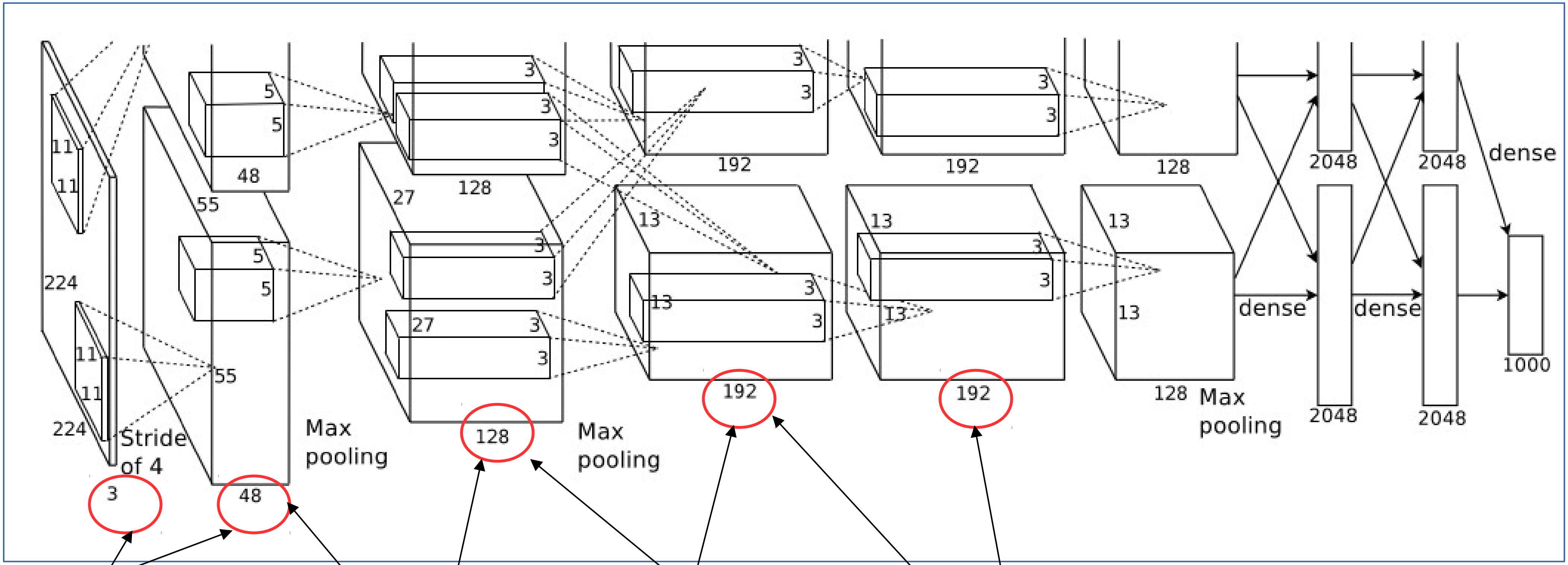
Reference: Alex Net

Reference: The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3)

https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html



Example 1: 48 kernels, the depth of the kernel is 3

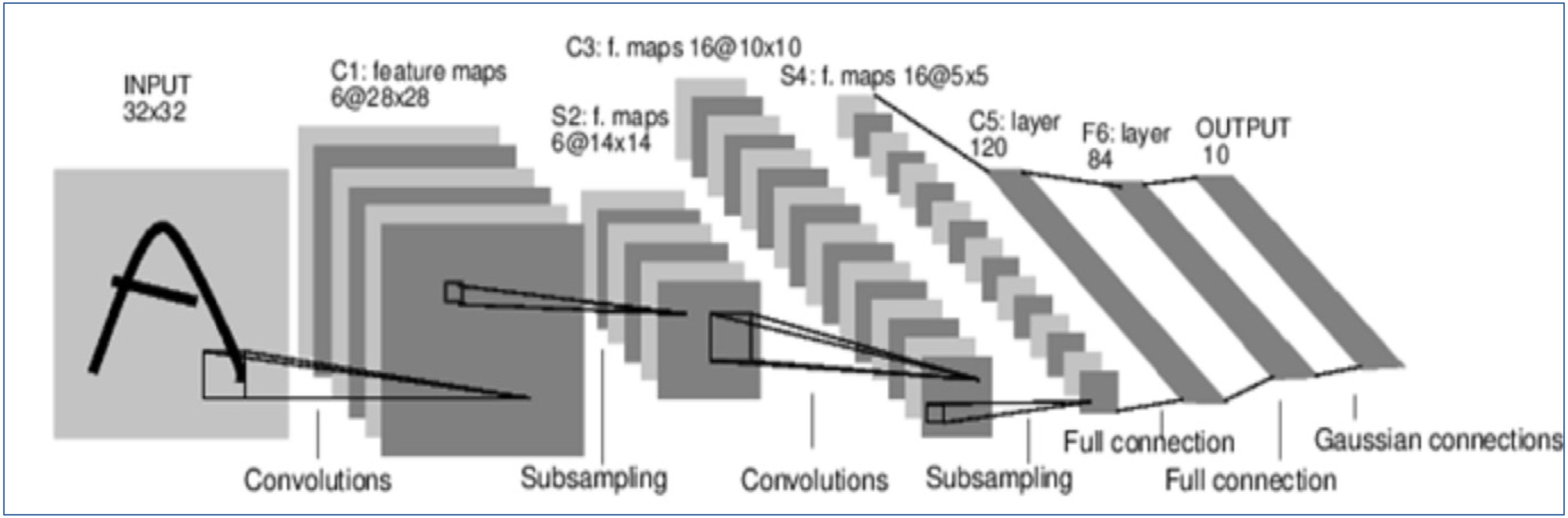
Example 2: 128 kernels, the depth of the kernel is 48

Example 3: 192 kernels, the depth of the kernel is 128

Example 4: 192 kernels, the depth of the kernel is 192

Reference 1: LeNet

http://timdettmers.com/2015/03/26/convolution-deep-learning/

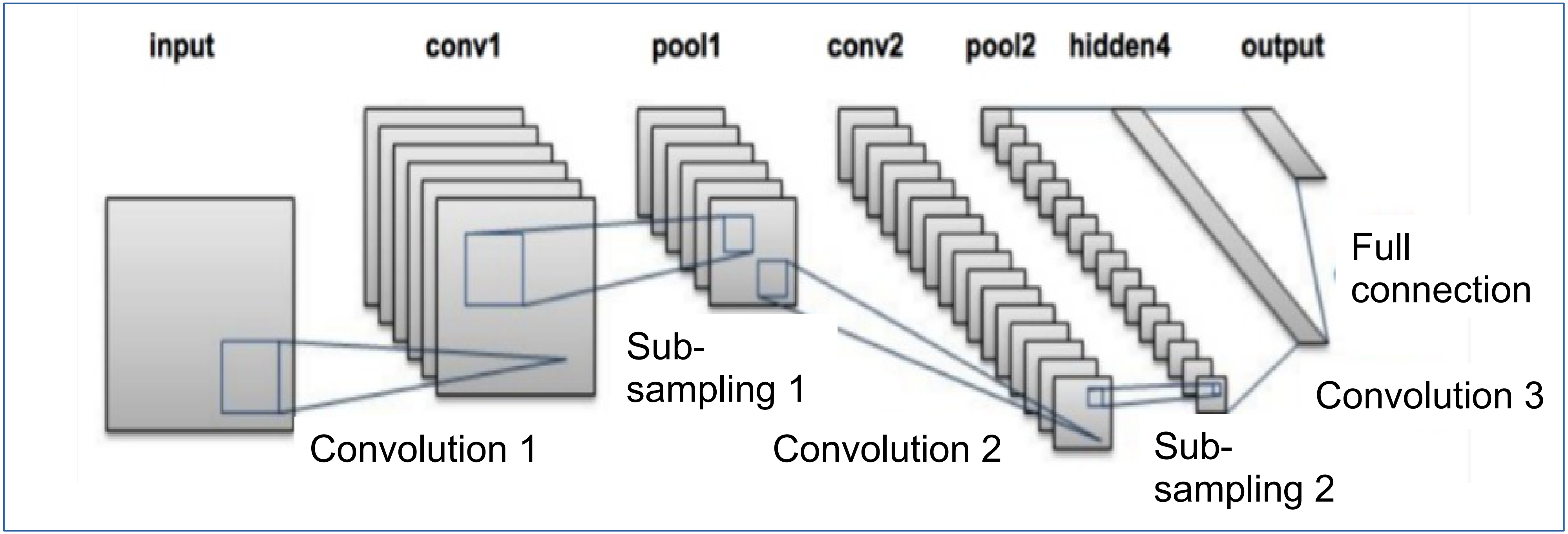


Convolution 1	Sub-sampling 1	Convolution 2	Sub-sampling 2	Full Conn 1	Full Conn2	Gau
C1	S2	C3	S4	Con5	F6	
Layer1	Layer2	Layer3	Layer4	Layer5	Layer6	output

Harry Li, Ph.D.

Reference 2: LeNet

https://www.pyimagesearch.com/2016/08/01/lenet-convolutional-neural-network-in-python/



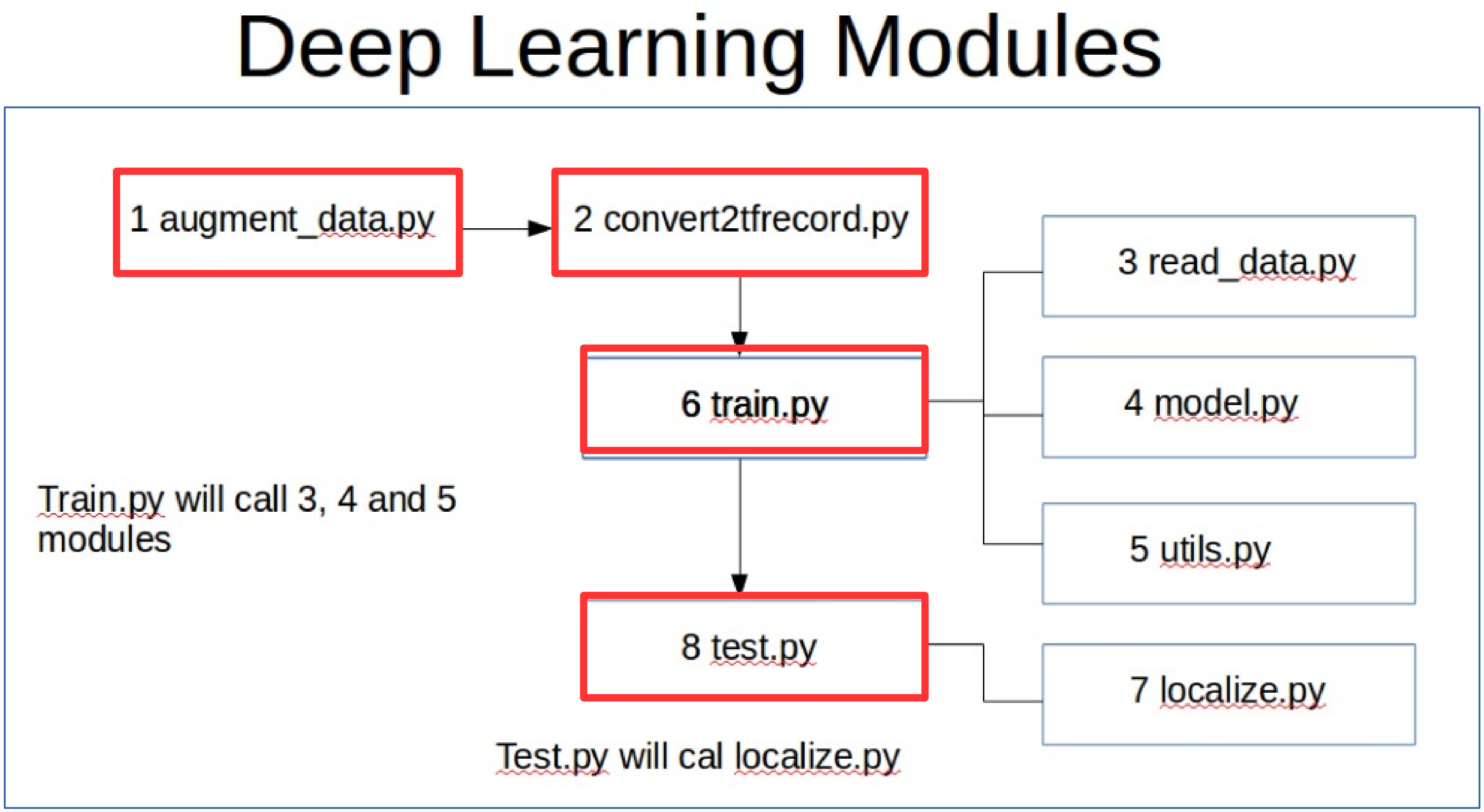
Convolution 1	Sub-sampling 1	Convolution 2	Sub-sampling 2	Convolution 3	Full connection
Conv1	pool1	Conv2	pool2	Hidden 4	Output
Layer1	Layer2	Layer3	Layer4	Layer5	

Harry Li, Ph.D.

CTI One Production Code: TDAT

1. Code name: TDAT for (Tensor flow based, Deep Learning enabled, AGV4000 Toolkit)

Architecture of the TDAT



Deep Learning Modules

Train.py will call 3, 4 and 5 modules

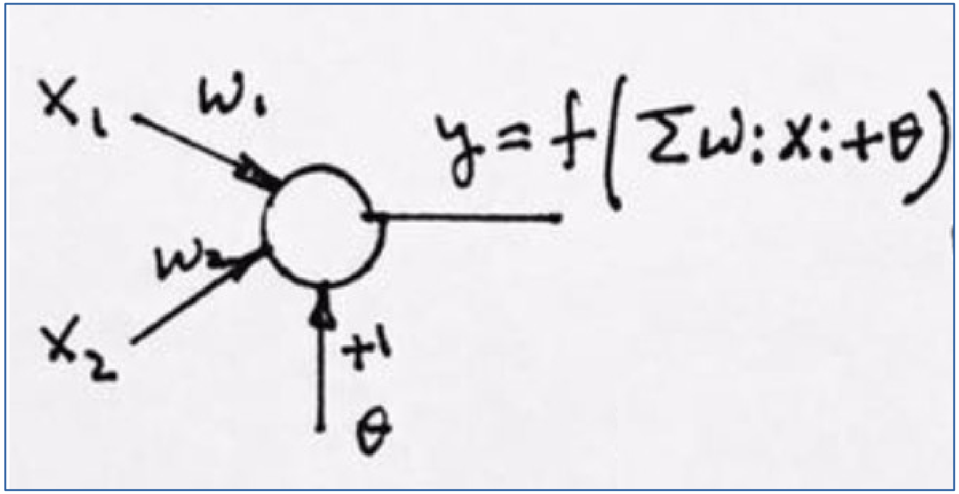
Test.py will cal localize.py

CTI One Production Sample Code

Table 1. Deep Learning Function Module Testing

Name of Module	Description	Execution and Application
1 augment_data.py	Augment cropped raw image data including Gaussian blur, motion blur, and Rotation to produce 20 new images.	$ python augment_data.py Note: raw data set directory path can be changed in program
2 convert2tfrecord.py	Convert image data set to tfrecord file.	$ python convert2tfrecord.py
3 read_data.py	Function of read data from tfrecord.	Called by train.py
4 model.py	3 models in model.py, lenet_advanced is used to train in this project.	Called by train.py
5 utils.py	Function of calculating loss and accuracy of training set	Called by train.py
6 train.py	Train the model using training data set.	$ python train.py
7 localize.py	Localize the traffic signs by pre-trained model.	Called by test.py
8 test.py	Deploy and test our trained model by reading image from real enviornment.	$ python test.py

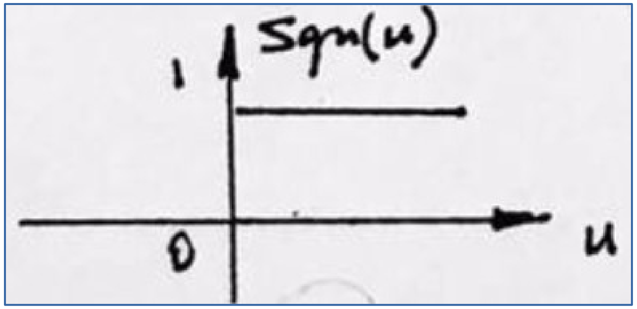
Road Map to LeNet (1)

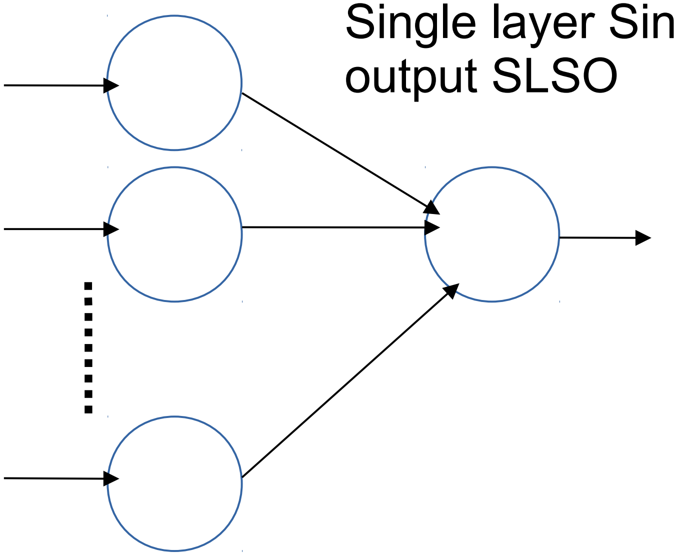


$$\boxed{y = f(\sum w_i x_i + \theta)} \quad \ldots (1)$$

Where

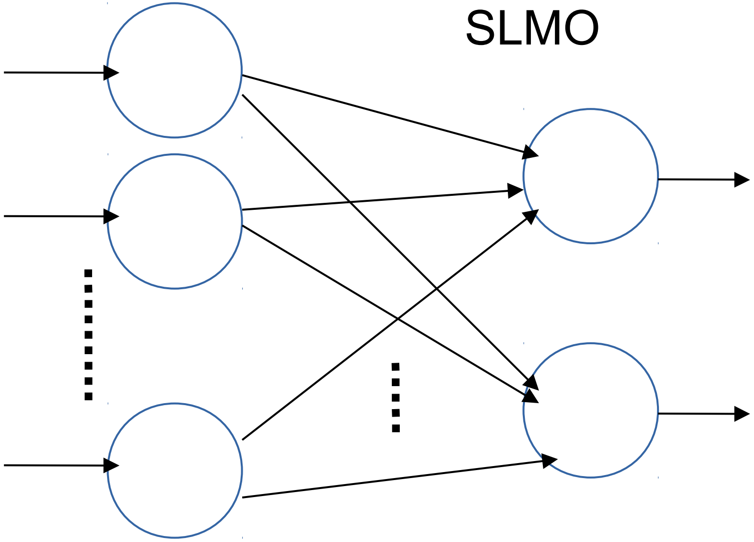
$$\boxed{y = f(\cdot) \text{ as } \mathrm{sgn}(\cdot)} \quad \ldots (2)$$



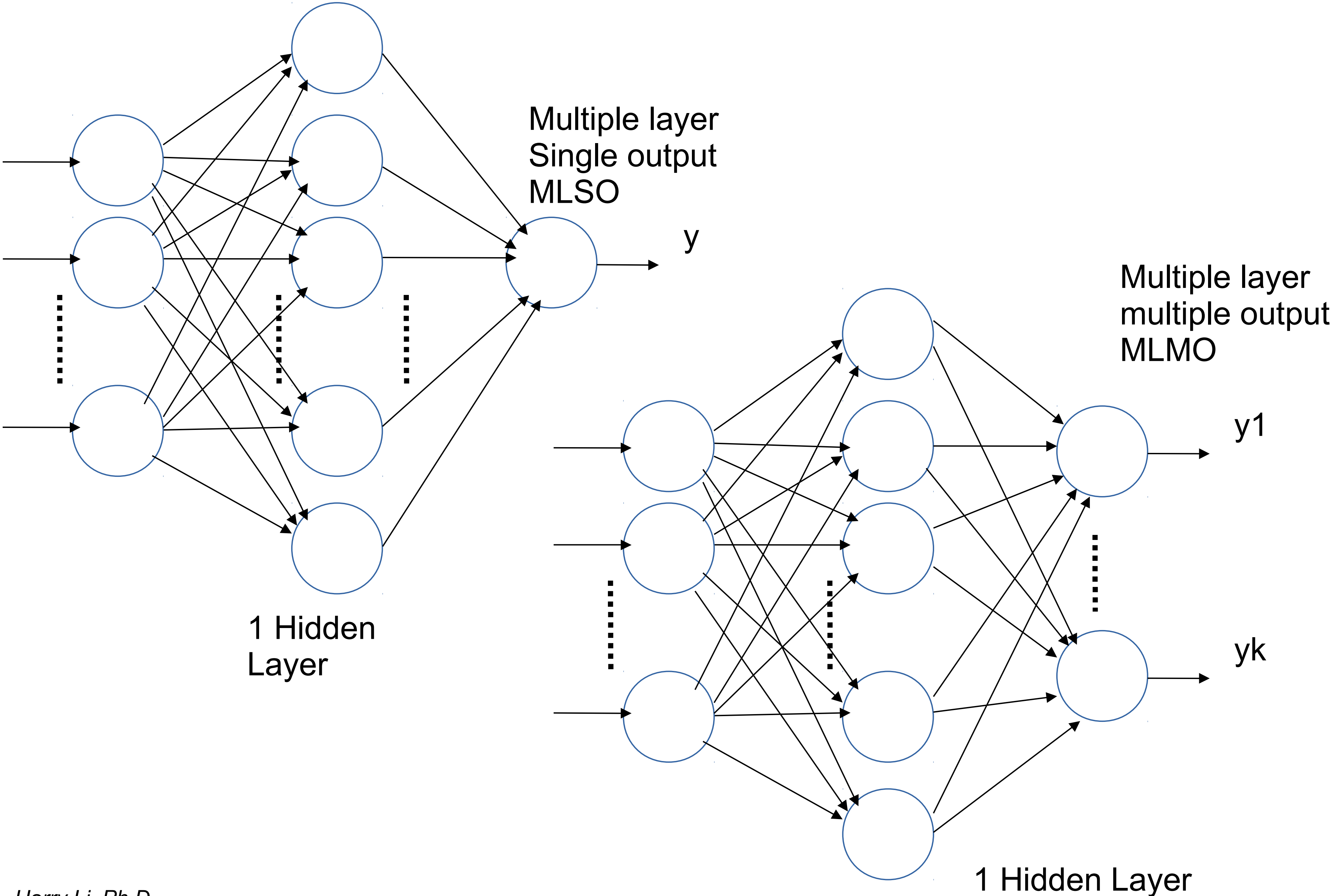


Single layer Single output SLSO

Single layer Multiple outputs SLMO



Road Map to LeNet (2)



Multiple layer
Single output
MLSO

y

1 Hidden
Layer

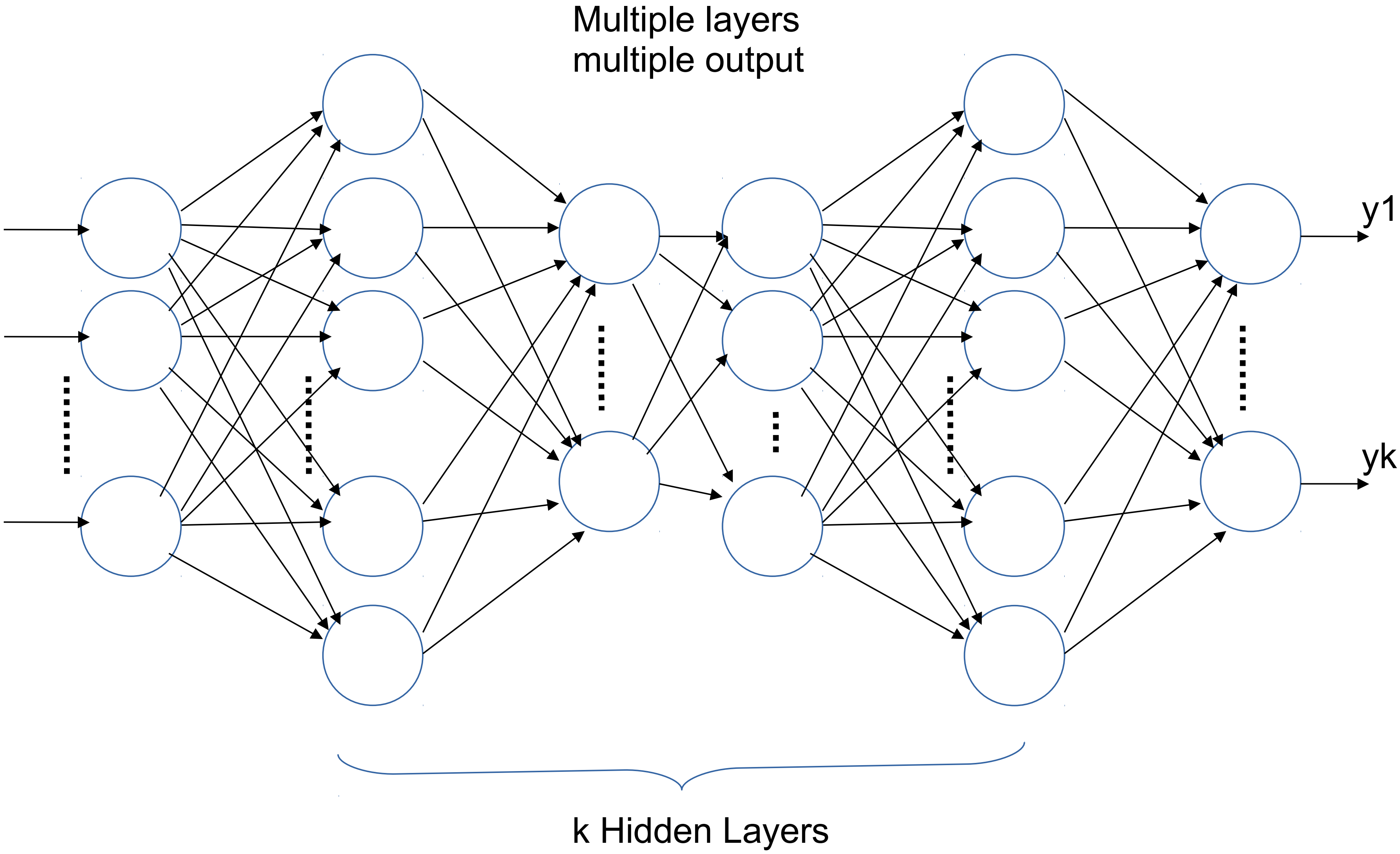
Multiple layer
multiple output
MLMO

y1

yk

1 Hidden Layer

Road Map to LeNet (3)



Example: 2D Convolution (1)

x

y

I_1(x,y)

100	100	0	0
100	100	0	0
100	100	0	0
100	100	0	0

k(x,y)

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

x

y

O_1(x,y)

0	-100	-100	0
0	-100	-100	0
0	-100	-100	0
0	-100	-100	0

x

y

I_2(x,y)

100	100	100	0
100	100	0	0
100	100	0	0
100	100	0	0

k(x,y)

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

x

y

O_2(x,y)

0	0	-100	-100
0	-100	-100	0
0	-100	-100	0
0	-100	-100	0

2D Convolution (2)

x, y

I_3(x,y)

100	100	100	0
100	100	100	0
100	100	0	0
100	100	0	0

k(x,y)

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

x, y

O_3(x,y)

0	0	-100	-100
0	0	-100	-100
0	-100	-100	0
0	-100	-100	0

x, y

I_4(x,y)

100	100	100	0
100	100	100	0
100	100	100	0
100	100	0	0

k(x,y)

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

x, y

O_4(x,y)

0	0	-100	-100
0	0	-100	-100
0	0	-100	-100
0	-100	-100	0

2D Convolution (3)

x

y

I_5(x,y)

100	100	100	0
100	100	100	0
100	100	100	0
100	100	100	0

k(x,y)

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

x

y

O_5(x,y)

0	0	-100	-100
0	0	-100	-100
0	0	-100	-100
0	0	-100	-100

x

y

I_6(x,y)

0	100	100	0
100	100	100	0
100	100	100	0
100	100	100	0

k(x,y)

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

x

y

O_6(x,y)

100	100	-100	-100
0	0	-100	-100
0	0	-100	-100
0	0	-100	-100

2D Convolution (4)

Version 2.0; Nov. 3rd, 2017

x

y

I_7(x,y)

0	100	100	0
0	100	100	0
100	100	100	0
100	100	100	0

k(x,y)

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

x

y

O_7(x,y)

100	100	-100	-100
100	100	-100	-100
0	0	-100	-100
0	0	-100	-100

x

y

I_8(x,y)

0	100	100	0
0	100	100	0
100	100	100	0
100	100	100	0

k(x,y)

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

x

y

O_8(x,y)

100	100	-100	-100
100	100	-100	-100
0	0	-100	-100
0	0	-100	-100

Note: I_7(x,y) and I_8(x,y) are the same. Typo.

2D Convolution (5)

x

y

I_9(x,y)

0	100	100	0
0	100	100	0
0	100	100	0
100	100	100	0

k(x,y)

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

x

y

O_9(x,y)

100	100	-100	-100
100	100	-100	-100
100	100	-100	-100
0	0	-100	-100

x

y

I_10(x,y)

0	100	100	0
0	100	100	0
0	100	100	0
0	100	100	0

k(x,y)

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

x

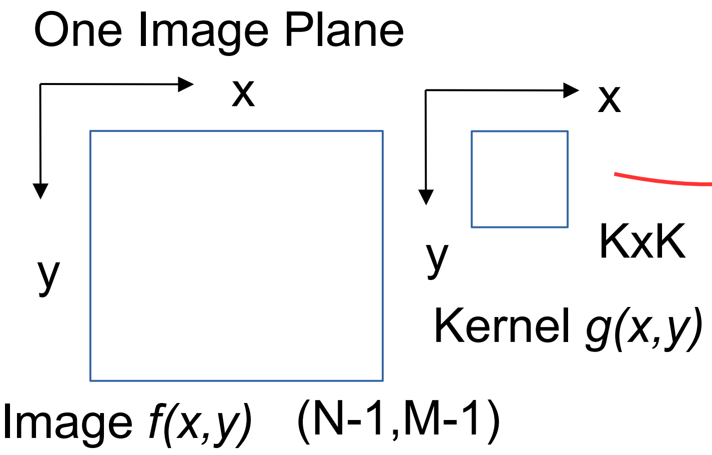
y

O_10(x,y)

100	100	-100	-100
100	100	-100	-100
100	100	-100	-100
100	100	-100	-100

Kernel Coefficients to Neural Nets

Version 2.0; Nov. 3rd, 2017



One Image Plane

x

y

Image *f(x,y)* (N-1,M-1)

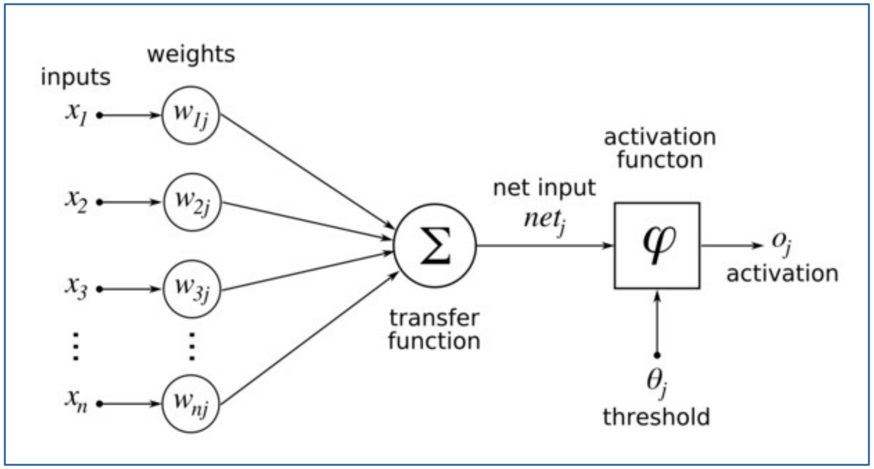
x

y

KxK

Kernel *g(x,y)*

w_11	w_12	w_13
w_21	w_22	w_23
w_31	w_32	w_33



Input from image I(x,y)	weight	Output O(x,y)
x_11	w_11	O(x,y) at w_22 location
x_12	w_12	
x_13	w_13	
x_21	w_21	
x_22	w_22	
x_23	w_23	
x_31	w_31	
x_32	w_32	
x_33	w_33	

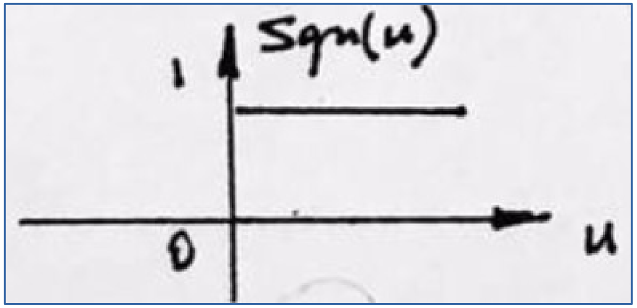
Modify The SLSO (1)

Version 2.0; Nov. 3rd, 2017

$$y = f\left(\sum w_i x_i + \theta\right)$$ … (1)

Where

$$y = f(\cdot) \text{ as } Sgn(\cdot)$$ … (2)



Previously,

Group 1, C1, as those from edge pixels

Group 2, C2, as those from non-edge pixels

Change to

Group 1, C1, from +100 edge pixel

Group 2, C2, from -100 edge,
Group 3, C3 from non-edge pixels

Hence, for the following

100	100	0	0
100	100	0	0
100	100	0	0
100	100	0	0

0	-100	-100	0
0	-100	-100	0
0	-100	-100	0
0	-100	-100	0

(1,1), y = 0; (1,2), y=1; (1,3), y=1; (1,4), y=0
(2,1), y = 0; (2,2), y=1; (2,3), y=1; (2,4), y=0
(3,1), y = 0; (3,2), y=1; (3,3), y=1; (3,4), y=0
(4,1), y = 0; (4,2), y=1; (4,3), y=1; (4,4), y=0

Changed to:

(1,1), y1y2: 00; (1,2), 01; (1,3), 01; (1,4), 00
(2,1), 00; (2,2),01; (2,3), 01; (2,4),00
(3,1), 00; (3,2), 01; (3,3),01; (3,4),00
(4,1), 00; (4,2), 01; (4,3), 01;(4,4),00

Change SLSO to SLMO

From the following,

0	100	100	0
0	100	100	0
0	100	100	0
0	100	100	0

100	100	-100	-100
100	100	-100	-100
100	100	-100	-100
100	100	-100	-100

(1,1), y = 1; (1,2), y=1; (1,3), y=1; (1,4), y=1
(2,1), y = 1; (2,2), y=1; (2,3), y=1; (2,4), y=1
(3,1), y = 1; (3,2), y=1; (3,3), y=1; (3,4), y=1
(4,1), y = 1; (4,2), y=1; (4,3), y=1; (4,4), y=1

SLSO

The training algorithm:

$$w_i^{+} = w_i^{-} + \eta(d-y)x_i$$

where

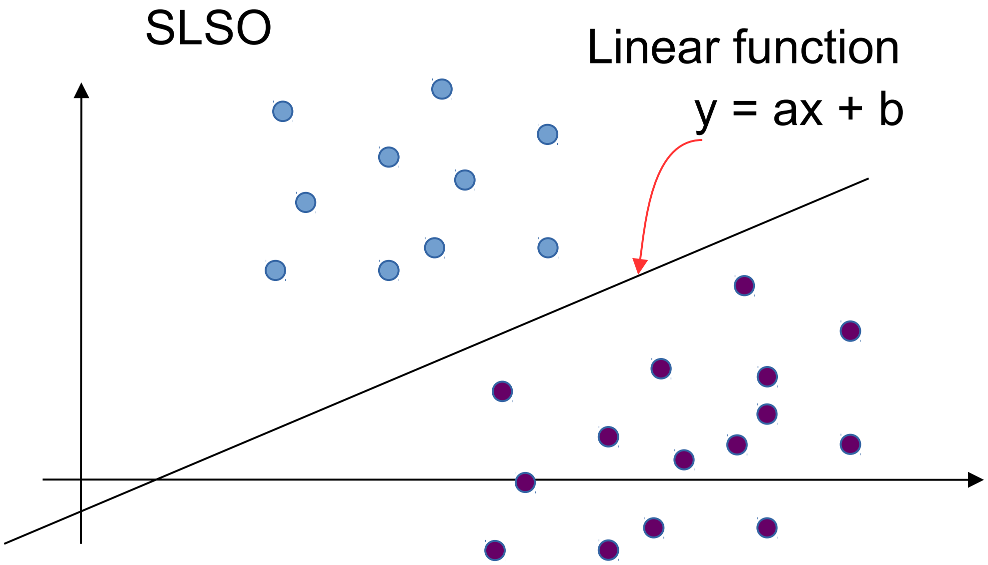
η is gain.
d : desired the output.
y: actual output.

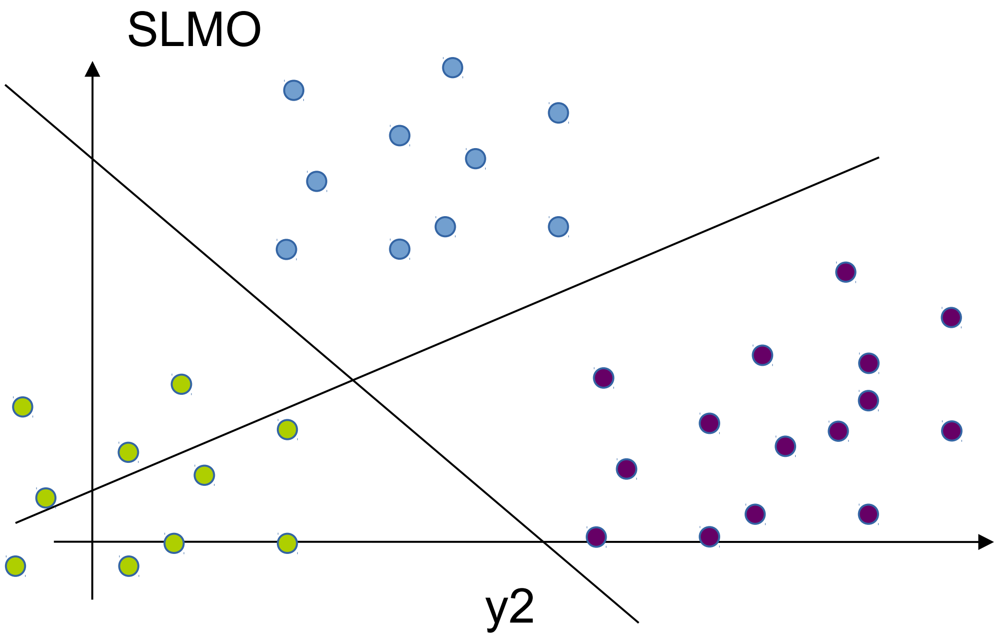
SLMO

(1,1), 01; (1,2), 01; (1,3), 10; (1,4), 10
(2,1), 01; (2,2), 01; (2,3), 10; (2,4), 10
(3,1), 01; (3,2), 01; (3,3), 10; (3,4), 10
(4,1), 01; (4,2), 01; (4,3), 10; (4,4), 10

SLSO vs. SLMO

Linear Decision Function





Remark 1: The single layer neural network decision making function is a linear function.

Versification: remember equation (1) from the previous section, we have

$$\boxed{y = f\left(\Sigma w_i x_i + \theta\right)} \quad \ldots (1)$$

Clearly,

$$\Sigma w_i x_i + \theta$$

Is a linear function (line, plane or in higher dimension), while f( ) an activation function plays a role of moving the linear function up or down.