LeNet

2019.06.21

在簡報之前…

- 感謝PyTorch New Taipei及Taoyuan讀書會, Kevin 先生及吳政龍 先生讓我直接”剪貼”他們的簡報檔到本簡報檔
- PyTorch New Taipei讀書會 – Kevin 先生
  - https://drive.google.com/file/d/1cgXbzt9wPS_NWhi0dELdayOSdoXRo_cd/view?usp=sharing
- PyTorch Taoyuan讀書會 – 吳政龍 先生
  - https://drive.google.com/open?id=1vYtARr9lieRlS3oAofJOpgnWN8-6P40l

LeNet Structure Code Example

```
from keras.models import Sequential
from keras import models, layers
import keras

#Instantiate an empty model
model = Sequential()

# C1 Convolutional Layer
model.add(layers.Conv2D(6, kernel_size=(5, 5), strides=(1, 1), activation='tanh', input_shape=(28,28,1),
padding="same"))

# S2 Pooling Layer
model.add(layers.AveragePooling2D(pool_size=(2, 2), strides=(1, 1), padding='valid'))

# C3 Convolutional Layer
model.add(layers.Conv2D(16, kernel_size=(5, 5), strides=(1, 1), activation='tanh', padding='valid'))
```

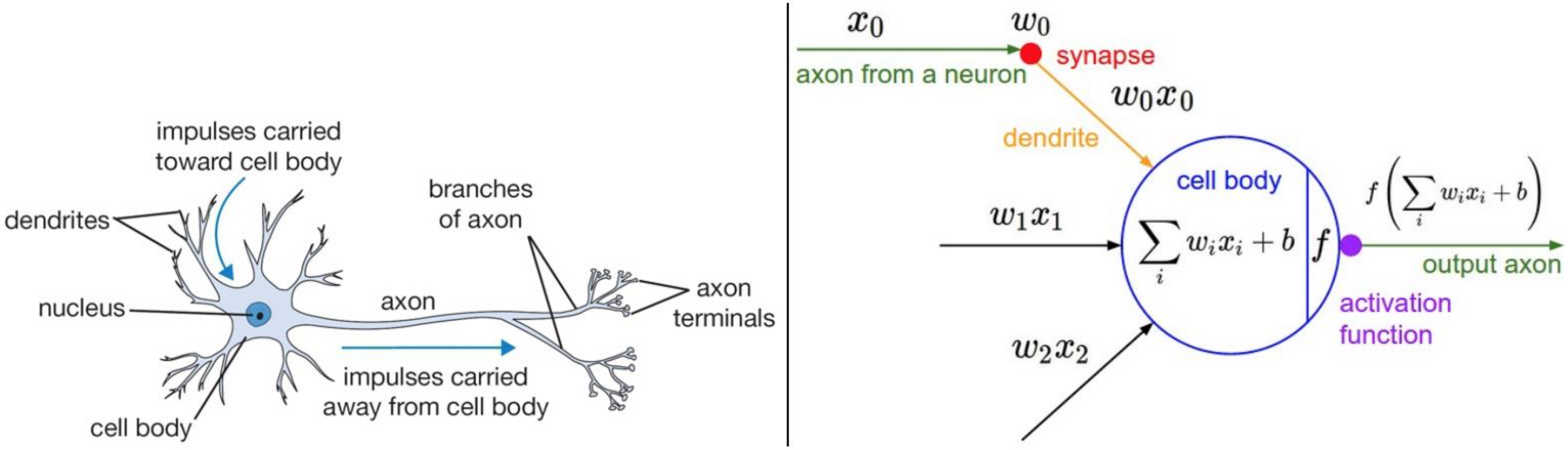
https://engmrk.com/lenet-5-a-classic-cnn-architecture/

LeNet Structure Code Example

```
# S4 Pooling Layer
model.add(layers.AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='valid'))

# C5 Fully Connected Convolutional Layer
model.add(layers.Conv2D(120, kernel_size=(5, 5), strides=(1, 1), activation='tanh', padding='valid'))
#Flatten the CNN output so that we can connect it with fully connected layers
model.add(layers.Flatten())

# FC6 Fully Connected Layer
model.add(layers.Dense(84, activation='tanh'))

#Output Layer with softmax activation
model.add(layers.Dense(10, activation='softmax'))

# Compile the model
model.compile(loss=keras.losses.categorical_crossentropy, optimizer='SGD', metrics=["accuracy"])
```

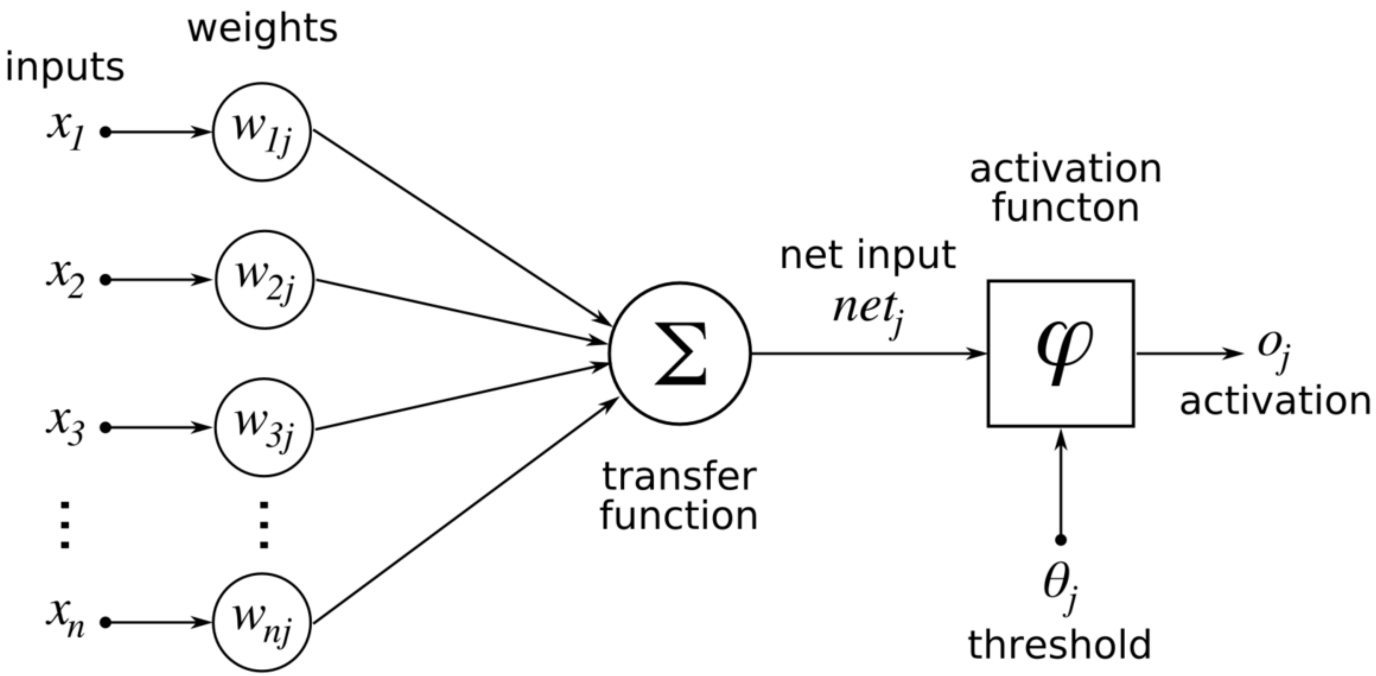
https://engmrk.com/lenet-5-a-classic-cnn-architecture/

Biological neuron & its mathematical model



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

https://isaacchanghau.github.io/post/activation_functions/



https://isaacchanghau.github.io/post/activation_functions/

Learning from Data

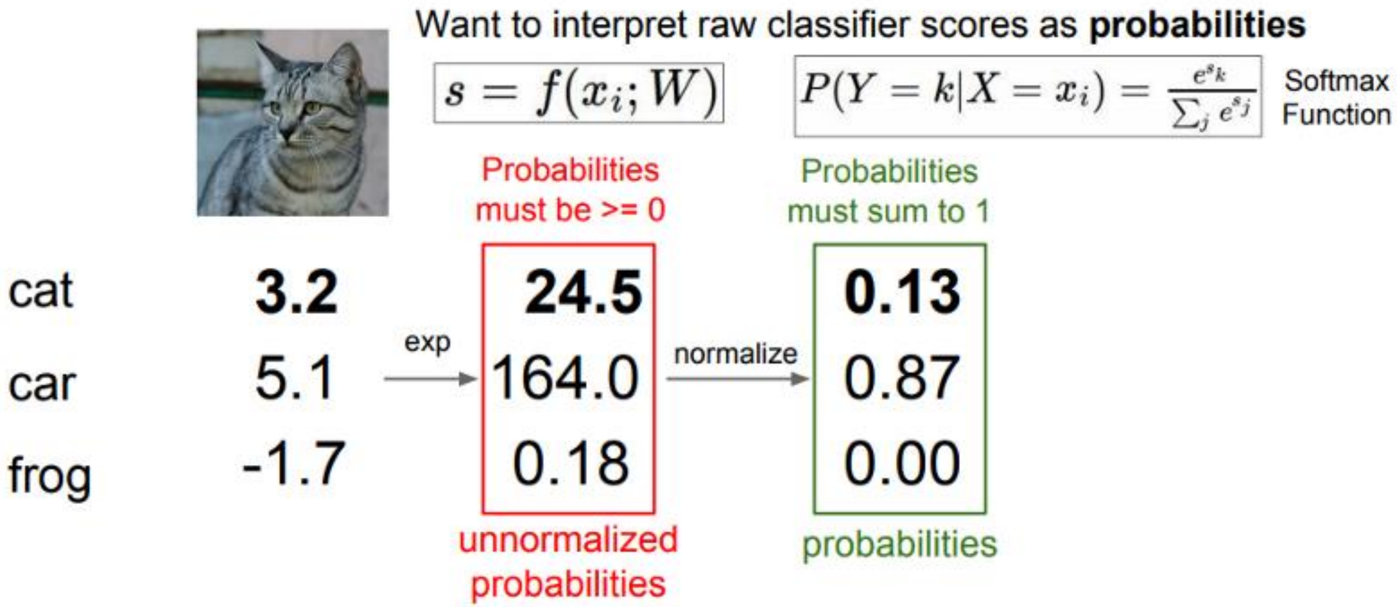
- $Y^p = F(Z^p, W)$
- Y^p = 分類標籤
- Z^p = 第p個特徵
- W = 所有可調整參數

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

source: Stanford cs231n_2018_lecture03

Learning from Data - **Loss Function**

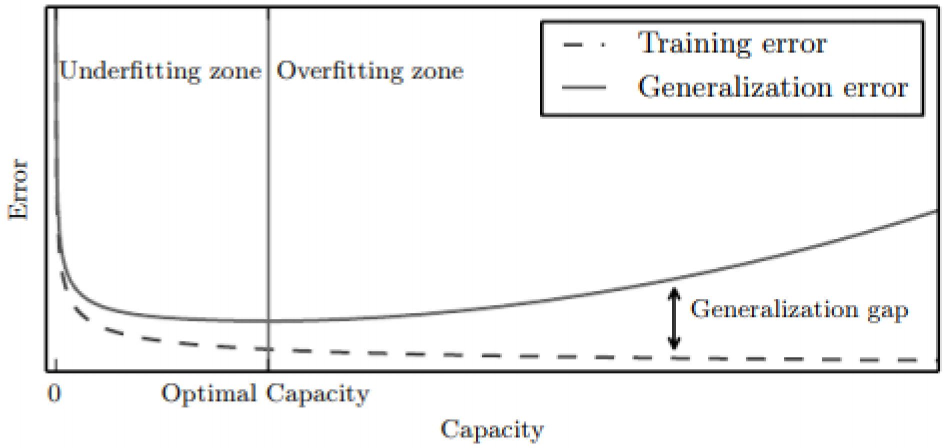
- $Y^p = F(Z^p, W)$
- $E^p = D(D^p, F(Z^p, W))$
  - 計算Y^p與D^p間的差距
- $E_{train} = \sum E^p / p$
- 調整W來獲得更小的E_{train}



Learning from Data

- 但其實Training不是這麼重要
- 重要的是Training出來之後的Testing表現才是重點

- $\mathrm{E}_{test} - E_{train} = k(\frac{h}{P})^{\alpha}$
  - P為訓練樣本數
  - h為訓練模型的複雜度
  - α介於0.5-1
  - k為一常數



Learning from Data

- Structural risk minimization
- $E_{train} + \beta H(W)$

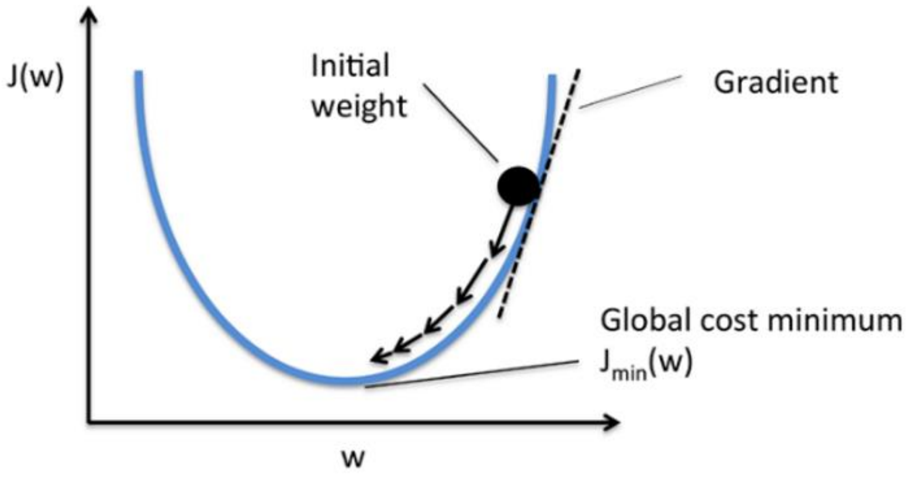
$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

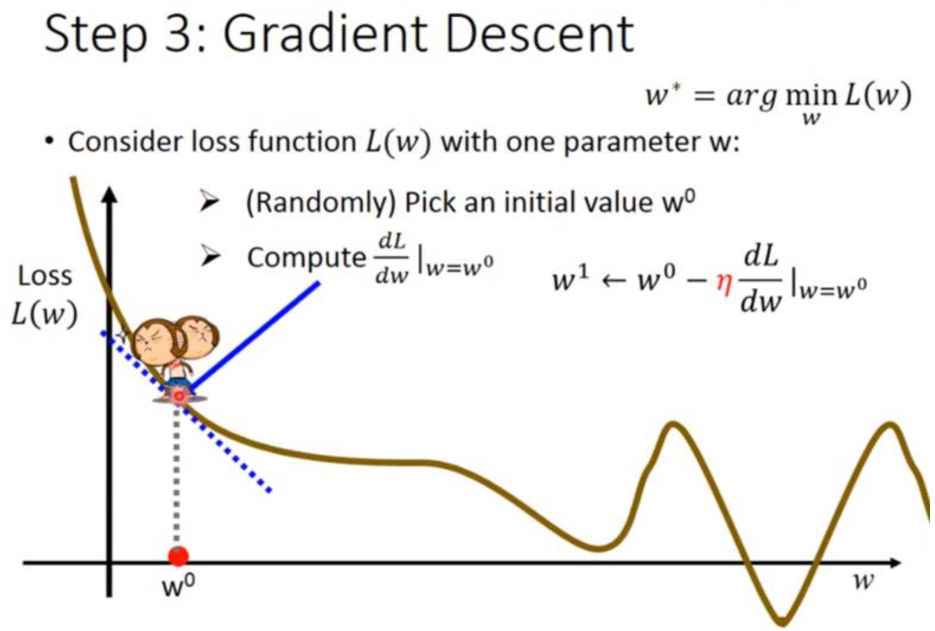
Gradient-Based Learning

- 論文的核心概念



Python Machine Learning by Sebastian Raschka

http://chico386.pixnet.net/al
bum/photo/171572850



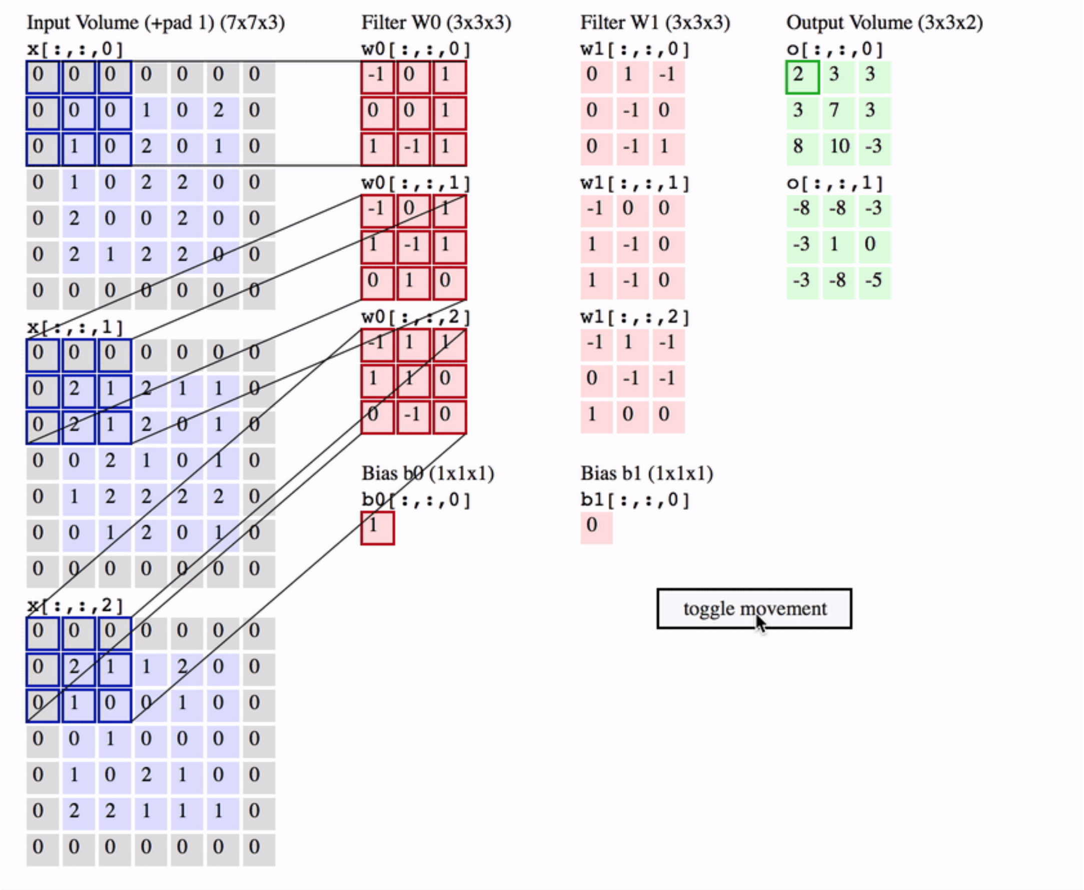
Back propagation

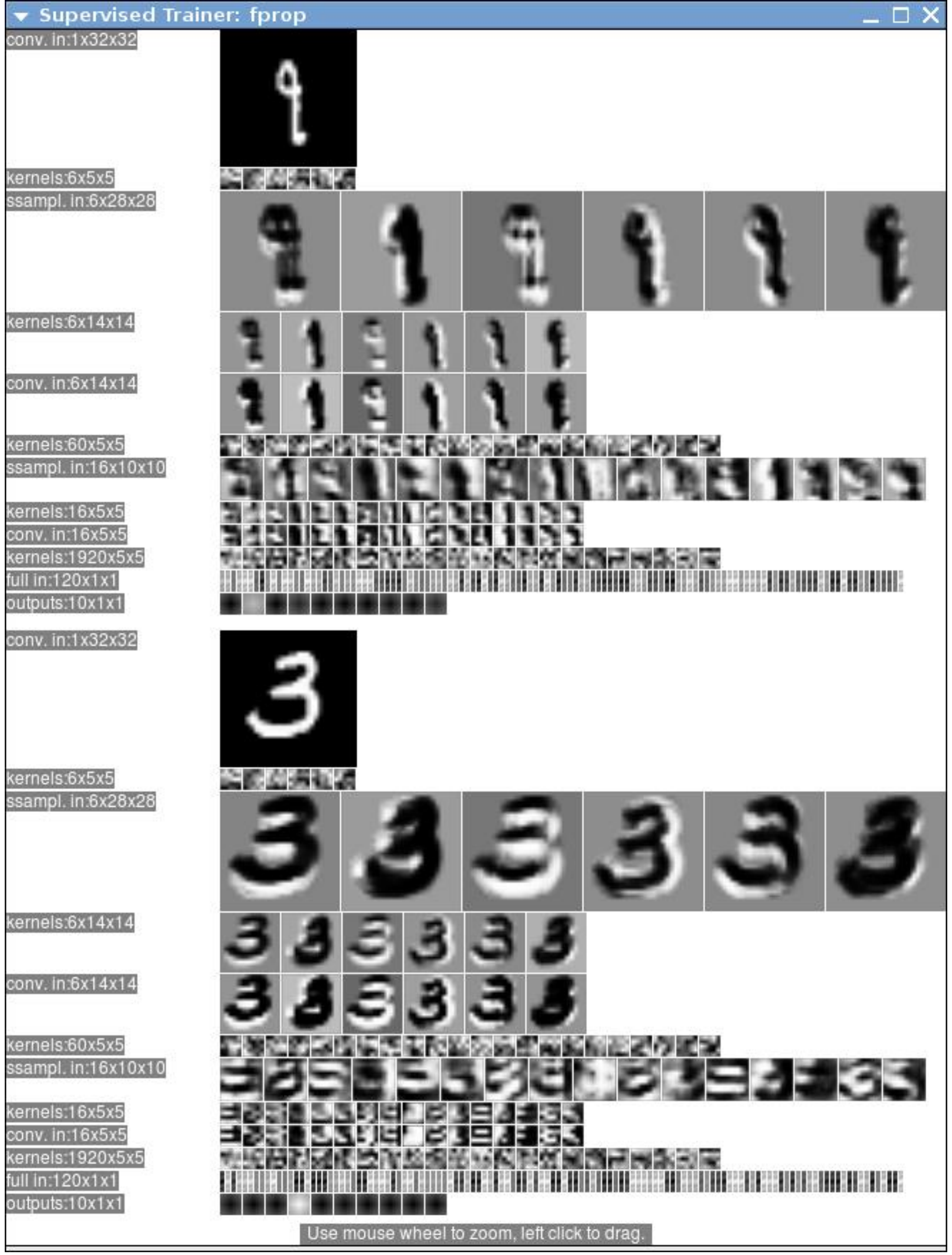
- 論文的核心公式
- $\mathrm{W}_k = W_{k-1} - \epsilon \frac{\partial E(W)}{\partial W}$
- $\mathrm{W}_k = W_{k-1} - \epsilon \frac{\partial E^{pk}(W)}{\partial W}$
  - SGD: Stochastic Gradient Descent
  - ϵ is learning rate(constant)

- 核心概念是微積分學到的Chain role
  - $\frac{d}{dx} f\big(g(x)\big) = f'\big(g(x)\big)g'(x)$

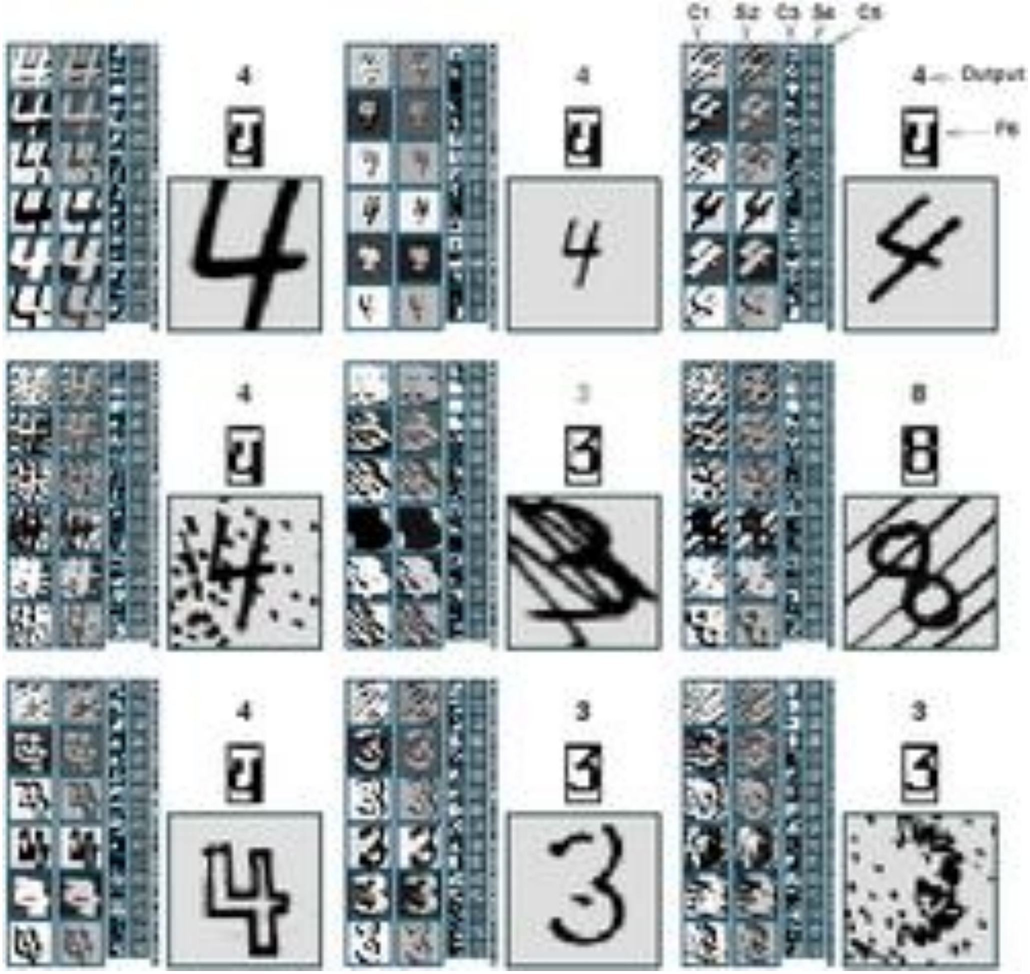
Convolutional Network

- Combine three architectural ideas to ensure some degree of shift, scale, and distortion invariance
  - Local Receptive Field
  - Shared Weight
  - Spatial or Temporal Subsampling
  - extract oriented edges, endpoints, corners





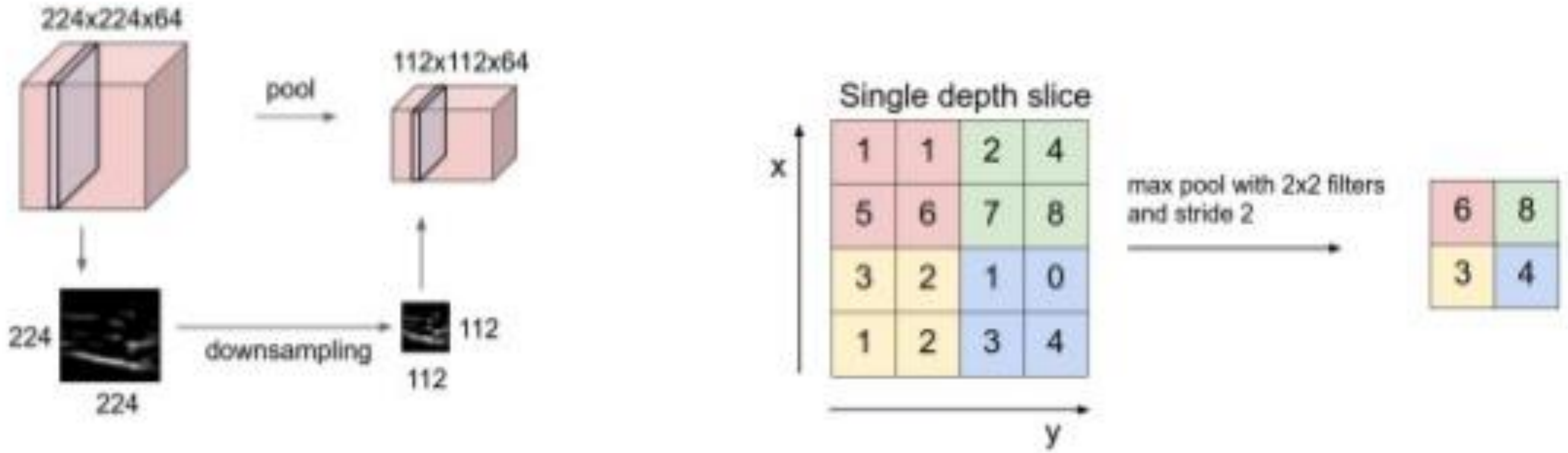
http://eblearn.sourceforge.net/old/tutorials/libeblearn/



https://www.slideshare.net/perone/deep-learning-convolutional-neural-networks

Pooling Layer

There are different types of pooling, the most used is the max-pooling and average pooling:

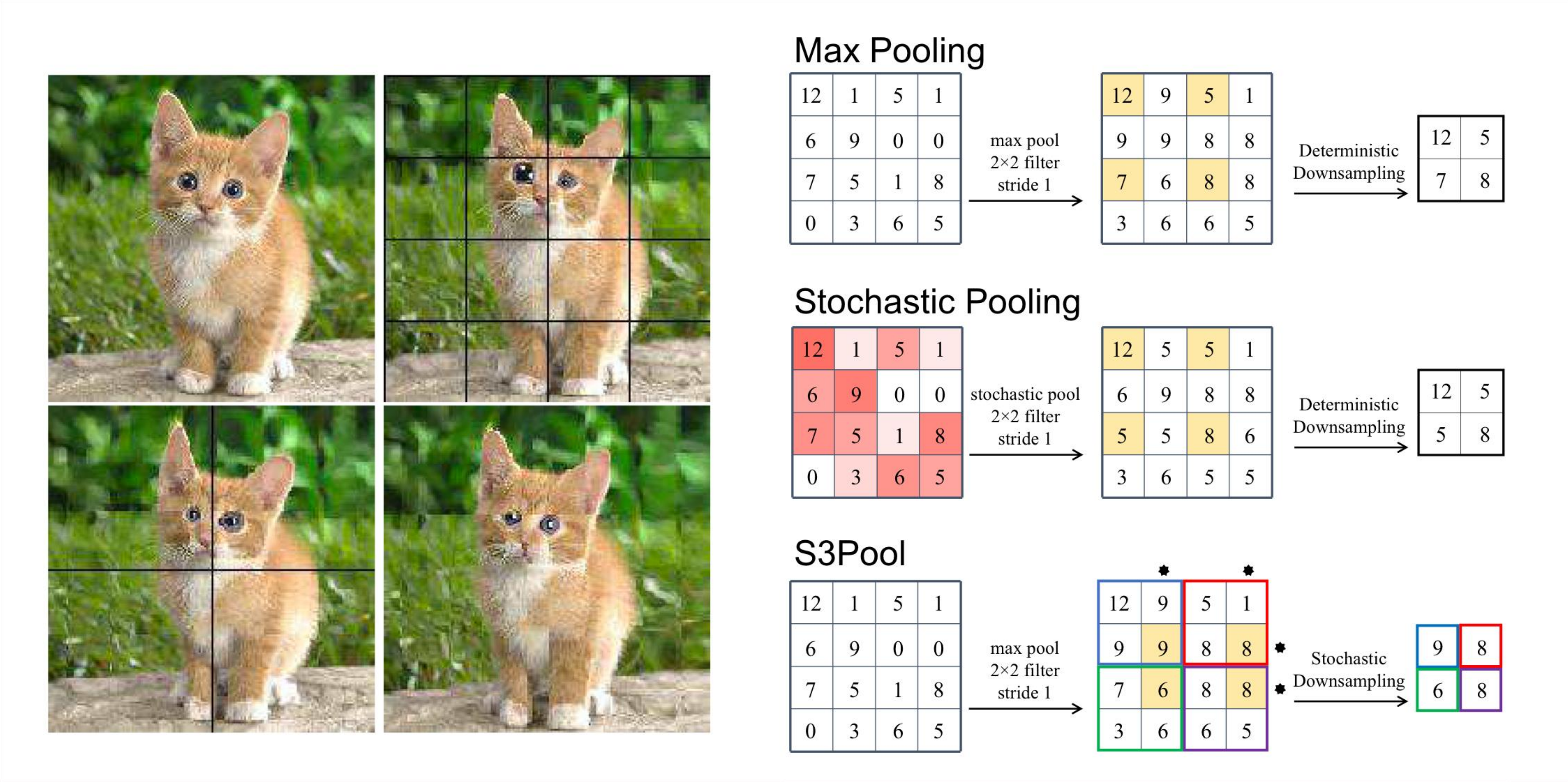


Pooling layers downsamples the volume spatially, reducing small translations of the features. They also provide a parameter reduction.

https://www.slideshare.net/perone/deep-learning-convolutional-neural-networks

Source: CS231n Convolutional Neural Networks for Visual Recognition.

Pooling Layer



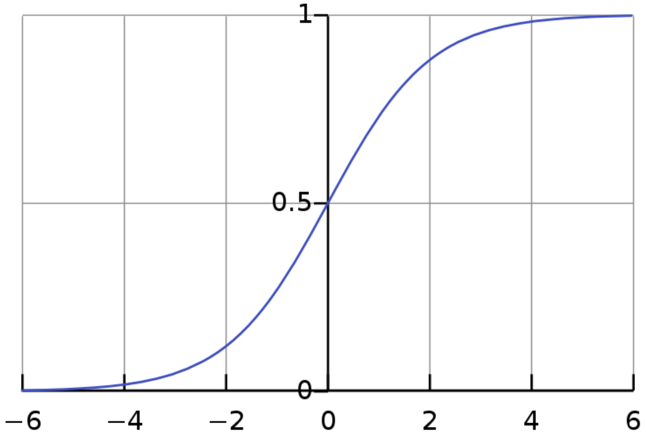
https://www.spacewu.com/posts/s3pool/

Activate Function - Sigmoid Function (Squashing Function)

- A **sigmoid function** is a mathematical function having a characteristic "S"-shaped curve or **sigmoid curve**. Often, *sigmoid function* refers to the special case of the logistic function shown in the first figure and defined by the formula

$$f(x) = \sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

$$f'(x) = f(x)(1 - f(x))$$

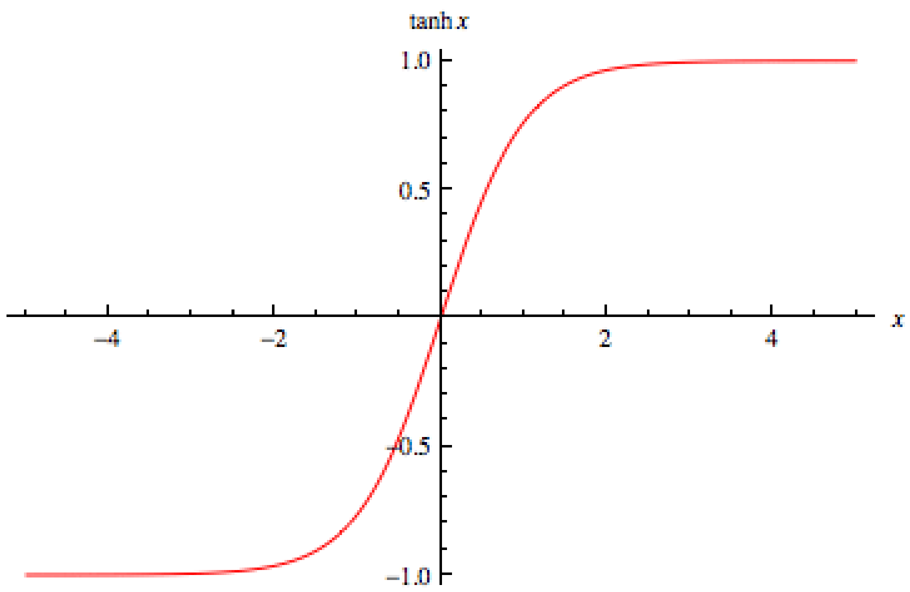


https://en.wikipedia.org/wiki/Sigmoid_function

Activate Function - Hyperbolic tangent

$$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

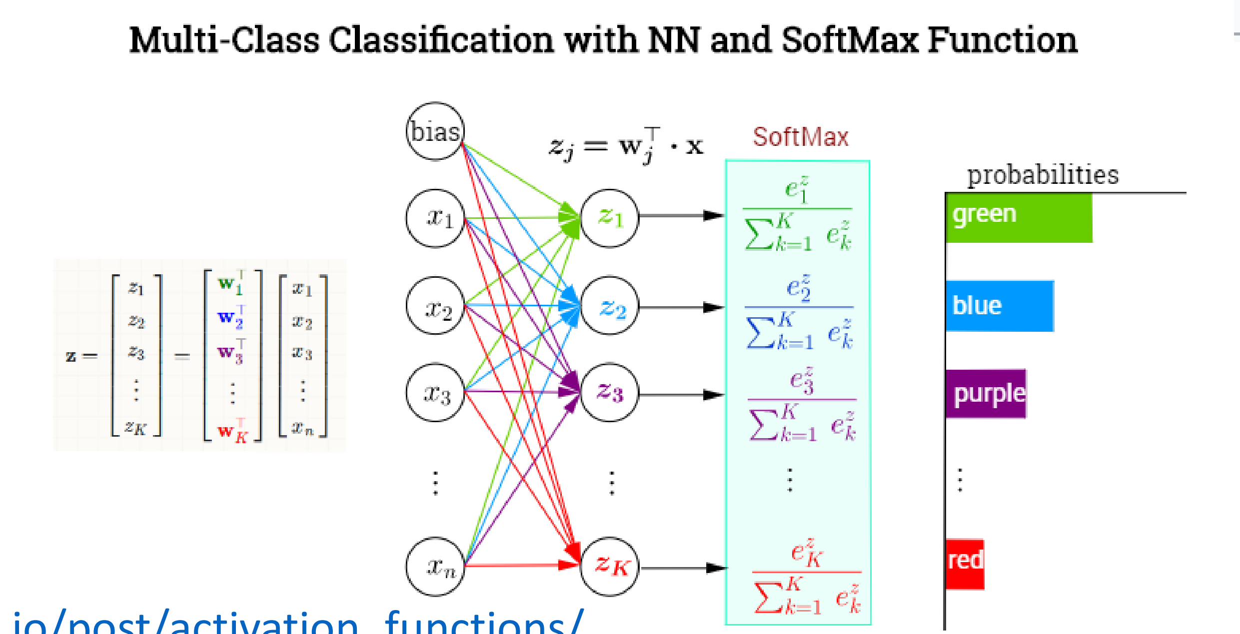
$$f'(x) = 1 - f(x)^2$$



Graph: http://mathworld.wolfram.com/HyperbolicTangent.html

Activate Function - SoftMax

Name	Equation	Derivatives	Range	Order of continuity
Softmax	$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$ for i = 1, …, J	$\frac{\partial f_i(\vec{x})}{\partial x_j} = f_i(\vec{x})(\delta_{ij} - f_j(\vec{x}))$[7]	$(0,1)$	C^∞



https://isaacchanghau.github.io/post/activation_functions/

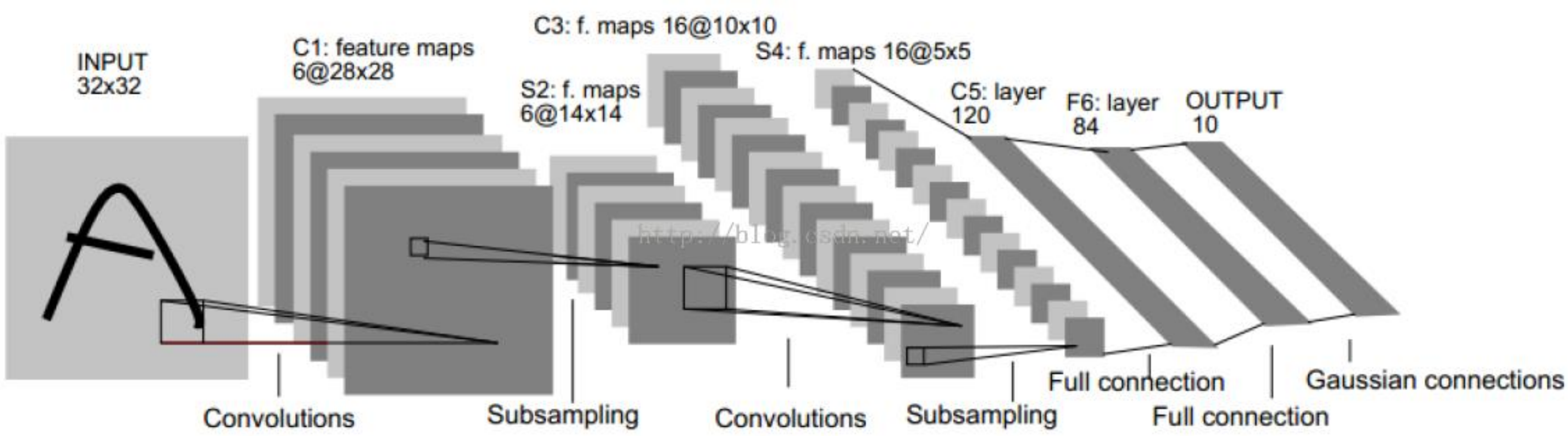
Loss Function - MSE(Mean Square Error)

- If a vector of ***n*** predictions generated from a sample of *n* data points on all variables, and **Y** is the vector of observed values of the variable being predicted, then the within-sample MSE of the predictor is computed as

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

https://en.wikipedia.org/wiki/Mean_squared_error

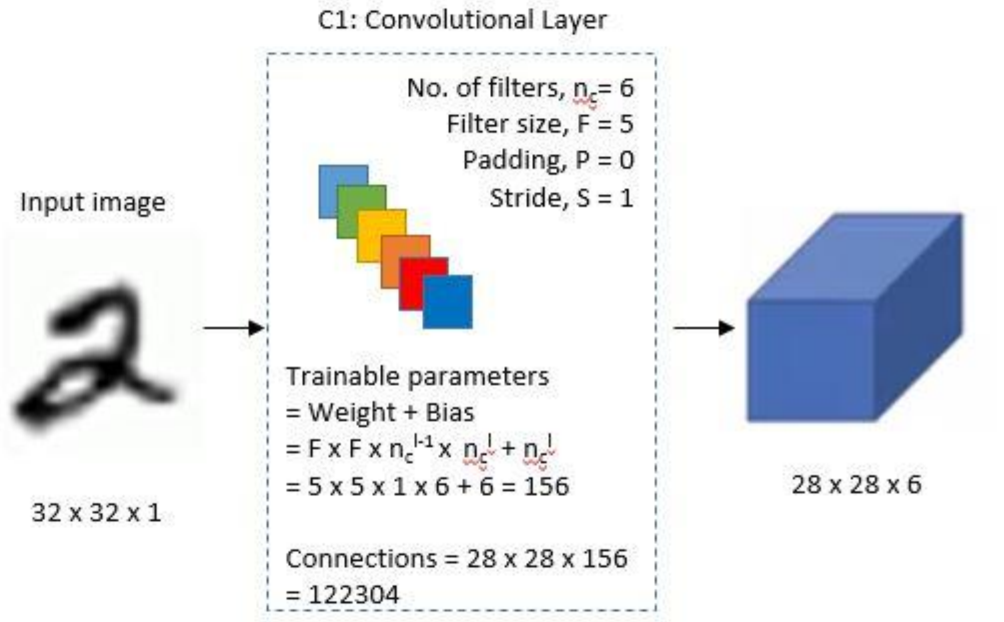
LeNet-5 Network架構介紹



3個卷積層(Convolution Layer)
2個池化層(Max Pooling Layer)
1個全連接層(Fully Connected Layer)

C1 Layer

- 卷積層
- 使用5x5的feature maps使原本放大為32x32的樣本變回28x28
- 輸出6張特徵圖進下一層
- 參數：(5x5+1)x6=156
- 連接點：(5x5+1)x28x28x6=122304



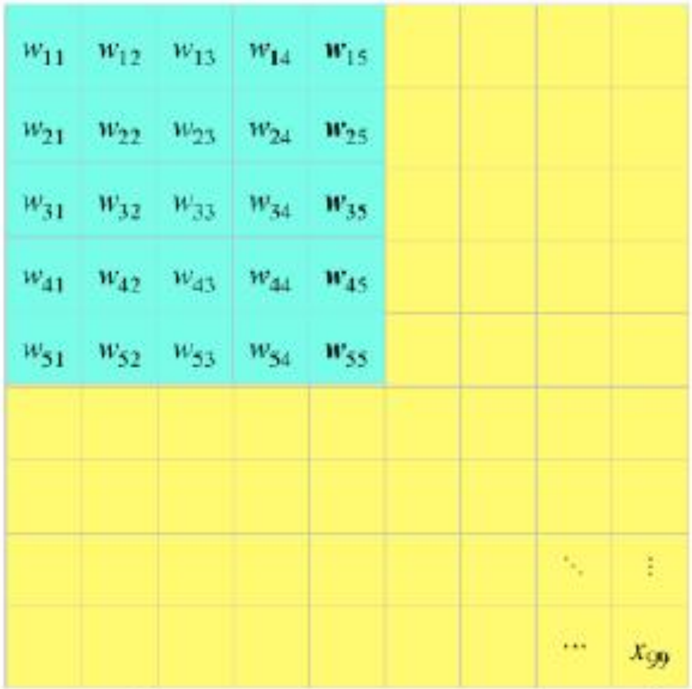
https://engmrk.com/lenet-5-a-classic-cnn-architecture/

Convolutional kernel

- The initial weight of each element is:
- Where Fi is the input amount (Fan - in)

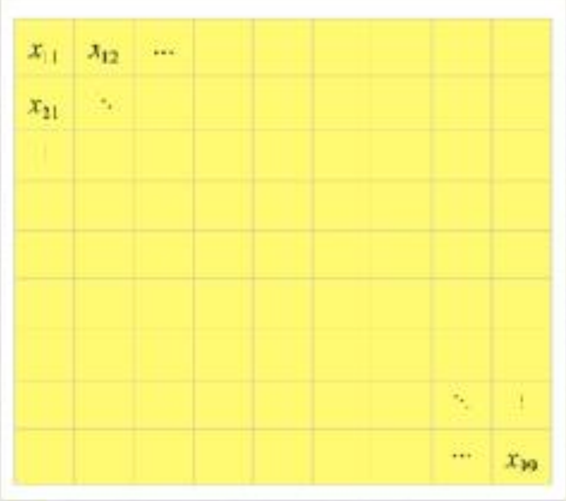
$$from \ \ -2.4/F_i \ to \ 2.4/F_i$$

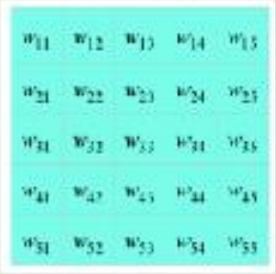
OPERATION OF CONVOLUTION NEURON

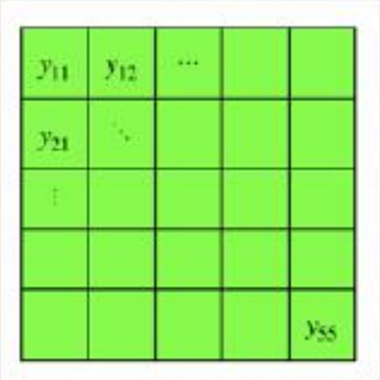


$$y_{11} = x_{11} * w_{11} + x_{12} * w_{12} + \ldots + x_{55} * w_{55} + b$$

Forward



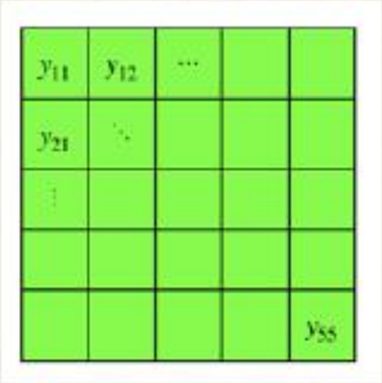


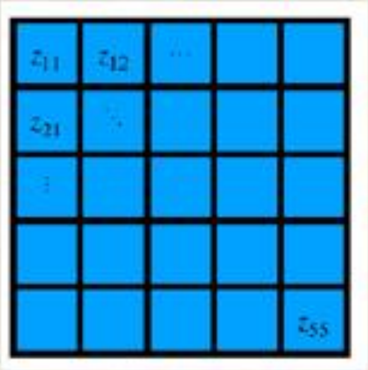


For any element of output array:

$$y_{ij} = \sum_{m=1,n=1}^{5,5} (x_{i+m-1,j+n-1} * w_{mn}) + b$$

Forward





$$z_{ij} = Atanh(Sy_{ij})$$

$$A = 1.7159$$

$$S = 2/3$$

Backward
Weight update

$$\because y_{ij} = \sum_{m=1,n=1}^{5,5} (x_{i+m-1,j+n-1} * w_{mn}) + b \quad ; \quad z_{ij} = tanh(y_{ij})$$

$$\partial z_{ij} / \partial y_{ij} = 1 - tanh^2(y_{ij}) = 1 - z_{ij}^2 \quad ; \quad \partial y_{ij} / \partial w_{mn} = x_{i+m-1,j+n-1}$$

$$\therefore \partial \mathbf{E} / \partial w_{11} = (\partial \mathbf{E} / \partial z_{11})(\partial z_{11} / \partial y_{11})(\partial y_{11} / \partial w_{11}) +$$

$$(\partial \mathbf{E} / \partial z_{12})(\partial z_{12} / \partial y_{12})(\partial y_{12} / \partial w_{11}) + \ldots (\partial \mathbf{E} / \partial z_{55})(\partial z_{55} / \partial y_{55})(\partial y_{55} / \partial w_{11})$$

$$= (E_{11})(1 - z_{11}^2)(x_{11}) + (E_{12})(1 - z_{12}^2)(x_{12}) + \ldots + (E_{55})(1 - z_{55}^2)(x_{55})$$

$$w_{11new} = w_{11} - \gamma(\partial \mathbf{E} / \partial w_{11})$$

$$E_{ij} = \partial \mathbf{E} / \partial z_{ij}$$

Backward
Weight update

$$\partial \mathbf{E}/\partial w_{55} = (\partial \mathbf{E}/\partial z_{11})(\partial z_{11}/\partial y_{11})(\partial y_{11}/\partial w_{55}) +$$

$$(\partial \mathbf{E}/\partial z_{12})(\partial z_{12}/\partial y_{12})(\partial y_{12}/\partial w_{55}) + \ldots (\partial \mathbf{E}/\partial z_{55})(\partial z_{55}/\partial y_{55})(\partial y_{55}/\partial w_{55})$$

$$= (E_{11})(1 - z_{11}^2)(x_{55}) + (E_{12})(1 - z_{12}^2)(x_{56}) + \ldots + (E_{55})(1 - z_{55}^2)(x_{99})$$

$$w_{55new} = w_{55} - \gamma(\partial \mathbf{E}/\partial w_{55})$$

$$\partial \mathbf{E}/\partial w_{ij} = \sum_{m=1,n=1}^{5,5} (E_{mn})(1 - z_{mn}^2)(x_{m+i,n+j})$$

$$w_{ijnew} = w_{ij} - \gamma(\partial \mathbf{E}/\partial w_{ij})$$

Backward
Bias update

$$\because y_{ij} = \sum_{m=1,n=1}^{5,5} (x_{i+m-1,j+n-1} * w_{mn}) + b \;\; ; \;\; z_{ij} = tanh(y_{ij})$$

$$\partial z_{ij}/\partial y_{ij} = 1 - tanh^2(y_{ij}) = 1 - z_{ij}^2 \;\; ; \;\; \partial y_{ij}/\partial b = 1$$

$$\therefore \partial \mathbf{E}/\partial b = (\partial \mathbf{E}/\partial z_{11})(\partial z_{11}/\partial y_{11})(\partial y_{11}/\partial b) +$$

$$(\partial \mathbf{E}/\partial z_{12})(\partial z_{12}/\partial y_{12})(\partial y_{12}/\partial b) + \ldots (\partial \mathbf{E}/\partial z_{55})(\partial z_{55}/\partial y_{55})(\partial y_{55}/\partial b)$$

$$= (E_{11})(1 - z_{11}^2) + (E_{12})(1 - z_{12}^2) + \ldots + (E_{55})(1 - z_{55}^2))$$

$$\partial \mathbf{E}/\partial b = \sum_{m=1,n=1}^{5,5} (E_{mn})(1 - z_{mn}^2)(1)$$

$$b_{new} = b - \gamma(\partial \mathbf{E}/\partial b)$$

Backward
Error back propagation

$$\because y_{ij} = \sum_{m=1,n=1}^{5,5} (x_{i+m-1,j+n-1} * w_{mn}) + b \quad ; \quad z_{ij} = tanh(y_{ij})$$

$$\partial z_{ij} / \partial y_{ij} = 1 - tanh^2(y_{ij}) = 1 - z_{ij}^2 \quad ; \quad \partial y_{ij} / \partial x_{i+m-1,j+n-1} = w_{mn}$$

$$\therefore \partial \mathbf{E} / \partial x_{11} = (\partial \mathbf{E} / \partial z_{11})(\partial z_{11} / \partial y_{11})(\partial y_{11} / \partial x_{11}) = (E_{11})(1 - z_{11}^2) w_{11}$$

$$\partial \mathbf{E} / \partial x_{12} = (\partial \mathbf{E} / \partial z_{11})(\partial z_{11} / \partial y_{11})(\partial y_{11} / \partial x_{12}) + (\partial \mathbf{E} / \partial z_{12})(\partial z_{12} / \partial y_{12})(\partial y_{12} / \partial x_{12})$$

$$= (E_{11})(1 - z_{11}^2) w_{12} + (E_{12})(1 - z_{12}^2) w_{11}$$

Backward
Error back propagation

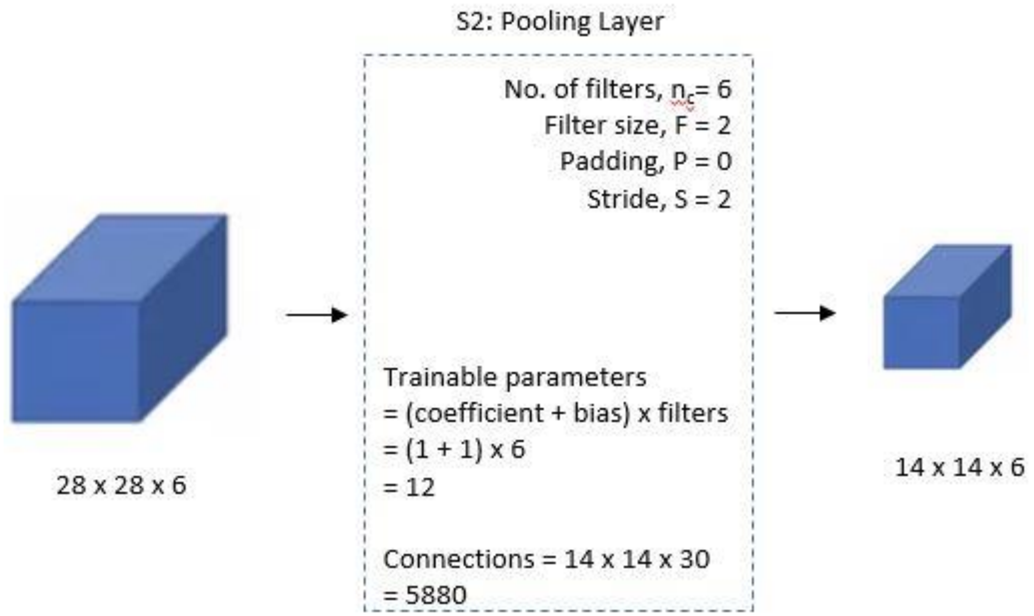
$$\partial \mathbf{E}/\partial x_{55} = (\partial \mathbf{E}/\partial z_{11})(\partial z_{11}/\partial y_{11})(\partial y_{11}/\partial x_{55}) + (\partial \mathbf{E}/\partial z_{12})(\partial z_{12}/\partial y_{12})(\partial y_{12}/\partial x_{55}) +$$
$$\ldots + (\partial \mathbf{E}/\partial z_{55})(\partial z_{55}/\partial y_{55})(\partial y_{55}/\partial x_{55})$$
$$= (E_{11})(1 - z_{11}^2)w_{55} + (E_{12})(1 - z_{12}^2)w_{54} + \ldots + (E_{55})(1 - z_{55}^2)w_{11}$$

$$\partial \mathbf{E}/\partial x_{99} = (\partial \mathbf{E}/\partial z_{55})(\partial z_{11}/\partial y_{55})(\partial y_{55}/\partial x_{99}) = (E_{55})(1 - z_{55}^2)w_{55}$$

$$\partial \mathbf{E}/\partial x_{ab} = \sum_{x_{ab} \subset Emn} (E_{mn})(1 - z_{mn}^2)(w_{ij})$$

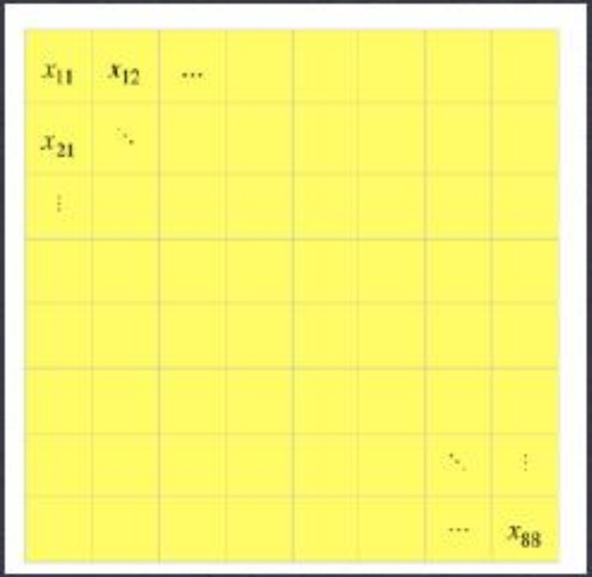
S2 Layer

- 池化層
- 使用2x2的框架進行池化，將圖形降階為14x14的大小
- 輸出6張特徵圖進下一層
- 參數： 6x2=12
- 連接點： 14x14x(2x2+1)x6=5880

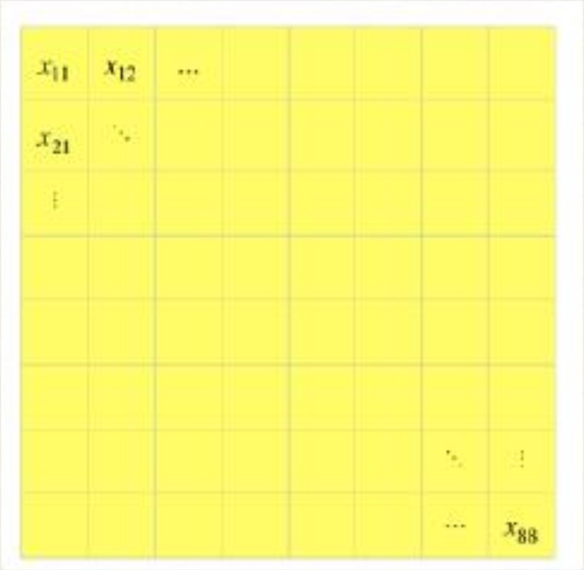


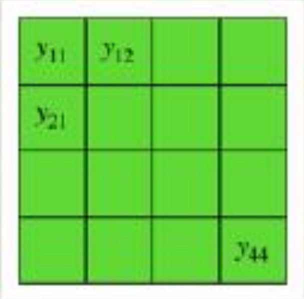
https://engmrk.com/lenet-5-a-classic-cnn-architecture/

OPERATION OF SUBSAMPLING (POOLING) NEURON



Forward





$$y_{11} = w(x_{11} + x_{12} + x_{21} + x_{22}) + b$$

$$y_{mn} = w \sum_{i=n, j=m}^{n+1, m+1} x_{ij} + b$$

Backward
Weight update

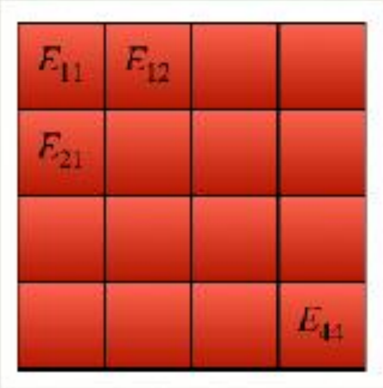
$$\because\ y_{ij} = w \sum_{n=i,m=j}^{i+1,j+1} x_{nm} + b$$

$$\partial y_{ij} / \partial w = \sum_{n=i,m=j}^{i+1,j+1} x_{nm}$$

$$\therefore\ \partial \mathbf{E} / \partial w = (\partial \mathbf{E}/\partial y_{11})(\partial y_{11}/\partial w) + (\partial \mathbf{E}/\partial y_{12})(\partial y_{12}/\partial w) + \ldots + (\partial \mathbf{E}/\partial y_{44})(\partial y_{44}/\partial w)$$

$$= \sum_{i=1,j=1}^{4,4} E_{ij} \Big(\sum_{n=i,m=j}^{i+1,j+1} x_{nm} \Big)$$

$$w_{new} = w - \gamma (\partial \mathbf{E}/\partial w)$$



$$E_{ij} = \partial \mathbf{E} / \partial y_{ij}$$

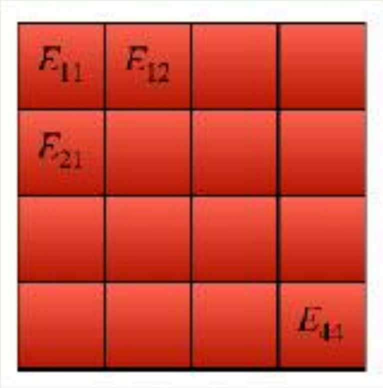
Backward
Bias update

$$\because y_{ij} = w \sum_{n=i,m=j}^{i+1,j+1} x_{nm} + b$$

$$\partial y_{ij} / \partial b = 1$$

$$\begin{aligned}\therefore \partial \mathbf{E} / \partial b &= (\partial \mathbf{E} / \partial y_{11})(\partial y_{11} / \partial b) + (\partial \mathbf{E} / \partial y_{12})(\partial y_{12} / \partial b) + \\ &\quad \ldots + (\partial \mathbf{E} / \partial y_{44})(\partial y_{44} / \partial b) \\ &= \sum_{i=1,j=1}^{4,4} E_{ij}(1)\end{aligned}$$

$$b_{new} = b - \gamma(\partial \mathbf{E} / \partial b)$$



$$E_{ij} = \partial \mathbf{E} / \partial y_{ij}$$

Backward
Error back propagation

$$\because y_{ij} = w \sum_{n=i,m=j}^{i+1,j+1} x_{nm} + b$$

$$\partial y_{ij} / \partial x_{nm} = w$$

$$\partial \mathbf{E} / \partial x_{11} = (\partial \mathbf{E} / \partial y_{11})(\partial y_{11} / \partial x_{11})$$

$$\cdots$$

$$\partial \mathbf{E} / \partial x_{15} = (\partial \mathbf{E} / \partial y_{13})(\partial y_{13} / \partial x_{15})$$

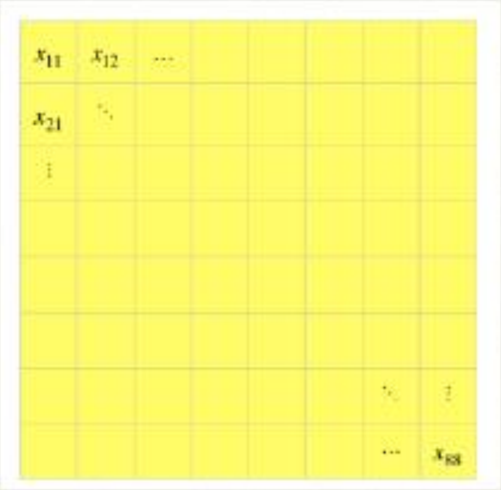
$$\cdots$$

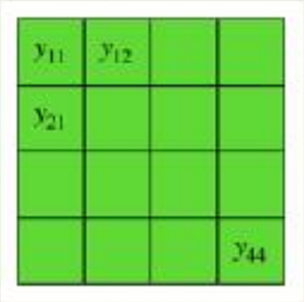
$$\partial \mathbf{E} / \partial x_{88} = (\partial \mathbf{E} / \partial y_{44})(\partial y_{44} / \partial x_{88})$$

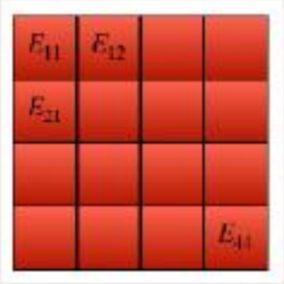
$$\partial \mathbf{E} / \partial x_{ab} = (\partial \mathbf{E} / \partial y_{ij})(\partial y_{ij} / \partial x_{ab})$$

where

$$y_{ij} \ni x_{ab}$$







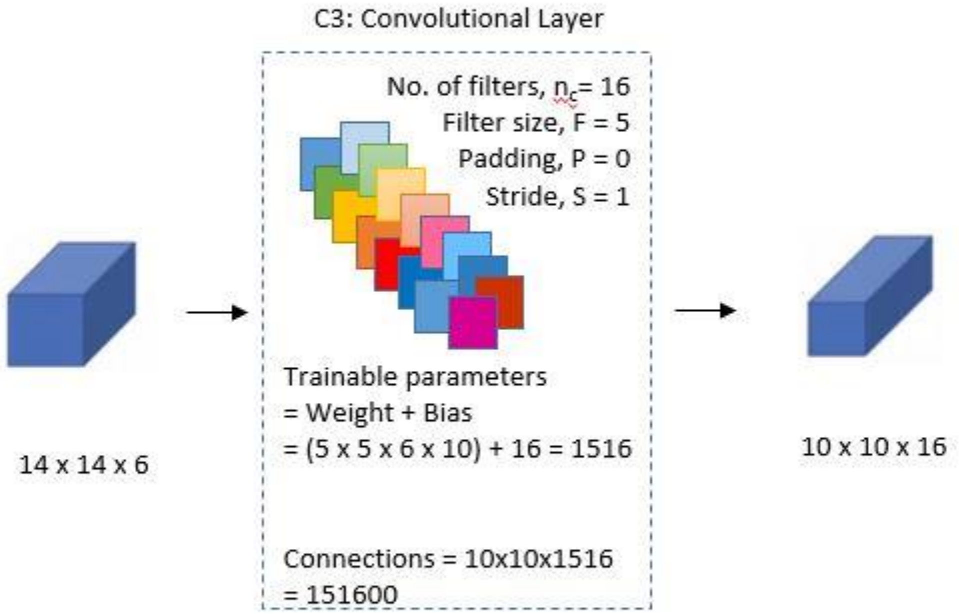
$$E_{ij} = \partial \mathbf{E} / \partial y_{ij}$$

C3 Layer

- 卷積層
- 使用5x5的feature maps輸出10x10大小
- 輸出共16張進下一層
- 參數： (5x5x6x10)+16=1516
- 連接點： 10x10x1515=151600

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

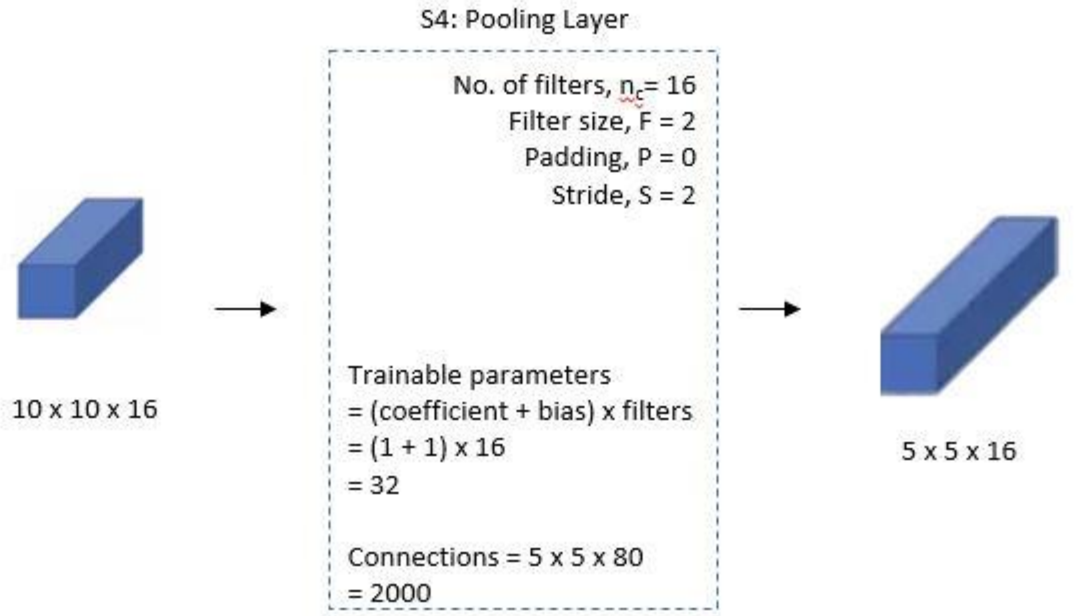
TABLE I
EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.



https://engmrk.com/lenet-5-a-classic-cnn-architecture/

S4 Layer

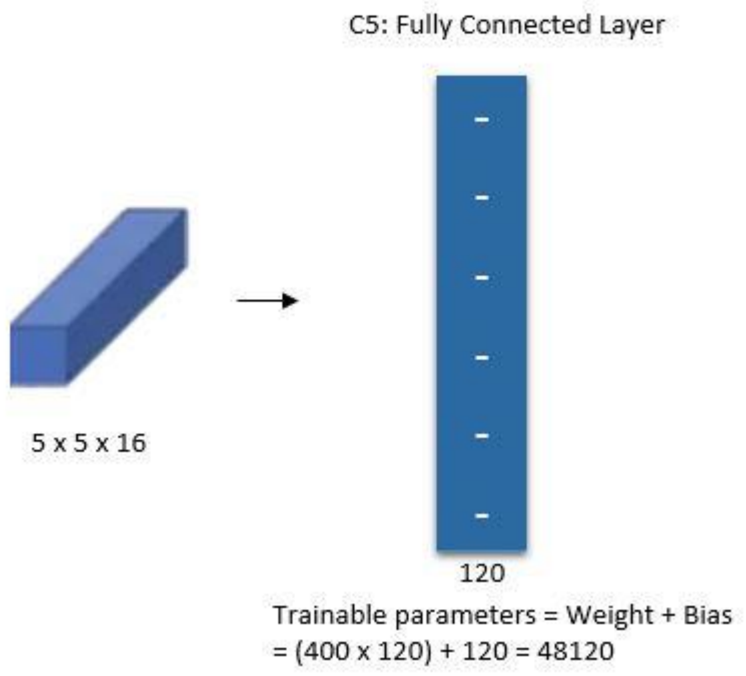
- 池化層
- 使用2x2的框架進行池化，將圖形降階為5x5的大小
- 輸出16張特徵圖進下一層
- 參數： (1+1)x16=32
- 連接點： 5x5x80=2000



https://engmrk.com/lenet-5-a-classic-cnn-architecture/

C5 Layer

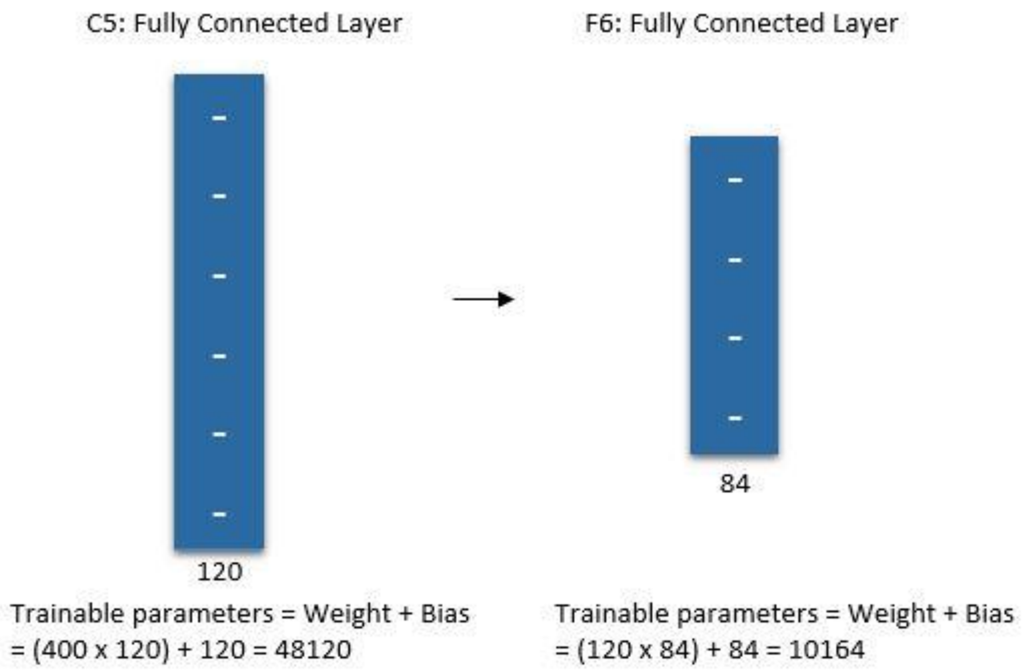
- 卷積層
- 使用5x5的feature maps輸出1x1大小
- 輸出共120張進下一層
- 參數&連接點： (400x120)+120=48120



https://engmrk.com/lenet-5-a-classic-cnn-architecture/

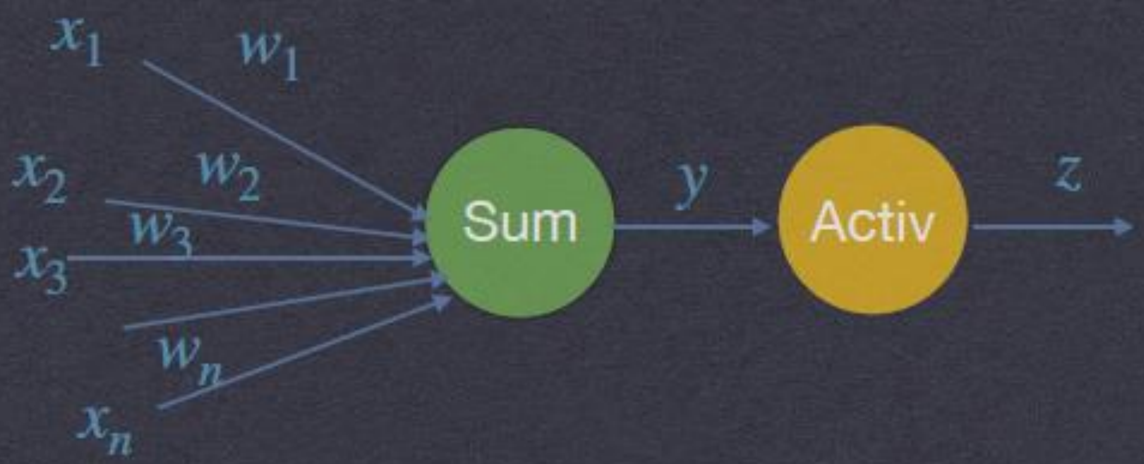
F6 Layer

- 全連接層
- 84個Fully Connected Units, 84=7x12, stylized image
- 參數&連接點： (120x84)+84=10164

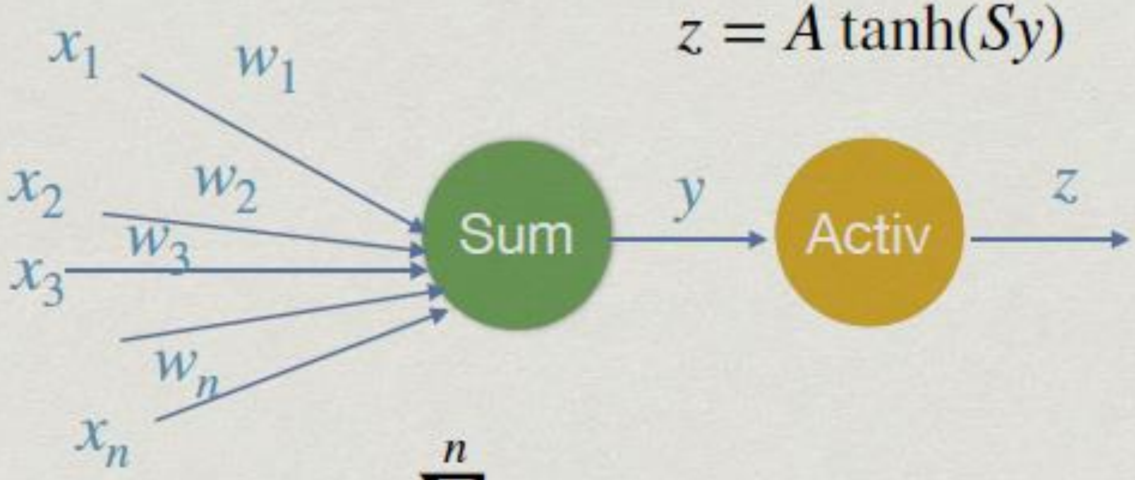


https://engmrk.com/lenet-5-a-classic-cnn-architecture/

OPERATION OF SCALAR NEURON



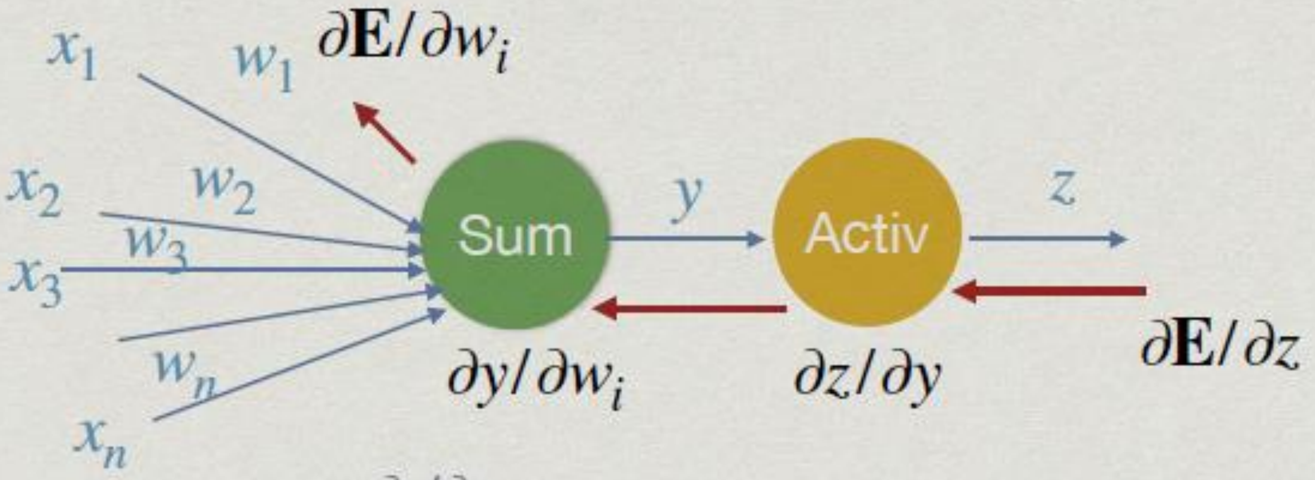
Forward



$$z = A \tanh(Sy)$$

$$y = \sum_{i}^{n} w_i x_i + b$$

Backward
Weight update



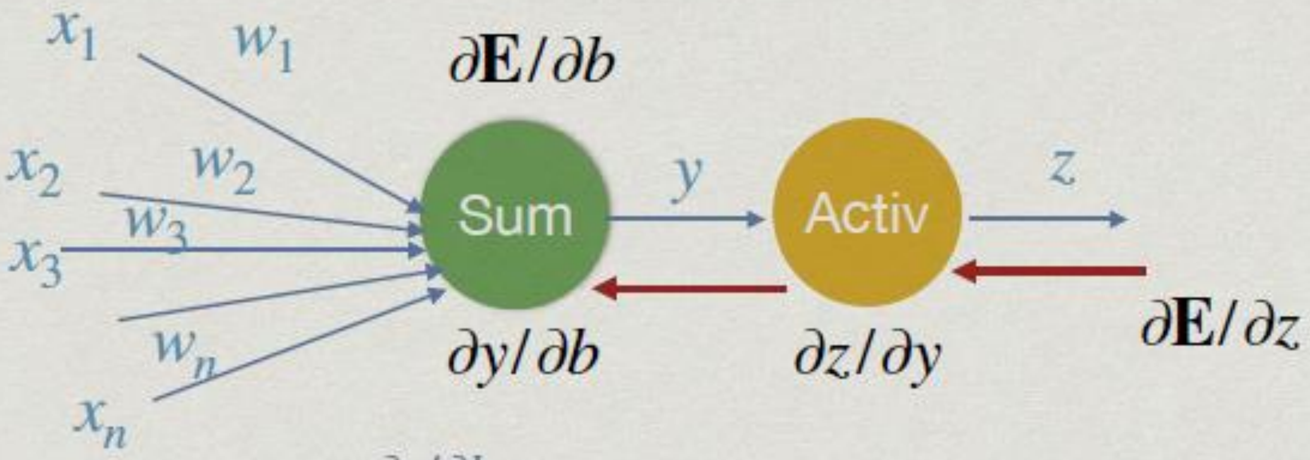
$$\partial \mathbf{E}/\partial w_i$$
$$= (\partial \mathbf{E}/\partial z)(\partial z/\partial y)(\partial y/\partial w_i)$$
$$= (\partial \mathbf{E}/\partial z)(A - (S/A)z^2)x_i$$

$$\partial y/\partial w_i =$$
$$\partial(\sum_i^n w_i x_i + b)/\partial w_i$$
$$= x_i$$

$$\partial z/\partial y = \partial(A \tanh(Sy))/\partial y$$
$$= A(1 - S \tanh^2(Sy))$$
$$= A - S/A(A^2 \tanh^2(Sy))$$
$$= A - (S/A)(z^2)$$

$$w_{inew} = w_i - \gamma(\partial \mathbf{E}/\partial w_i)$$

Backward
Bias update



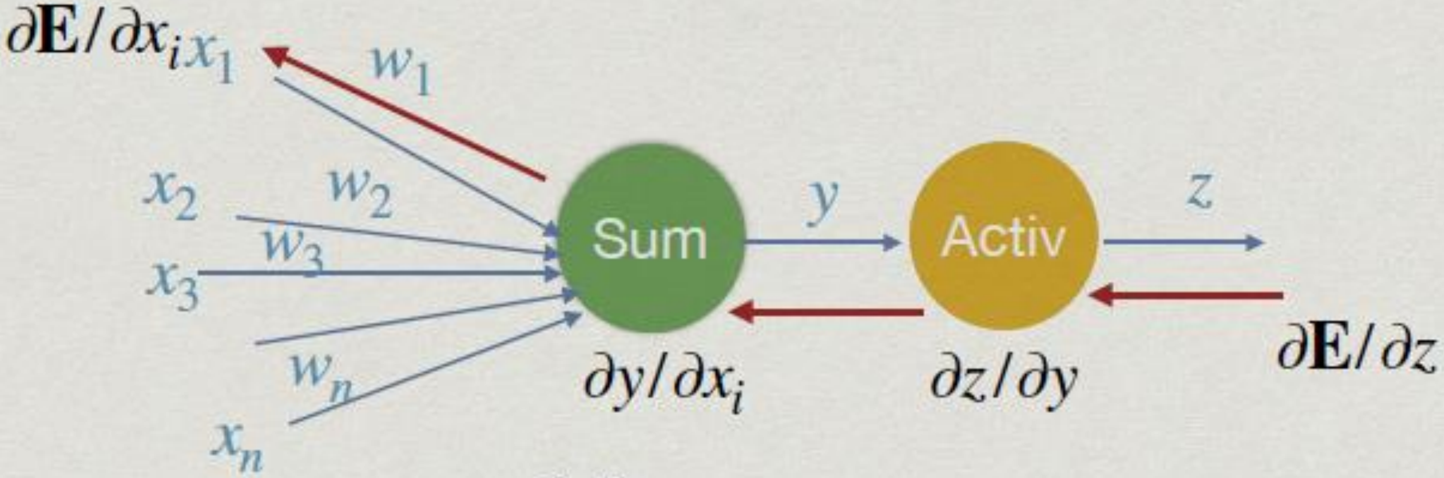
$$
\begin{aligned}
&\partial \mathbf{E}/\partial b \\
&= (\partial \mathbf{E}/\partial z)(\partial z/\partial y)(\partial y/\partial w_i) \\
&= (\partial \mathbf{E}/\partial z)(A - (S/A)z^2)1
\end{aligned}
$$

$$
\begin{aligned}
&\partial y/\partial b = \\
&\partial\Big(\sum_i^n w_i x_i + b\Big)/\partial b \\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
\partial z/\partial y &= \partial(A \tanh(Sy))/\partial y \\
&= A(1 - S\tanh^2(Sy)) \\
&= A - S/A(A^2 \tanh^2(Sy)) \\
&= A - (S/A)(z^2)
\end{aligned}
$$

$$b_{new} = b - \gamma(\partial \mathbf{E}/\partial b)$$

Backward
Error propagation



$$\begin{aligned}
&\partial \mathbf{E}/\partial x_i \\
&= (\partial \mathbf{E}/\partial z)(\partial z/\partial y)(\partial y/\partial x_i) \\
&= (\partial \mathbf{E}/\partial z)(A - (S/A)z^2)w_i
\end{aligned}$$

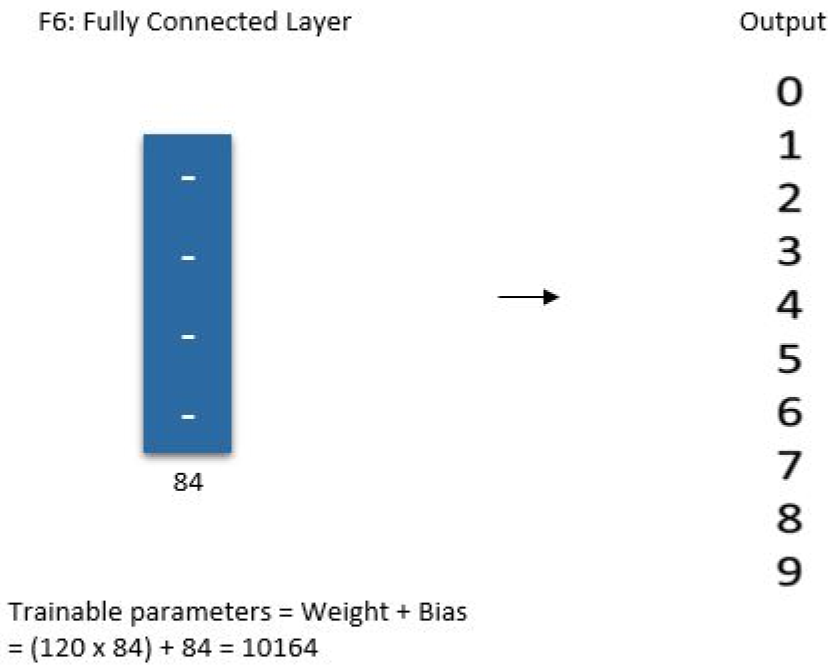
$$\partial y/\partial x_i = \partial\Big(\sum_i^n w_i x_i + b\Big)/\partial x_i = w_i$$

$$\begin{aligned}
\partial z/\partial y &= \partial(A\tanh(Sy))/\partial y \\
&= A(1 - S\tanh^2(Sy)) \\
&= A - S/A(A^2\tanh^2(Sy)) \\
&= A - (S/A)(z^2)
\end{aligned}$$

$$x_{inew} = x_i - \gamma(\partial \mathbf{E}/\partial x_i)$$

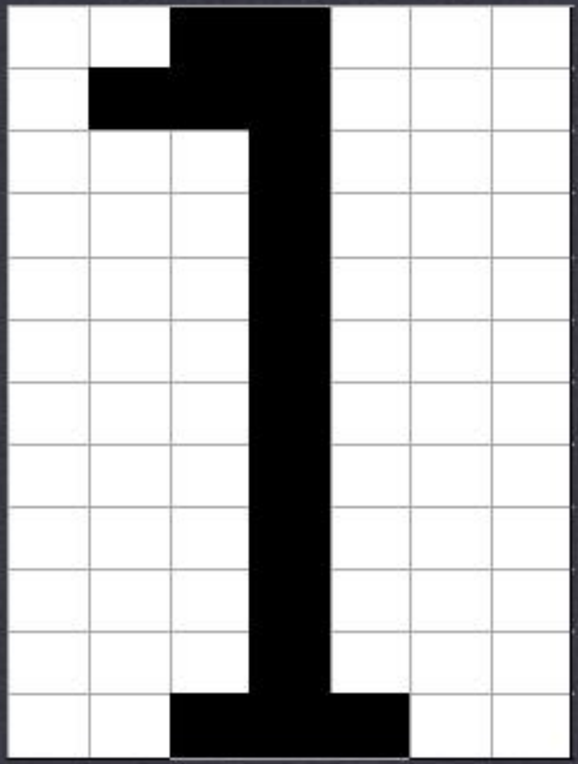
Output Layer

- 全連接層
- Softmax output layer ŷ包含10個數值，0-9

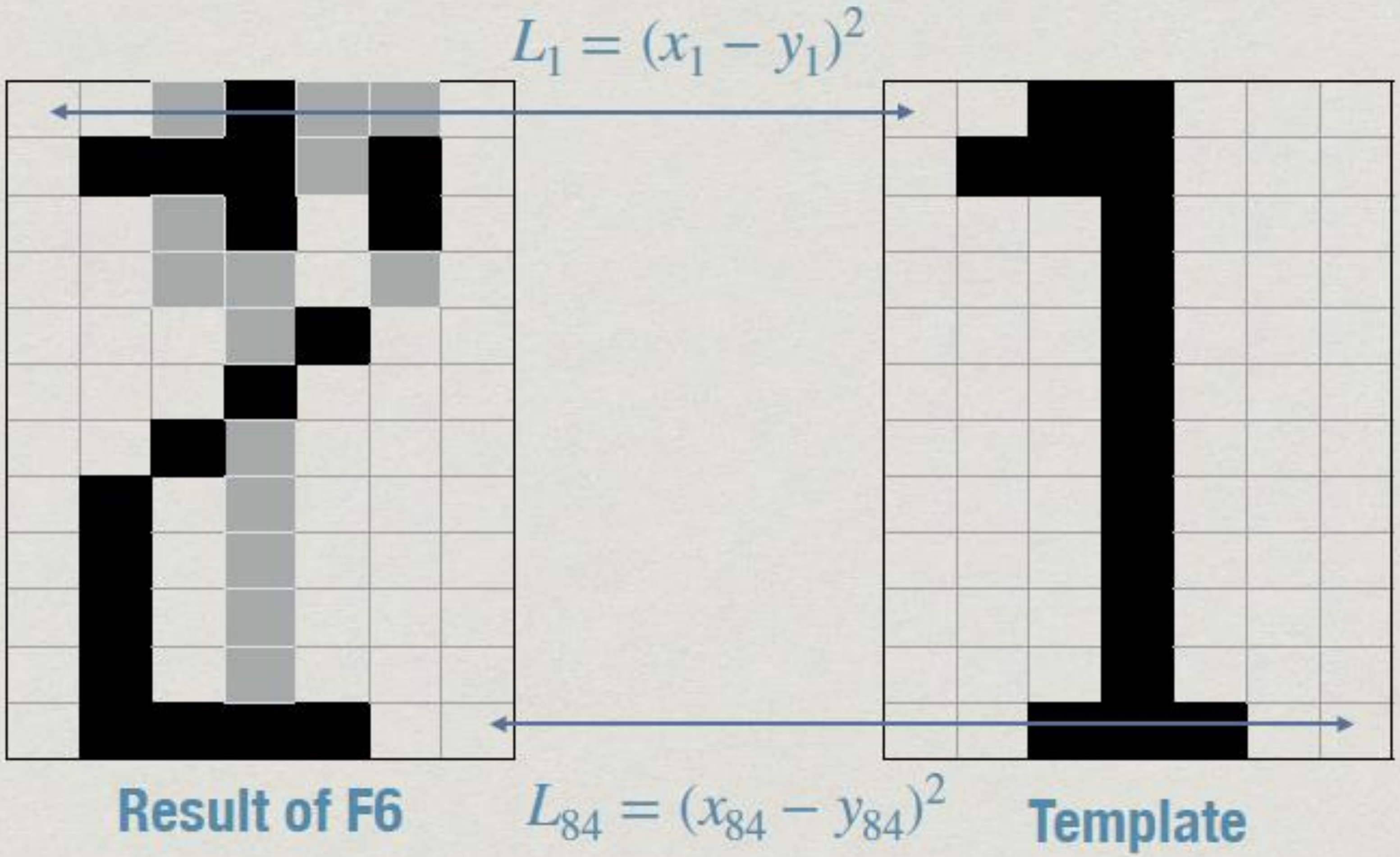


https://engmrk.com/lenet-5-a-classic-cnn-architecture/

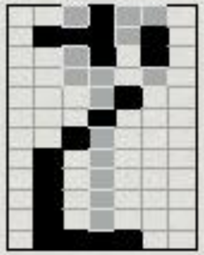
OPERATION OF OUTPUT LAYER



Forward



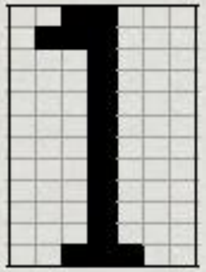
Forward



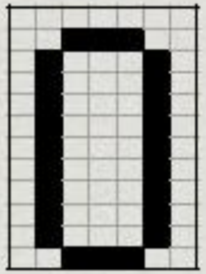
$$L_1 = \sum_{i=1}^{84} (x_i - y_{1i})^2$$

$$L_j = \sum_{i=1}^{84} (x_i - y_{ji})^2$$

$$L = \min(L_j) \qquad L_0 = \sum_{i=1}^{84} (x_i - y_{0i})^2$$



.
.
.



Result of F6

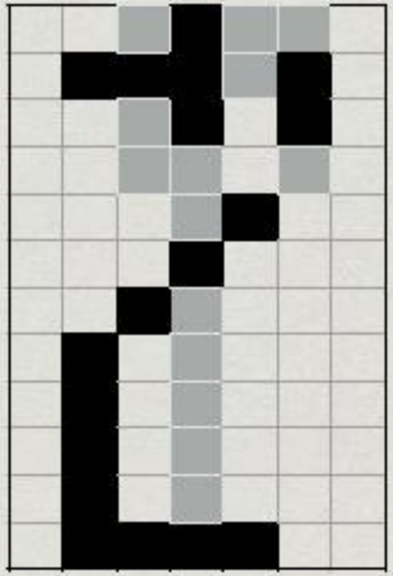
Templates

Backward

$$L = \sum_{i=1}^{84} (x_i - y_i)^2$$

$$\partial L / \partial x_i = 2(x_i - y_i)$$

$$\mathbf{E} = \{\partial L/\partial x_1, \ldots, \partial L/\partial x_{84}\}$$



PRE-TRAIN OF LENET 5

Pre-train

- Unlike modern CNN, LeNet needs pre-train to adjust learning rate before each epoch.
- It randomly takes small portion of data(500/60000) to pre-train.

Math of Pre-train

Stochastic Diagonal Levenberg-Marquardt Method

$$w_{new} = w - \epsilon(\partial \mathbf{E}/\partial w)$$

$$\epsilon = \eta/(\mu + h)$$

where μ is hand picked constant

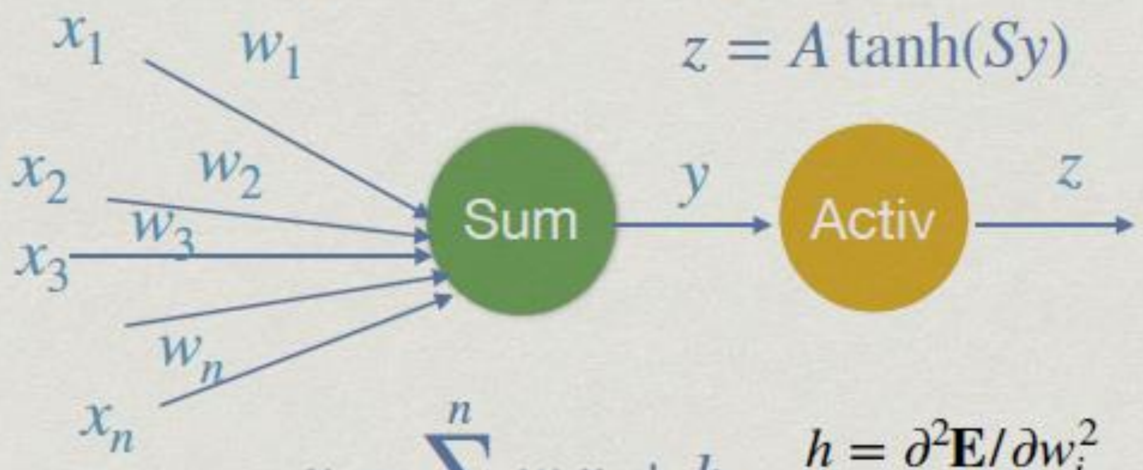
h is the second deri of $\mathbf{E}$ with respect to w

η is a stochastic number

Math of Pre-train

Stochastic Diagonal Levenberg-Marquardt Method

$$z = A\tanh(Sy)$$



$$y = \sum_{i}^{n} w_i x_i + b$$

$$\begin{aligned} h &= \partial^2 \mathbf{E}/\partial w_i^2 \\ &= \partial/\partial w_i((\partial \mathbf{E}/\partial z)(\partial z/\partial y)(\partial y/\partial w_i)) \\ &= (\partial/\partial w_i(\partial \mathbf{E}/\partial z))(\partial z/\partial y)(\partial y/\partial w_i) \\ &\quad + (\partial \mathbf{E}/\partial z)(\partial/\partial w_i(\partial z/\partial y))(\partial y/\partial w_i) \\ &\quad + (\partial \mathbf{E}/\partial z)(\partial z/\partial y)(\partial y/\partial w_i)^2 \end{aligned}$$

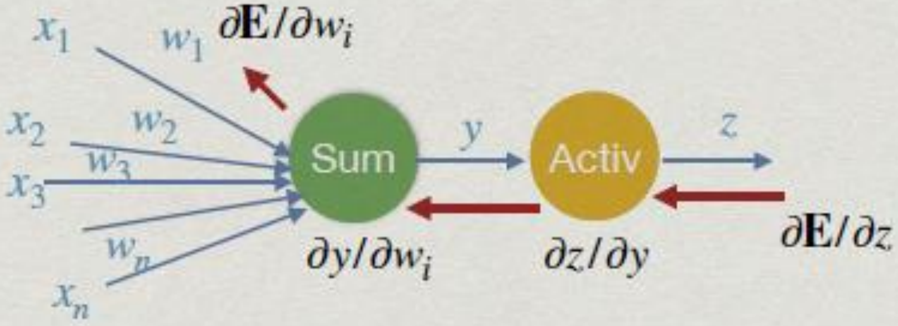
Math of Pre-train

$$
\begin{aligned}
h &= \partial^2 \mathbf{E} / \partial w_i^2 \\
&= \partial / \partial w_i ((\partial \mathbf{E} / \partial z)(\partial z / \partial y)(\partial y / \partial w_i)) \\
&= (\partial / \partial w_i (\partial \mathbf{E} / \partial z))(\partial z / \partial y)(\partial y / \partial w_i) \\
&\quad + (\partial \mathbf{E} / \partial z)(\partial / \partial w_i (\partial z / \partial y))(\partial y / \partial w_i) \\
&\quad + (\partial \mathbf{E} / \partial z)(\partial z / \partial y)(\partial^2 y / \partial w_i^2) \\
&\quad \because (\partial^2 y / \partial w_i^2) = 0 \\
&= (\partial / \partial w_i (\partial \mathbf{E} / \partial z))(\partial z / \partial y)(\partial y / \partial w_i) \\
&\quad + (\partial \mathbf{E} / \partial z)(\partial / \partial w_i (\partial z / \partial y))(\partial y / \partial w_i) \\
&= A + B
\end{aligned}
$$

$$
\begin{aligned}
A &= (\partial / \partial w_i (\partial \mathbf{E} / \partial z))(\partial z / \partial y)(\partial y / \partial w_i) \\
&\quad \because \partial z / \partial w_i = (\partial z / \partial y)(\partial y / \partial w_i) \\
A &= (\partial^2 \mathbf{E} / \partial z^2)(\partial z / \partial y)^2 (\partial y / \partial w_i)^2
\end{aligned}
$$

$$
\begin{aligned}
B &= (\partial \mathbf{E} / \partial z)(\partial / \partial w_i (\partial z / \partial y))(\partial y / \partial w_i) \\
&= (\partial \mathbf{E} / \partial z)(\partial^2 z / \partial y^2)(\partial y / \partial w_i)^2
\end{aligned}
$$

Math of Pre-train



$$
\begin{aligned}
h &= A + B \\
&= (\partial^2 \mathbf{E}/\partial z^2)(\partial z/\partial y)^2(\partial y/\partial w_i)^2 \\
&\quad + \cancel{(\partial \mathbf{E}/\partial z)(\partial^2 z/\partial y^2)(\partial y/\partial w_i)^2} \\
&= (\partial^2 \mathbf{E}/\partial z^2)(\partial z/\partial y)^2(\partial y/\partial w_i)^2
\end{aligned}
$$

$$
\begin{aligned}
&\partial y/\partial w_i = \\
&\partial\Big(\sum_i^n w_i x_i + b\Big)/\partial w_i \\
&= x_i
\end{aligned}
$$

$$
\begin{aligned}
\partial z/\partial y &= \partial(A\tanh(Sy))/\partial y \\
&= A(1 - S\tanh^2(Sy)) \\
&= A - S/A(A^2\tanh^2(Sy)) \\
&= A - (S/A)(z^2)
\end{aligned}
$$

Evaluation of h

- Run 500 samples:
- Once h is evaluated, the learning rate of the very weight is determined before epoch start.

$$h = (1/500) \sum_{n=1}^{500} h_n$$

$$h_n = (\partial^2 \mathbf{E_n} / \partial z^2)(\partial z / \partial y)^2 (\partial y / \partial w_i)^2$$

Special Contents

- HOS
- 文字辨識不會只辨識一個字母而是
  - 郵遞區號
  - 支票數字
  - 文字
- Word-Level辨識的優勢
  - 拒絕分割錯誤的特徵
  - 降低整體辨識錯誤率
- Viterbi transformer
  - 找出最好的詮釋路徑

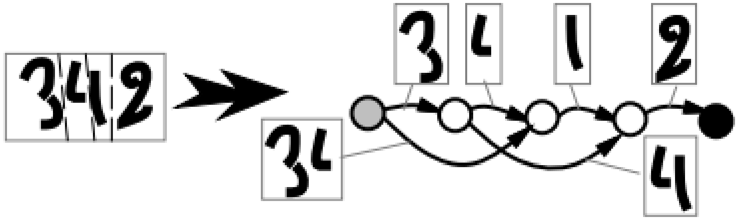


Fig. 16. Building a segmentation graph with Heuristic Over-Segmentation.

Lecun, 1998