} e^{\mathbf{x}^{\mathrm{T}}\mathbf{w}_l}}$$

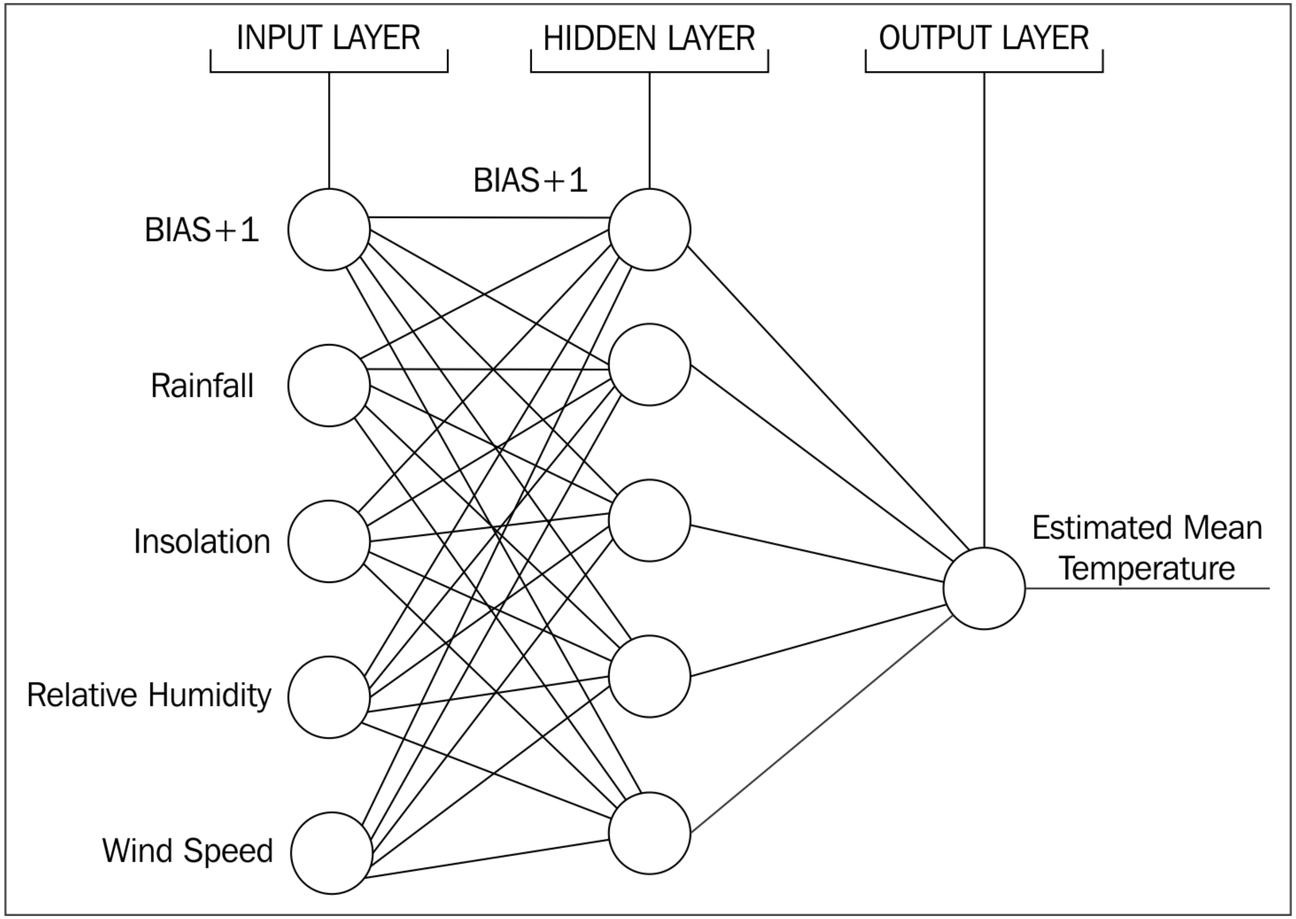
$$y = \arg\max_{y_i} P(y_i|\mathbf{x})$$

Artificial Neural Network

- Deep neural network, DNN
  - Multiple hidden layers
  - The number of hidden layers ≥ 1
  - Also called deep learning neural network
  - R. Dechter, "**Learning while searching in constraint-satisfaction problems**," University of California, Computer Science Department, Cognitive Systems Laboratory, 1986.

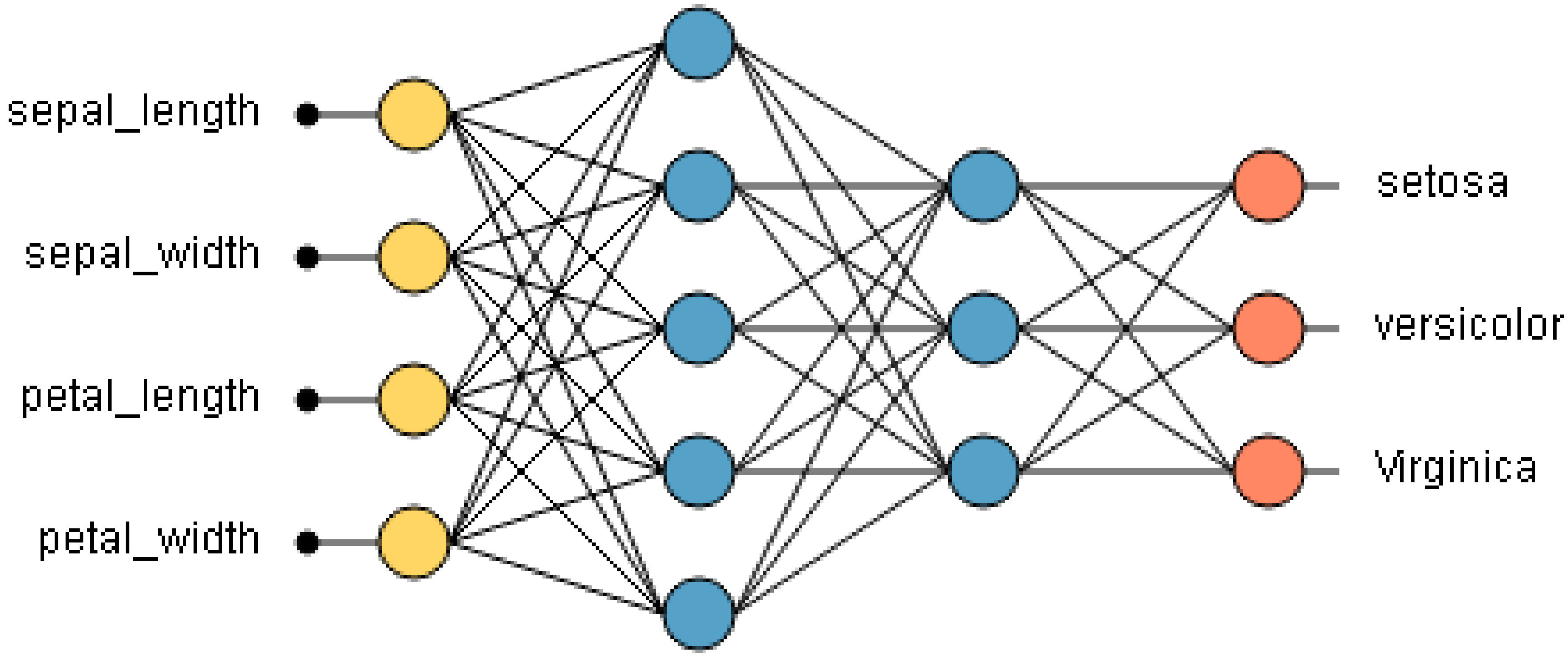
Artificial Neural Network

- Example: Forecasting weather via neural network
  - Fabio Soares and Alan Souza, "Neural Network Programming with Java", Packt, 2016.



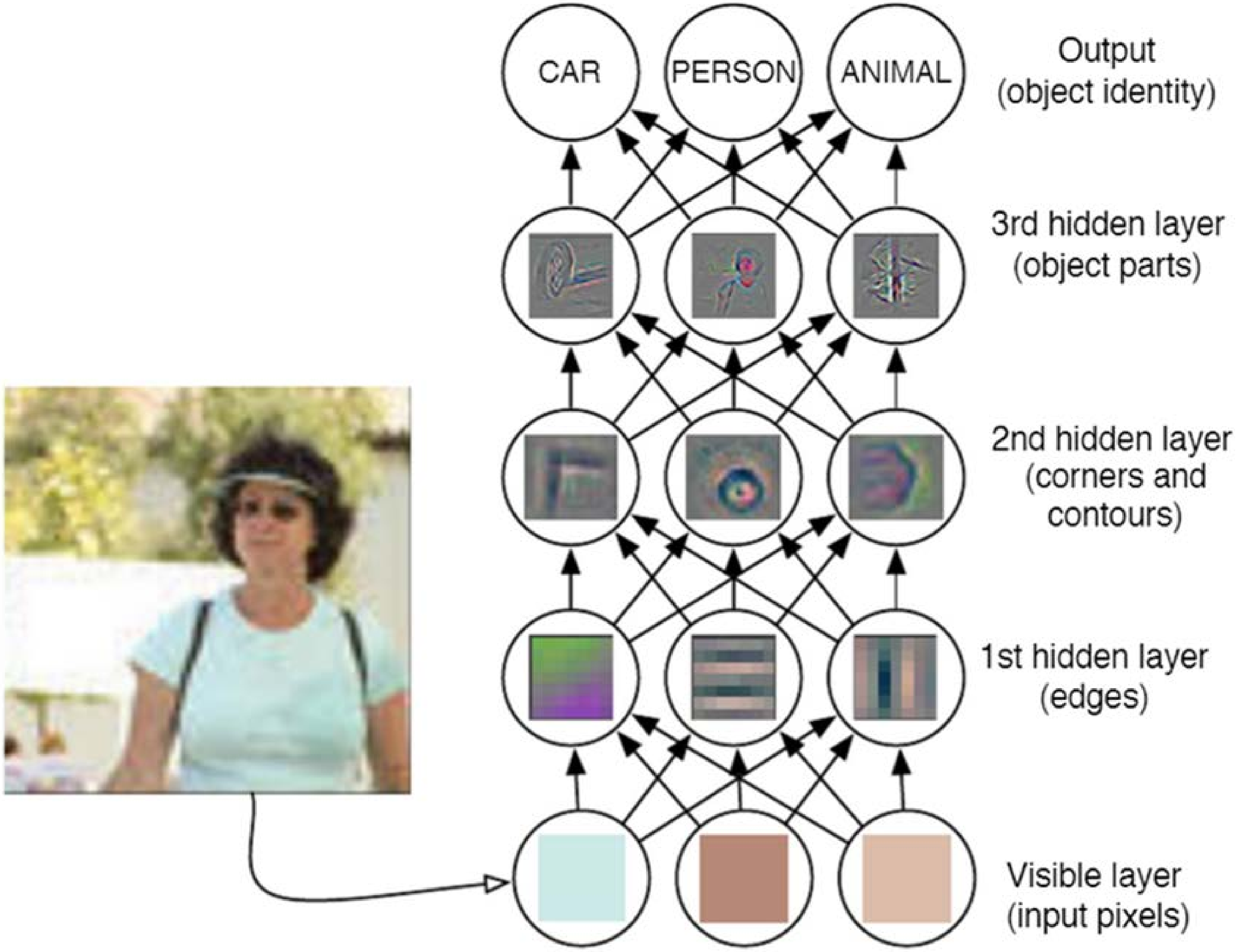
Artificial Neural Network

- Example: Iris flowers
  - Roberto Lopez, "Introduction to neural networks, " Neural Designer.



Artificial Neural Network

- Example: Object recognizing from images
  - I. Goodfellow, Y. Bengio, and A. Courville, "**Deep Learning**," *MIT Press*, 2016.

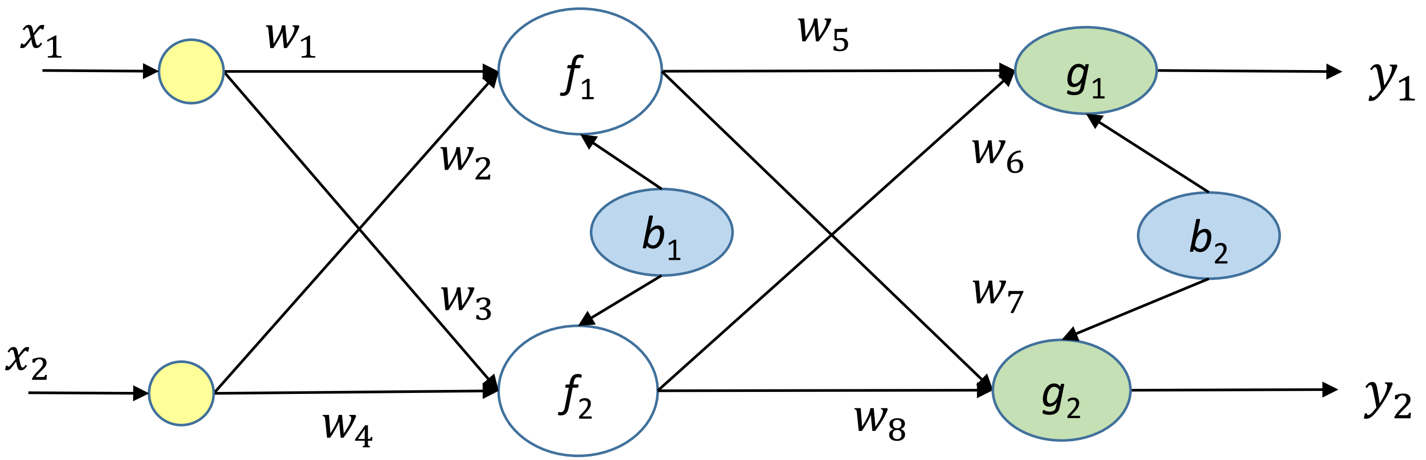


Backpropagation

- How to decide all weights in a neural network?
- We have a training dataset and each training instance has a target result.
- According the errors of the target results with the outputs of neural network to adjust the weights from the last layer to the first layer.

Backpropagation

- Error calculation



$$y_1 = g_1(f_1 w_5 + f_2 w_6 + b_2)$$

$$= g_1(\,(x_1 w_1 + x_2 w_2 + b_1) w_5 + (x_1 w_3 + x_2 w_4 + b_1) w_6 + b_2)$$

$$E = \frac{1}{2} \sum_{i=1}^{2} (t_i - y_i)^2 = \sum_{i=1}^{2} E(t_i, y_i)$$

Backpropagation

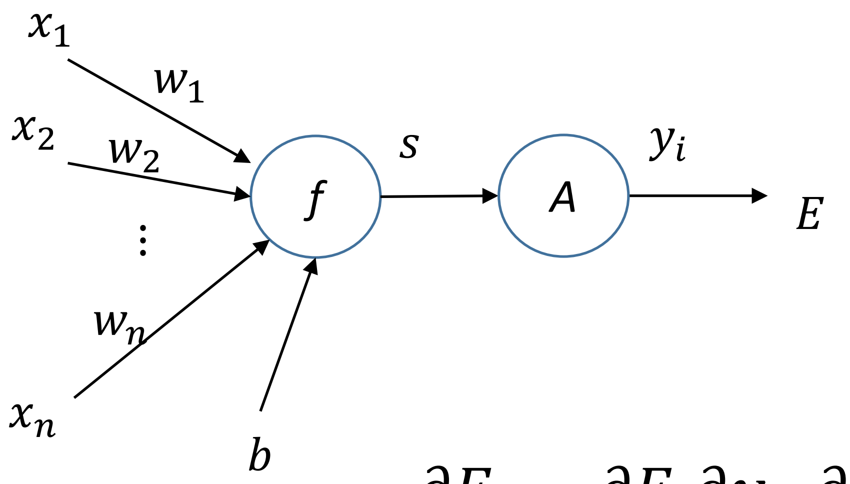
- From the last layer to the first layer

1. Estimate the partial derivative of E with respect to w_i.
2. Then apply gradient descent to update the weight w_i
3. Repeat step 1.

$$\frac{\partial E}{\partial w_5} \Rightarrow \text{update } w_5 \Rightarrow \frac{\partial E}{\partial w_6} \Rightarrow \text{update } w_6 \Rightarrow$$
$$\frac{\partial E}{\partial w_7} \Rightarrow \text{update } w_7 \Rightarrow \frac{\partial E}{\partial w_8} \Rightarrow \text{update } w_8 \Rightarrow$$
$$\frac{\partial E}{\partial w_1} \Rightarrow \text{update } w_1 \Rightarrow \frac{\partial E}{\partial w_2} \Rightarrow \text{update } w_2 \Rightarrow$$
$$\frac{\partial E}{\partial w_3} \Rightarrow \text{update } w_3 \Rightarrow \frac{\partial E}{\partial w_4} \Rightarrow \text{update } w_4$$

Backpropagation

- For the output layer



$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial s}\frac{\partial s}{\partial w_j}$$

- Applying the gradient descent

$$\Delta w_j = -\frac{\partial E}{\partial w_j}$$

Backpropagation

- Since

$$\frac{\partial E}{\partial y_i} = \frac{\partial(\frac{1}{2}\sum_{i=1}^{h}(t_i - y_i)^2)}{\partial y_i} = \frac{\partial E(t_i, y_i)}{\partial y_i} = -(t_i - y_i) = y_i - t_i$$

- Assuming that *A* is a logistic function = *L*(s)

$$\frac{\partial y_i}{\partial s} = \frac{\partial L(s)}{\partial s} = L(s)(1 - L(s))$$

- And

$$\frac{\partial s}{\partial w_j} = \frac{\partial(\sum_{i=1}^{n} x_i w_i + b)}{\partial w_j} = x_j$$

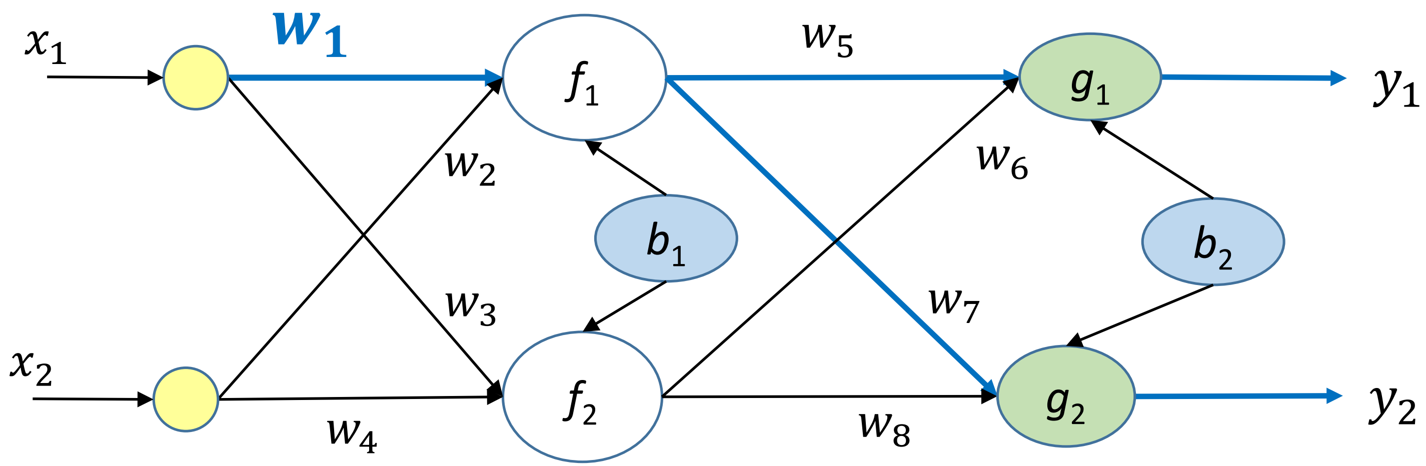
Backpropagation

- Therefore,

$$\Delta w_j = -\frac{\partial E}{\partial w_j} = -(y_i - t_i)L(s)(1 - L(s))x_j$$

Backpropagation

- For hidden layers



$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial f_1}\frac{\partial f_1}{\partial w_1}$$

Chain rule

$$= \left(\frac{\partial E(t_1, y_1)}{\partial f_1} + \frac{\partial E(t_2, y_2)}{\partial f_1}\right)\frac{\partial f_1}{\partial w_1}$$

Backpropagation

- For hidden layers

$$\frac{\partial E(t_1, y_1)}{\partial f_1} = \frac{\partial s_1}{\partial f_1}\frac{\partial E(t_1, y_1)}{\partial s_1}$$

$$= \frac{\partial (f_1 w_5 + f_2 w_6 + b_2)}{\partial f_1}\frac{\partial E(t_1, y_1)}{\partial s_1}$$

$$= w_5 \frac{\partial E(t_1, y_1)}{\partial y_1}\frac{\partial y_1}{\partial s_1}$$

These are calculated before

Backpropagation

- A neuron in hidden layer may has the logistic activation

$$\frac{\partial f_1}{\partial w_1} = \frac{\partial f_1}{\partial r_1}\frac{\partial r_1}{\partial w_1} = L(r_1)(1 - L(r_1))x_1$$

- Therefore,

$$\Delta w_1 = -\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial f_1}\frac{\partial f_1}{\partial w_1}$$

$$= (\frac{\partial E(t_1, y_1)}{\partial f_1} + \frac{\partial E(t_2, y_2)}{\partial f_1})L(r_1)(1 - L(r_1))x_1$$

  - where

$$\frac{\partial E(t_1, y_1)}{\partial f_1} = w_5 \frac{\partial E(t_1, y_1\ )}{\partial y_1}\frac{\partial y_1}{\partial s_1}$$

$$\frac{\partial E(t_2, y_2)}{\partial f_1} = w_7 \frac{\partial E(t_2, y_2\ )}{\partial y_2}\frac{\partial y_2}{\partial s_1}$$

Backpropagation

- Vanishing gradient problem
  - In the final layer *k*, $\frac{\partial E}{\partial w_j^k}$ is large ➔ OK!
  - In (*k* – 1)th layer, $\frac{\partial E}{\partial w_j^{k-1}}$ is smaller than $\frac{\partial E}{\partial w_j^k}$ ➔ Still OK!
  - …
  - In the first layer, $\frac{\partial E}{\partial w_j^1}$ is closed to zero ➔ Not OK!
  - The sigmoid activation may cause this problem

$$0 \le \frac{\partial L(s)}{\partial s} = L(s)\big(1 - L(s)\big) \le 0.25$$

  - Using ReLU can solve this problem

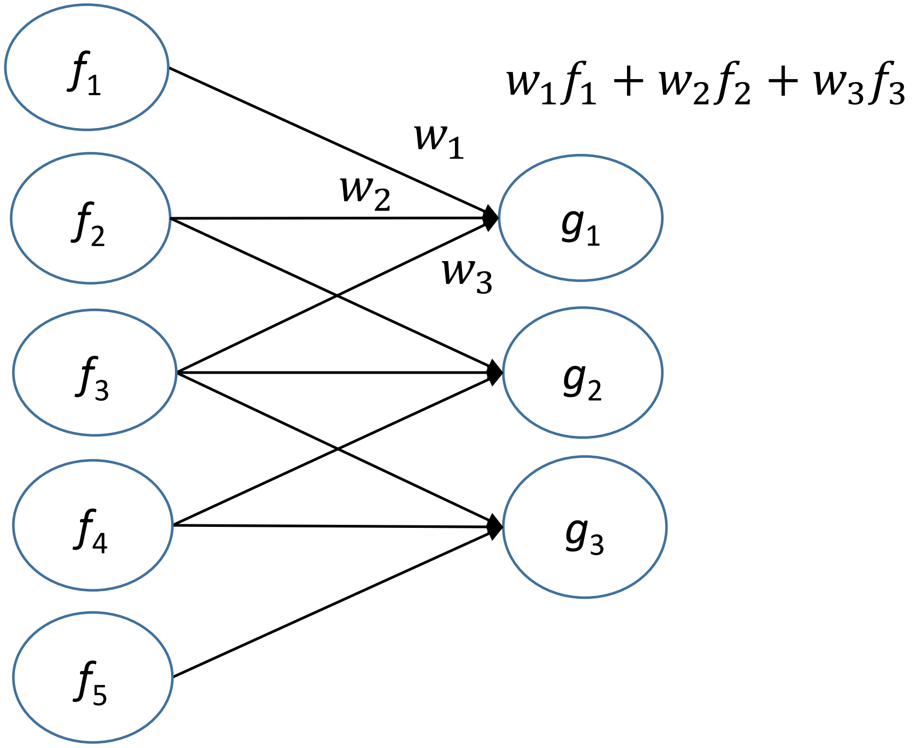
$$\frac{\partial R(s)}{\partial s} = \begin{cases} 1 & s \ge 0 \\ 0 & s < 0 \end{cases}$$

Convolutional Neural Network, CNN

- Fully-connected neural network
  - In a fully connected layer, each neuron is connected to every neuron in the previous layer, and each connection has it's own weight.
  - General purpose connection pattern
  - No assumptions about the features
  - High consumption of memory (weights) and computation (connections).

Convolutional Neural Network, CNN

- In a convolutional layer
  - Each neuron only connected form a set of related neurons in the previous layer
  - Locality relation
  - Shared weights
    - Each neuron using the same weights for the input

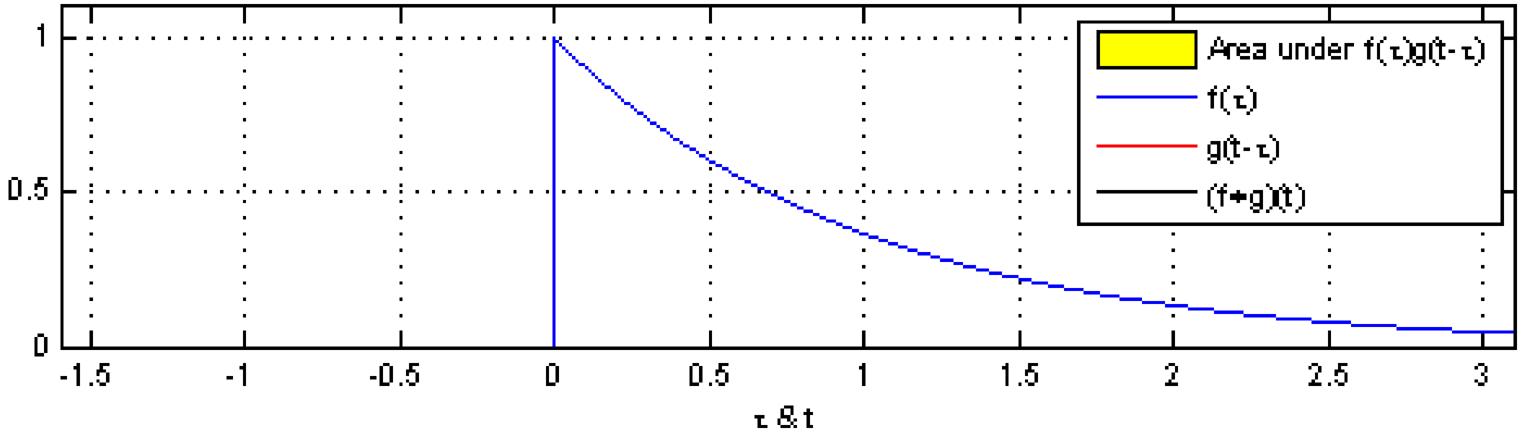


Convolutional Neural Network, CNN

- Convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - t)d\tau$$

  - *g* is called **convolution kernel, operator, or mask.**



Convolutional Neural Network, CNN

- Discrete convolution

$$(f * g)[n] = \sum_{m=-M}^{M} f[n-m]g[m]$$

  - where *M* is the index range of *g*
  - example:
    - $f\,[-3: 3] = \{1, 9, 0, 5, 2, 4, 3\}$
    - $g\,[-1: 1] = \{-2, 0, 2\}$
    - $(f * g)[2] = f\,[2 - (-1)]g[-1] + f\,[2 - 0]g[0] + f\,[2 - 1]g\,[1]$

$$= f\,[3]\mathrm{g}\,[-1] + f\,[2]g\,[0] + f\,[1]g\,[1]$$

$$= 3(-2) + 4(0) + 2(2)$$

$$= -2$$

Convolutional Neural Network, CNN

- 2D discrete convolution

$$(f * g)[m][n] = \sum_{y=-H}^{H} \sum_{x=-W}^{W} f[m-y][n-x]g[y][x]$$

f[-3:2][-3:2]

1	4	5	8	2	1
0	3	1	5	7	2
1	2	1	4	7	1
2	2	0	3	6	2
0	1	7	3	5	3
1	3	4	1	2	4

f[-3][-3]

g[-1:1][-1:1]

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

(f *g)[0][0] = -45

1(3)	5(0)	7(-3)
1(2)	4(0)	7(-2)
0(1)	3(0)	6(-1)

Convolutional Neural Network, CNN

- Zero padding

$f[-3:2][-3:2] \rightarrow f[-4:3][-4:3]$

0	0	0	0	0	0	0	0
0	1	4	5	8	2	1	0
0	0	3	1	5	7	2	0
0	1	2	1	4	7	1	0
0	2	2	0	3	6	2	0
0	0	1	7	3	5	3	0
0	1	3	4	1	2	4	0
0	0	0	0	0	0	0	0

$g[-1:1][-1:1]$

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

$(f * g)[2][2] = 11$

0(3)	0(0)	0(-3)
2(2)	1(0)	0(-2)
7(1)	2(0)	0(-1)

Convolutional Neural Network, CNN

- Commonly-used 2D kernels
  - Mean $g(x,y) = \frac{1}{(2H+1)\times(2W+1)}$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

  - Gaussian $g(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Convolutional Neural Network, CNN

- Commonly-used 2D kernels
  - X derivative, $g_x =$

a	0	-a
b	0	-b
a	0	-a

a = b = 1

1	0	-1
1	0	-1
1	0	-1

  - Y derivative , $g_y =$

-a	-b	-a
0	0	0
a	b	a

a = 2, b = 5

-2	-5	-2
0	0	0
2	5	2

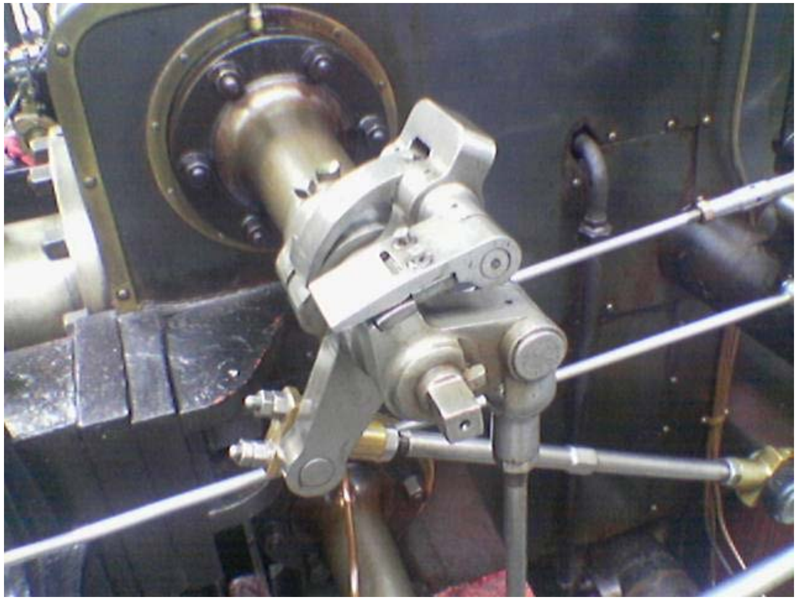
  - Edge detection: $|(f \ * g_x), (f \ * g_y)|$
    - Sobel operator

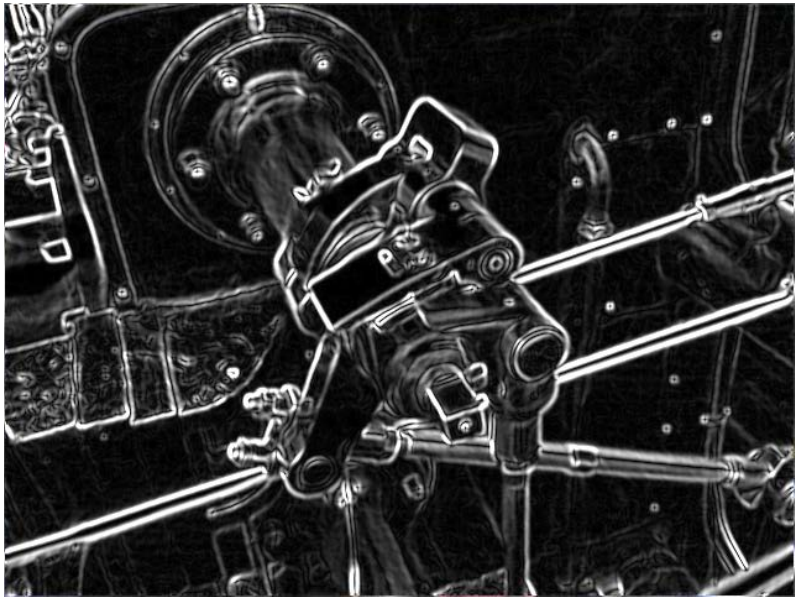
Convolutional Neural Network, CNN

- Convolution in image processing
  - Filtering
  - Gaussian filtering

Convolutional Neural Network, CNN

- Convolution in image processing
  - Edge detection (Sobel operator)





Convolutional Neural Network, CNN

- Max pooling
  - Finding the maximum from a masked area

f[-3:2][-3:2]

1	4	5	8	2	1
0	3	1	5	7	2
1	2	1	4	7	1
2	2	0	3	6	2
0	1	7	3	5	3
1	3	4	1	2	4

7

6

Convolutional Neural Network, CNN

- Stride
  - The step of the convolution operation
  - It affects the output size

Input size: 6 x 6

Kernel size: 3 x 3 (max pooling)

Stride: 1

1	4	5	8	2	1
0	3	1	5	7	2
1	2	1	4	7	1
2	2	0	3	6	2
0	1	7	3	5	3
1	3	4	1	2	4

Output size: 4 x 4

5	8	8	8
3	5	7	7
7	7	7	7
7	7	7	6

Convolutional Neural Network, CNN

- Stride

Input size: 6 x 6
Kernel size: 3 x 3 (max pooling)
Stride: 3

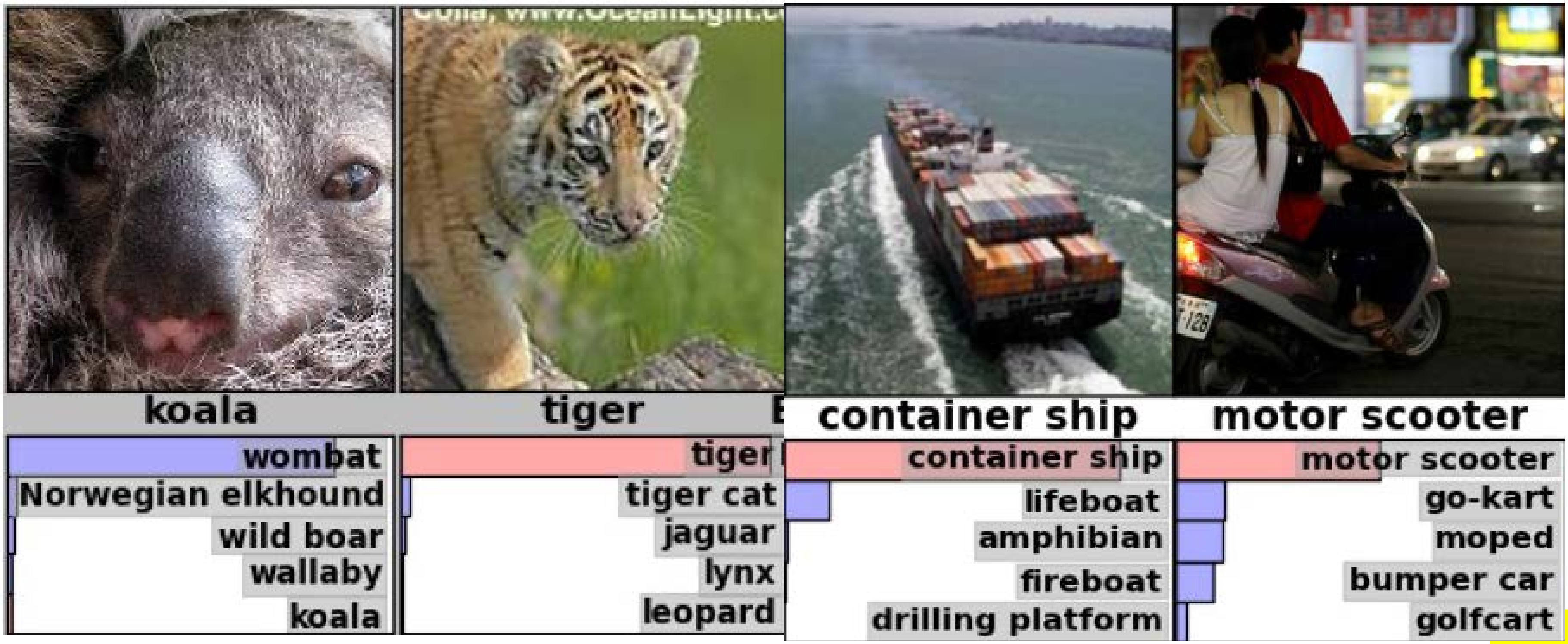
1	4	5	8	2	1
0	3	1	5	7	2
1	2	1	4	7	1
2	2	0	3	6	2
0	1	7	3	5	3
1	3	4	1	2	4

Output size: 2 x 2

5	8
7	6

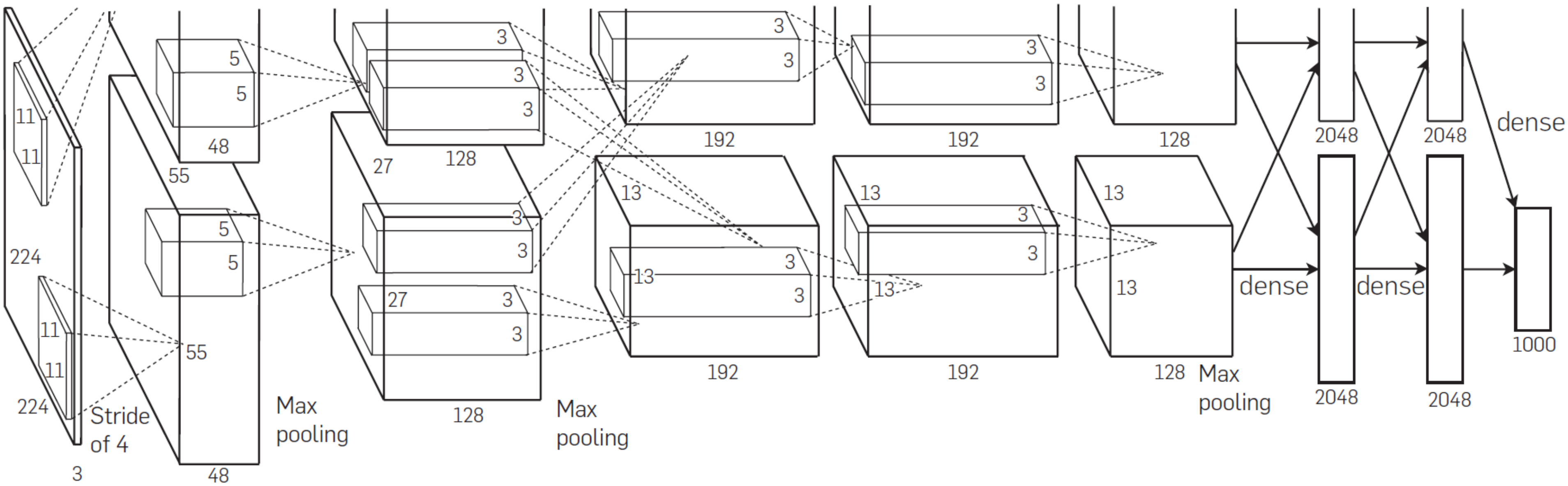
Convolutional Neural Network, CNN

- Example 1: Image classification
  - Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. **ImageNet classification with deep convolutional neural networks**. *Commun. ACM* 60, 6 (May 2017), 84-90.
  - Using CNN to classify the 1.2 million high-resolution images



Convolutional Neural Network, CNN

- Example 1: Image classification



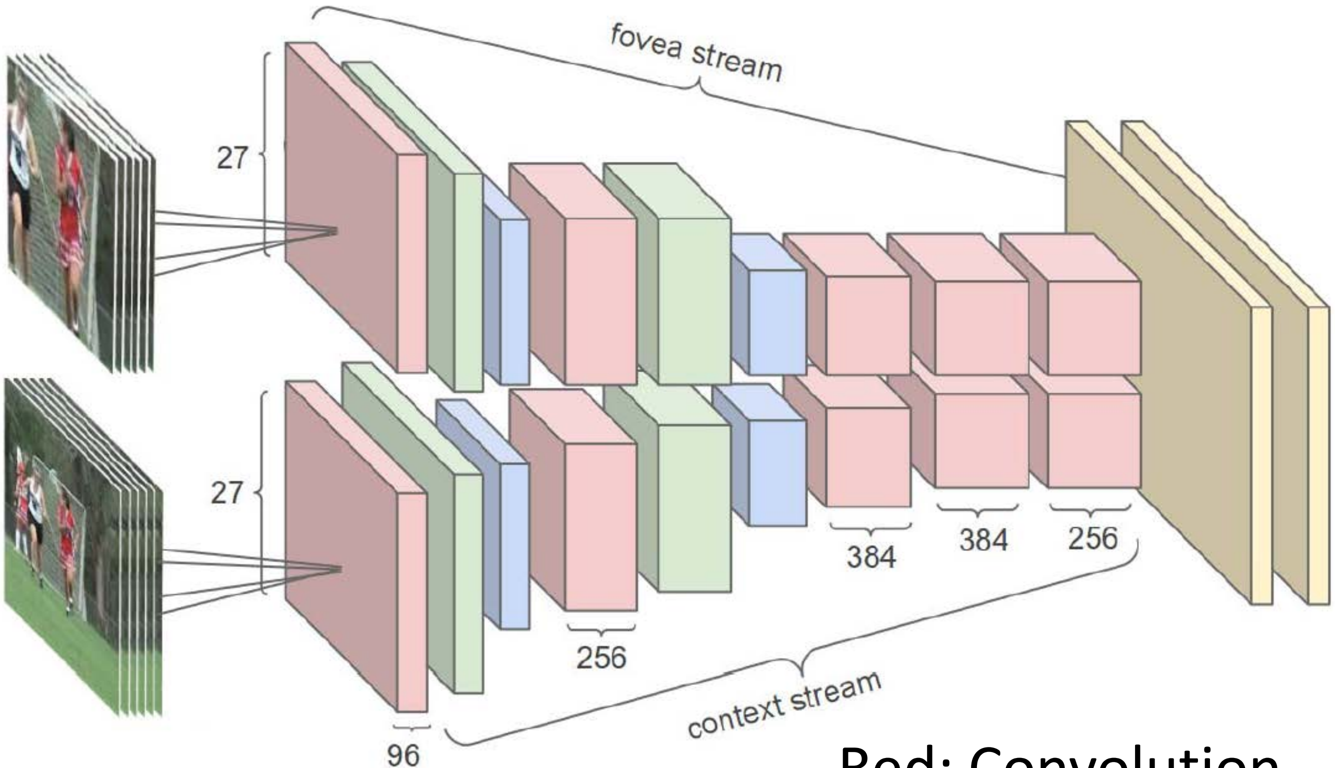
- In the first convolution layer
  - Number of kernels: 96
  - Kernel size: 11 x 11
  - Stride: 4 ➔(0, 4, 8, …, 220)
  - Zero padding
  - ➔ output size: 55 x 55 x 96

Convolutional Neural Network, CNN

- Example 2: Video classification
  - A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei, "**Large-Scale Video Classification with Convolutional Neural Networks**," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, 2014, pp. 1725-1732.
  - 1 million videos ➔ 487 categories

Convolutional Neural Network, CNN

- Example 2: Video classification



Red: Convolution
Green: Normalization
Blue: Pooling
Yellow: Fully connected layers

Convolutional Neural Network, CNN

- Example 2: Video classification
  - Input size: 178 x 178 pixels per frame
  - Input frames are fed into two separate streams
    - Context stream that models low-resolution image
      - Downsampled to 89 x 89 pixels
    - Fovea stream that processes high-resolution center crop.
      - the middle portion of a frame.
  - Both streams consist of alternating convolution (red), normalization(green), and pooling layers(blue).
  - Both streams converge to two fully connected layers (yellow).

Convolutional Neural Network, CNN

- Example 3: Speech recognition
  - O. Abdel-Hamid, A. r. Mohamed, H. Jiang, L. Deng, G. Penn and D. Yu, "**Convolutional Neural Networks for Speech Recognition**," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533-1545, Oct. 2014.

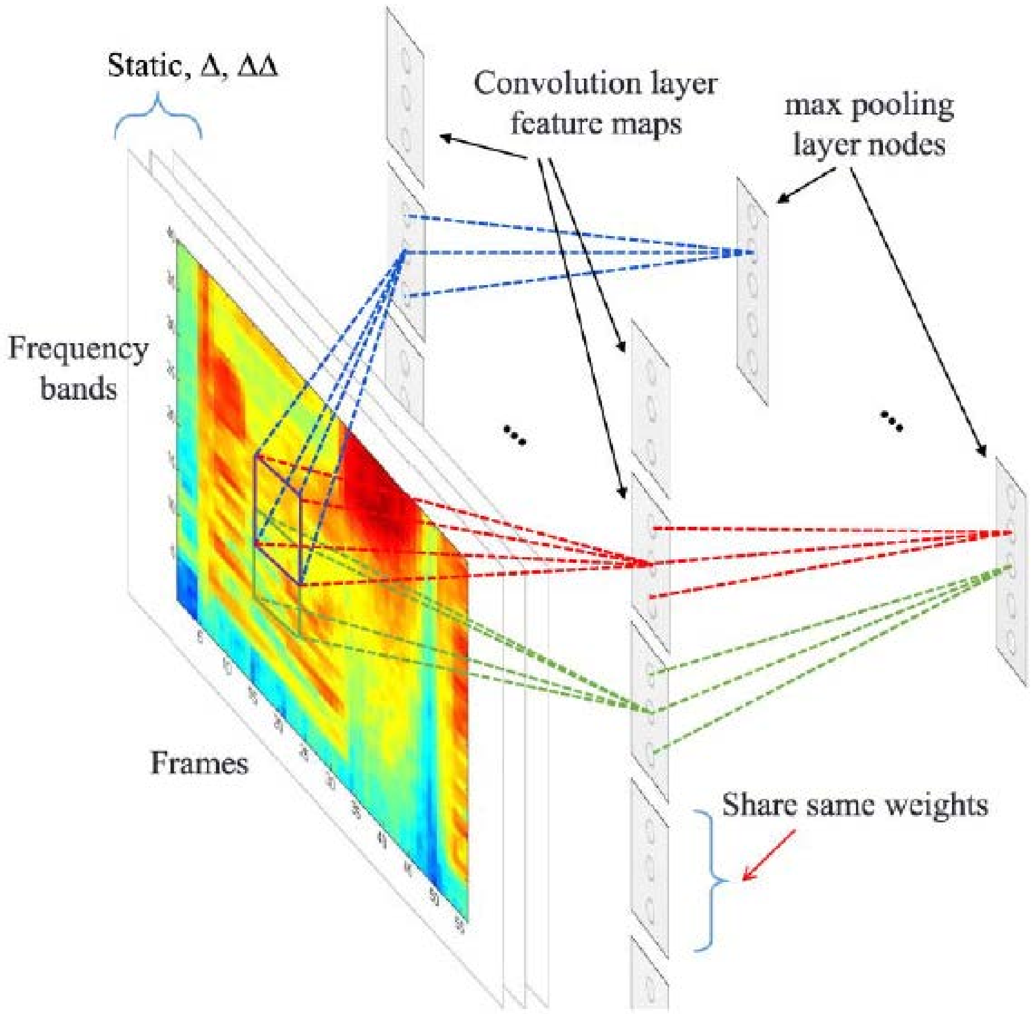
	No PT	With PT
DNN	37.1%	35.4%
CNN	34.2%	33.4%

WER: Word error rate

PT: Pretraining

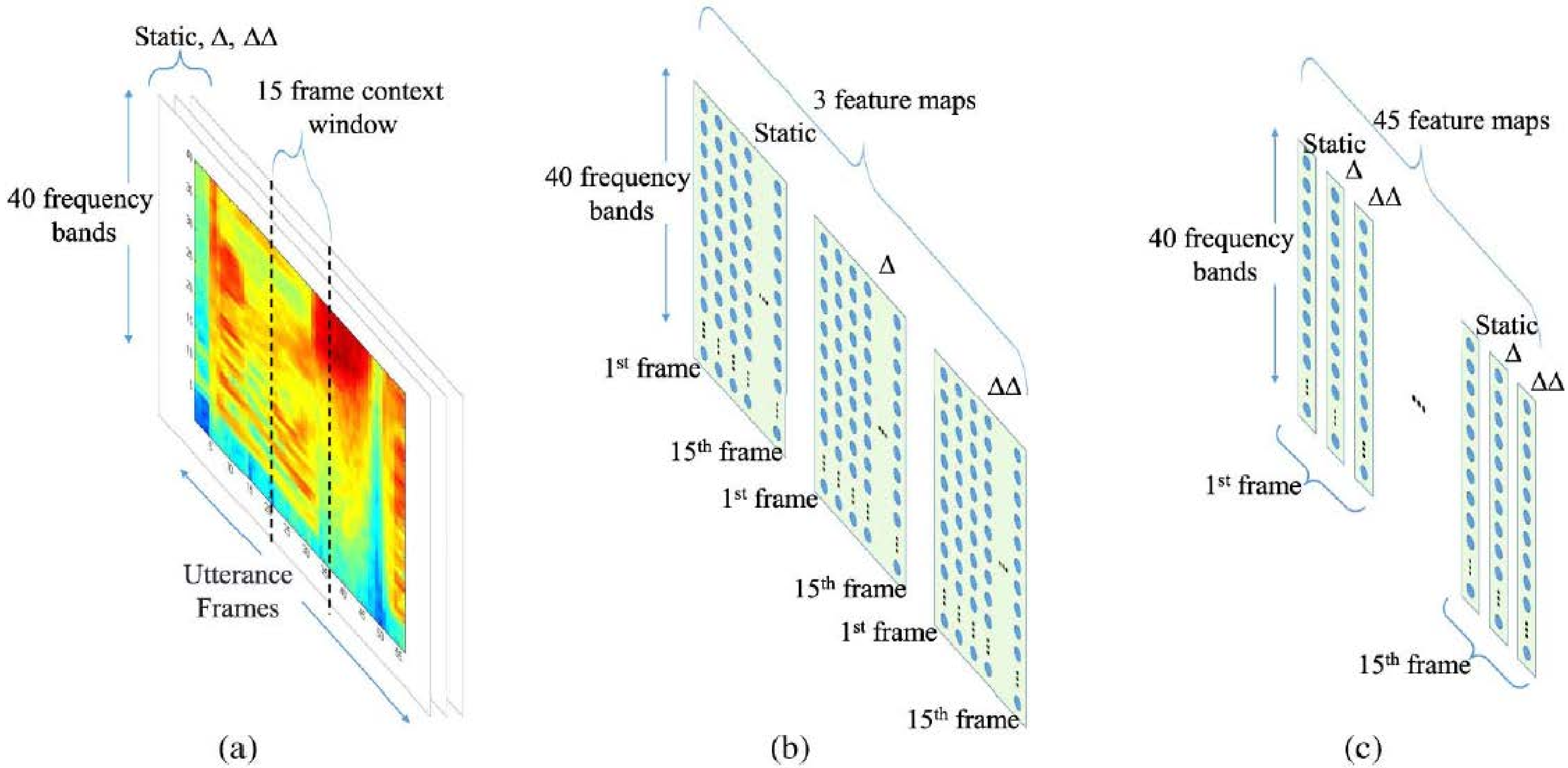
Convolutional Neural Network, CNN

- Example 3: Speech recognition
  - Structure



Convolutional Neural Network, CNN

- Example 3: Speech recognition
  - Input data



Fourier Transform

- 1D FT

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i u x} dx$$

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{2\pi i u x} du$$

- Discrete 1D FT
  - For *N* numbers:

$$F[u] = \sum_{x=0}^{N-1} f[x] e^{-2\pi i \frac{ux}{n}}$$

$$f[x] = \sum_{u=0}^{N-1} F[u] e^{2\pi i \frac{ux}{n}}$$

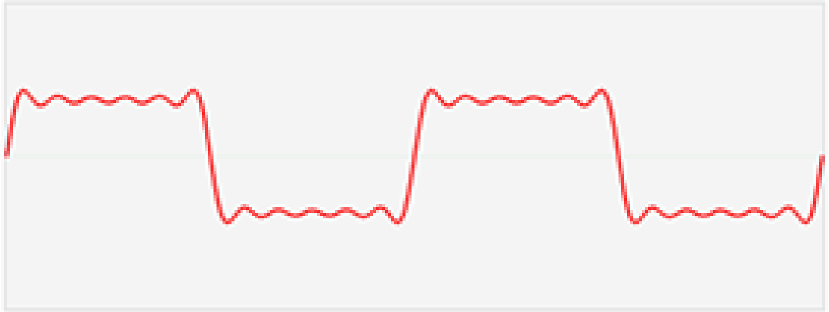
$i = \sqrt{-1}$

Euler's formula:
$e^{ix} = \cos x + i \sin x$

Euler's identity:
$e^{i\pi} + 1 = 0$

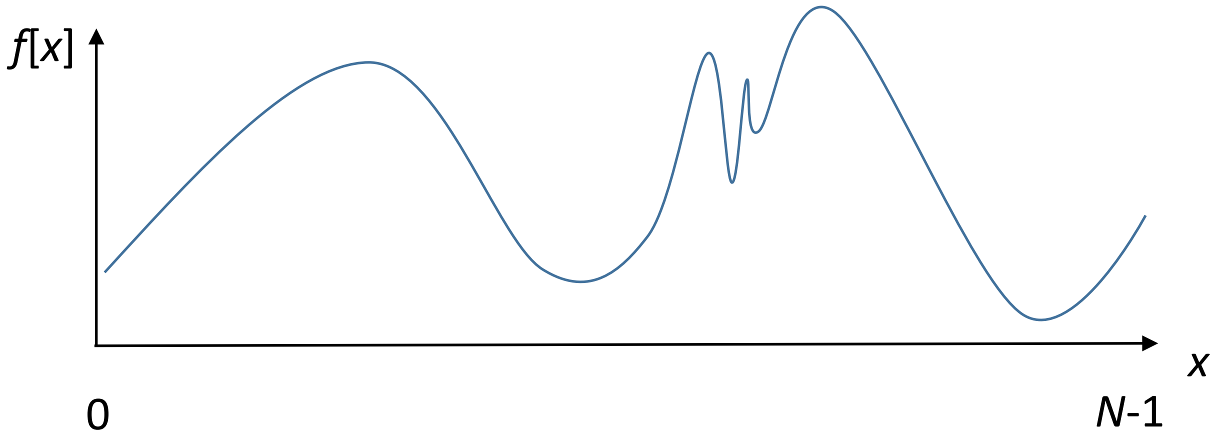
Fourier Transform

- The purpose of FT
  - Any periodic signal can be decomposed by many sine and cosine waves with different frequencies and amplitudes.
  - The frequency is called **angular frequency**
    - The unit is **radian** rather than Hz
    - The range of angular freq. is [-$\boldsymbol{\pi}$, $\boldsymbol{\pi}$]



Fourier Transform

- Time domain $f[x]$



- Frequency domain $F[u]$