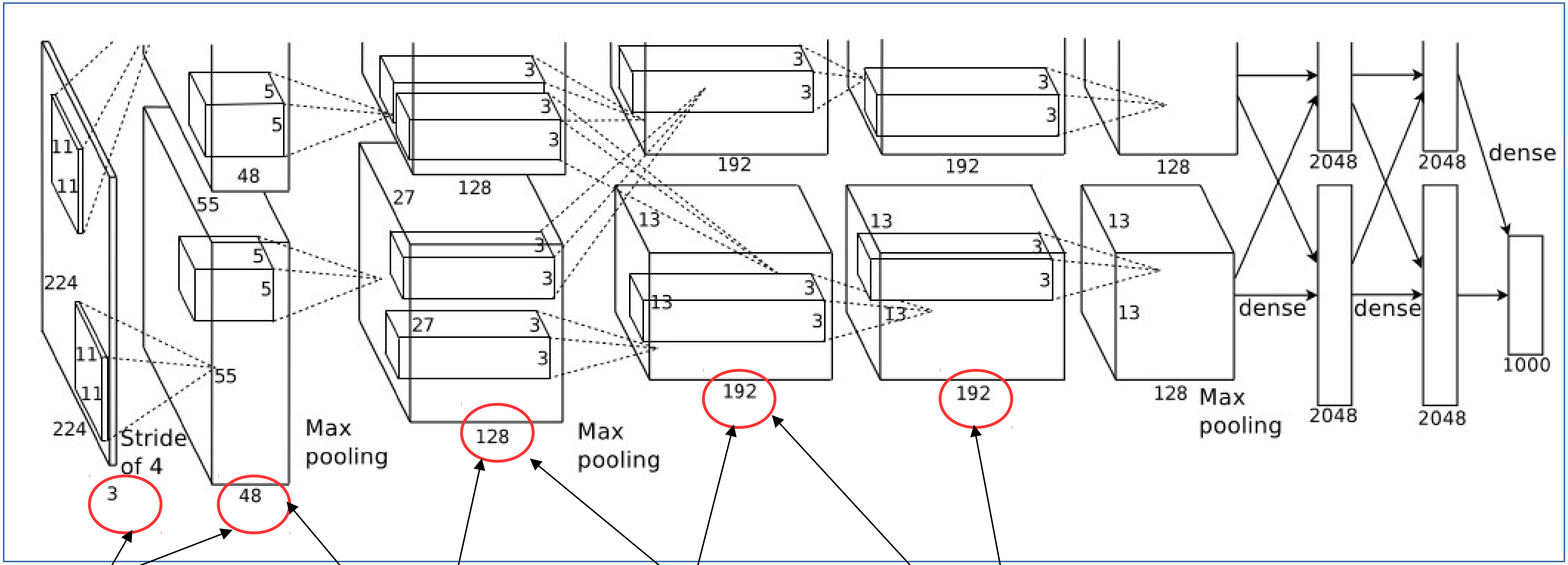
Reference: Alex Net

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

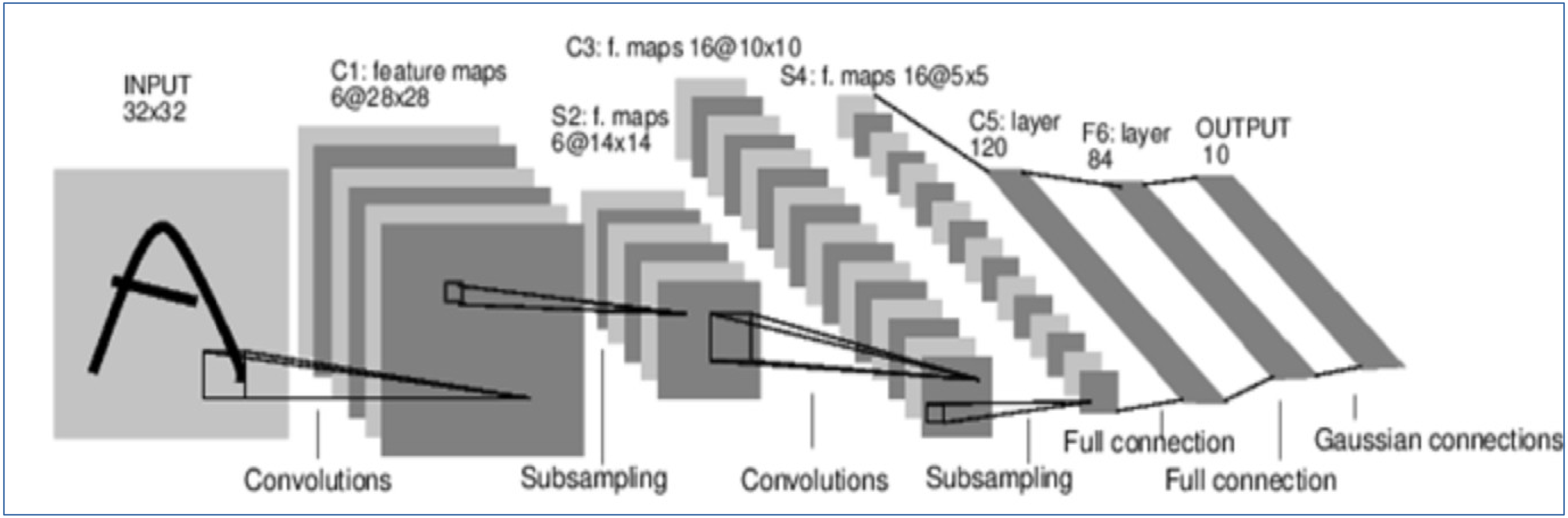
Example 2: 128 kernels, the depth of the kernel is 48

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

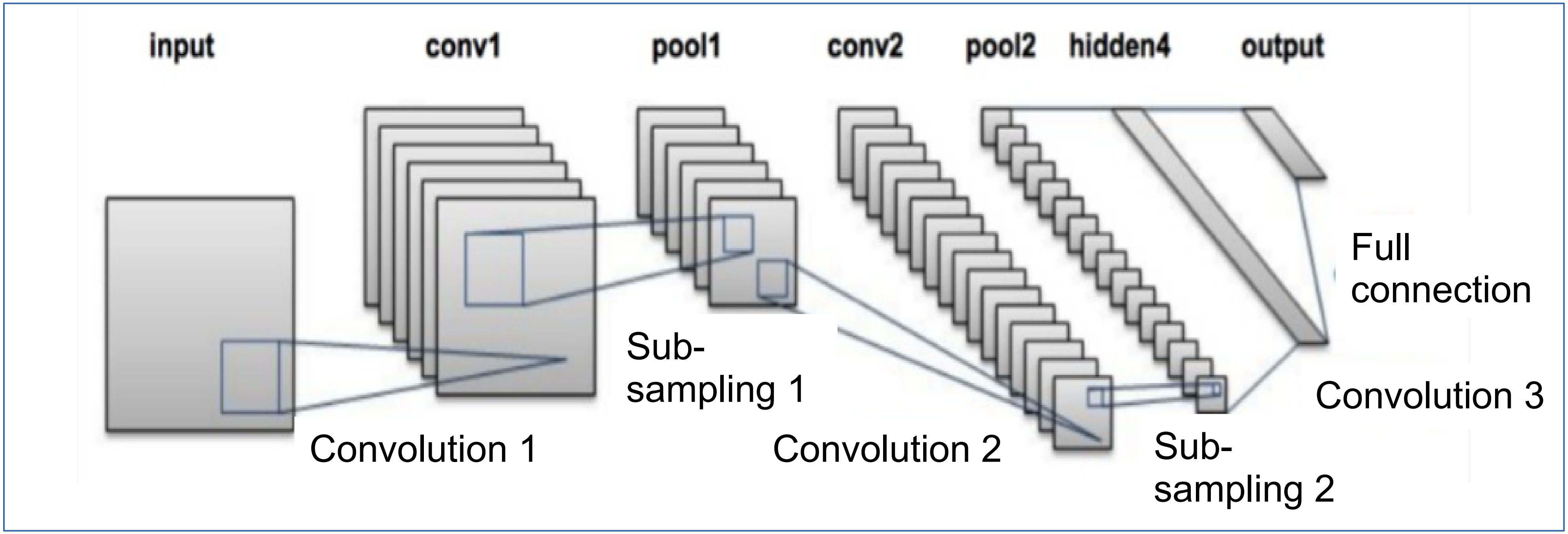
http://timdettmers.com/2015/03/26/convolution-deep-learning/



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

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	

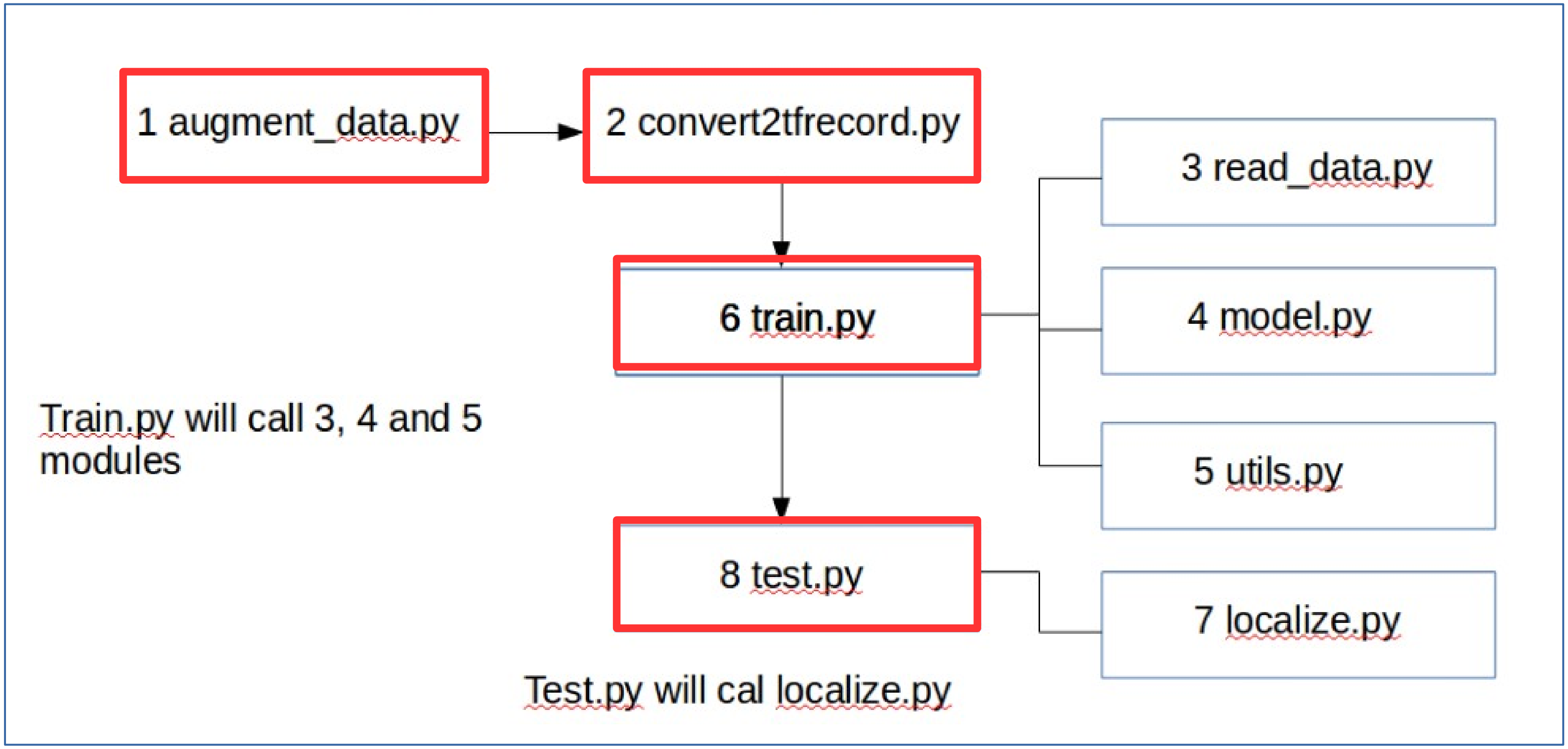
Harry Li, Ph.D.

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

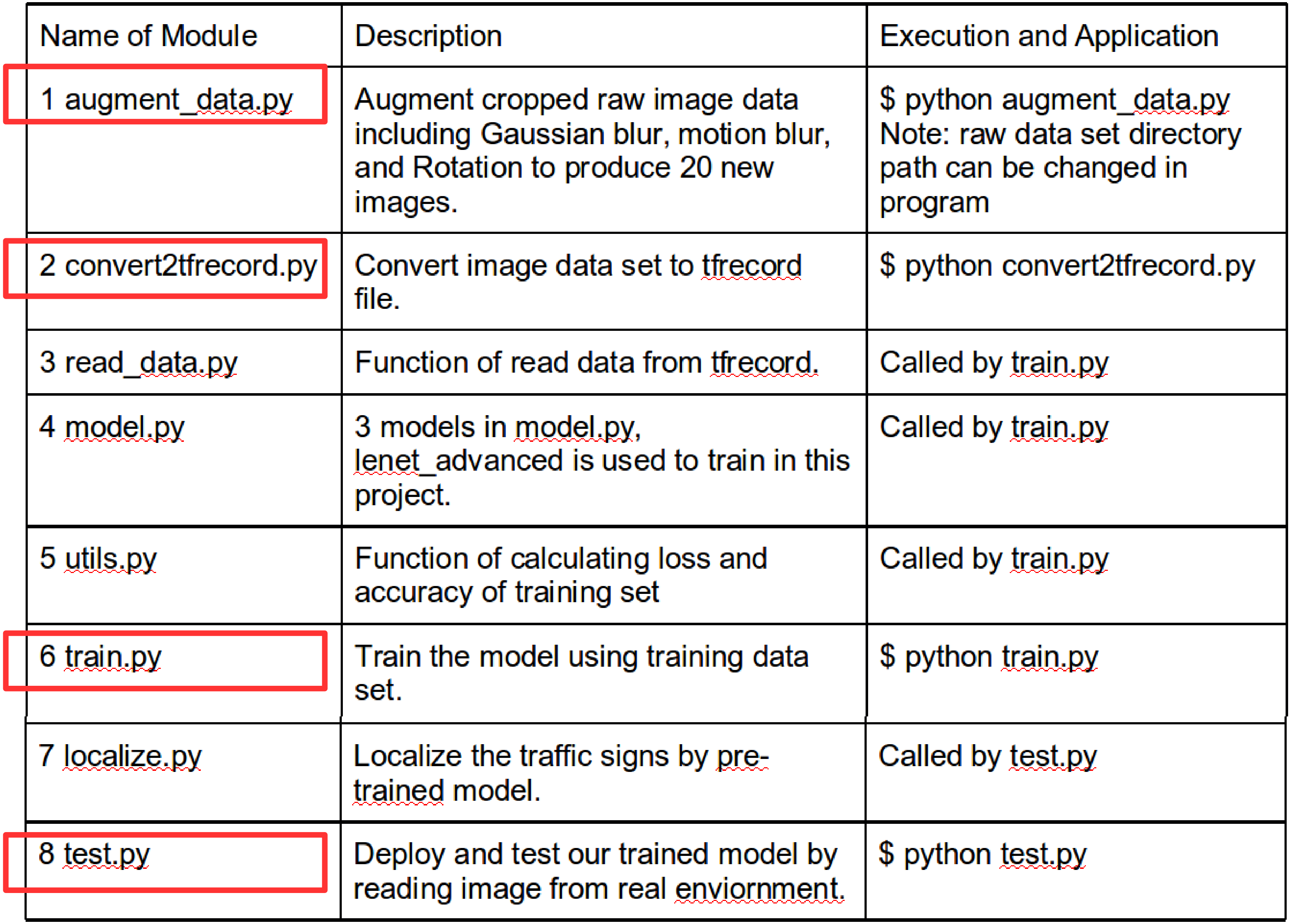


Train.py will call 3, 4 and 5 modules

Test.py will cal localize.py

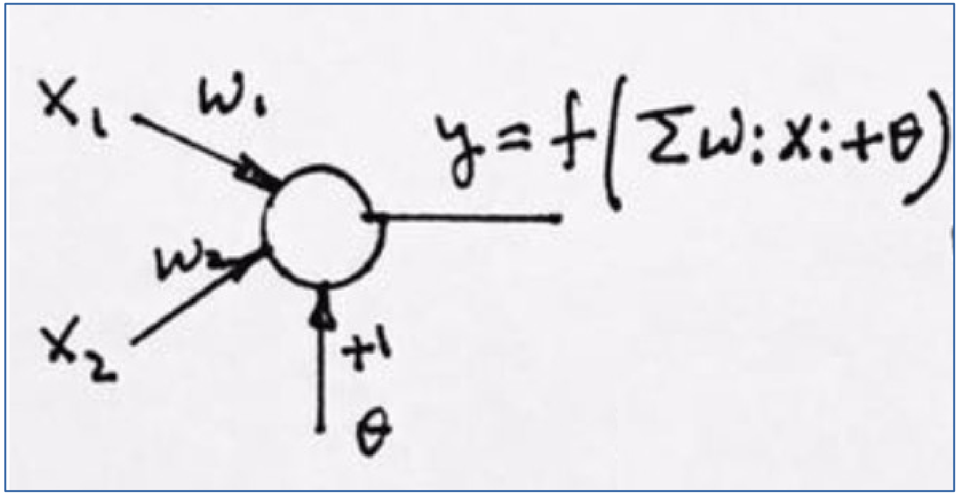
CTI One Production Sample Code

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

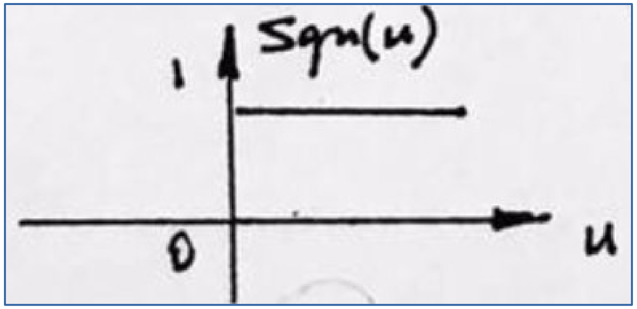
Road Map to LeNet (1)

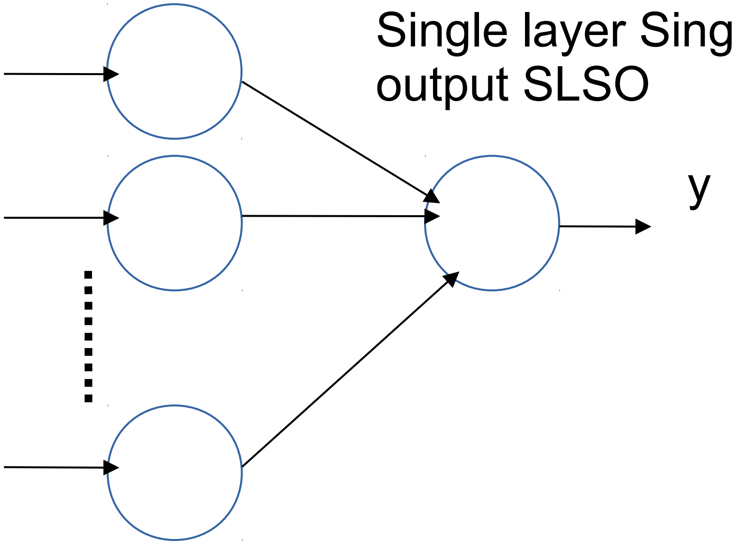


$$y=f(\Sigma w_i x_i+\theta) \quad \ldots (1)$$

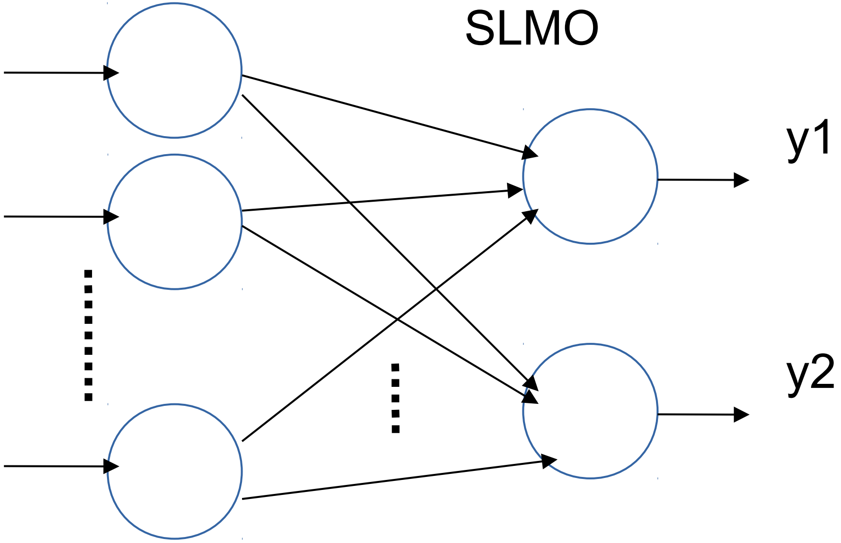
Where

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



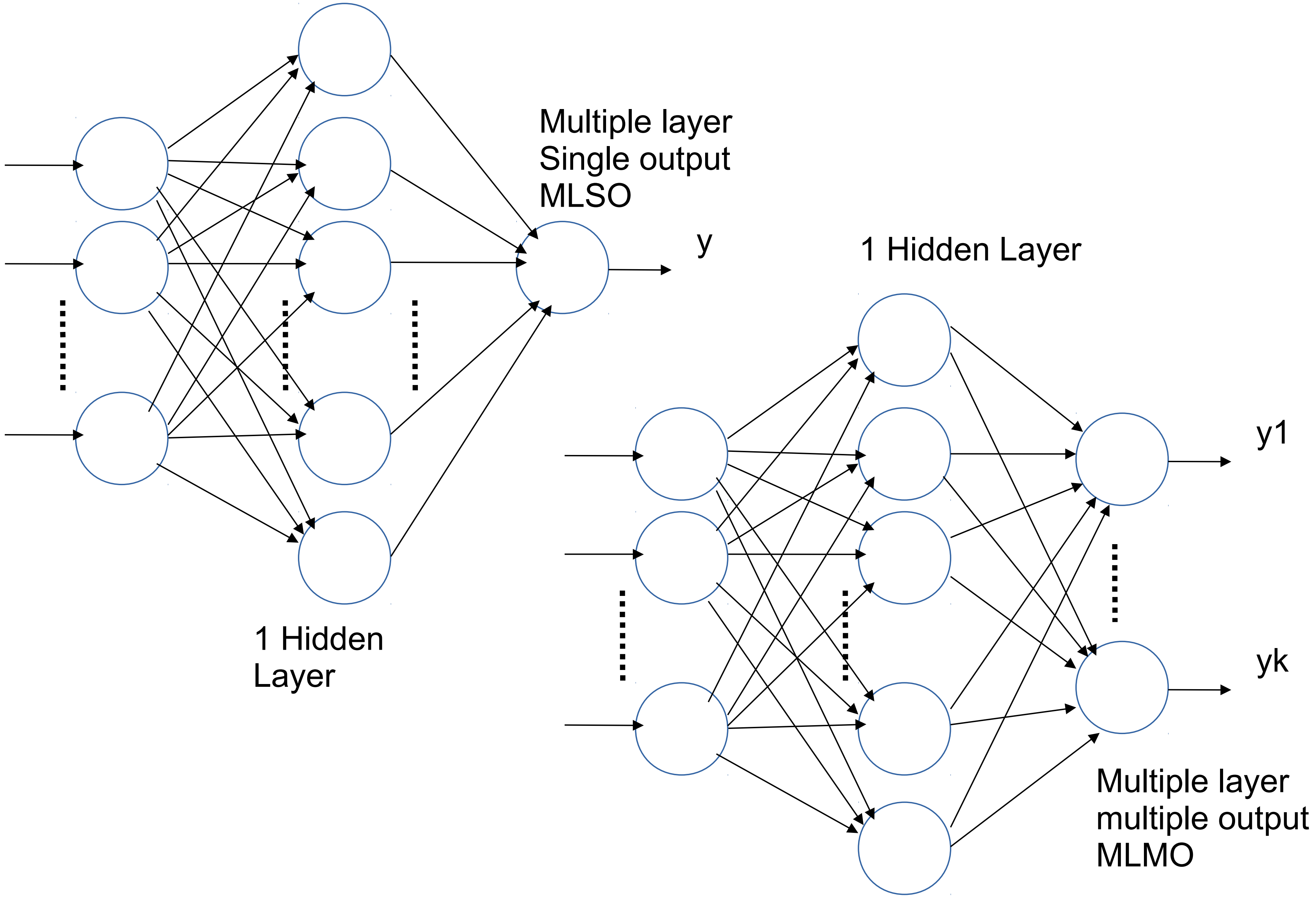


Single layer Single output SLSO



Single layer Multiple outputs SLMO

Road Map to LeNet (2)



Multiple layer
Single output
MLSO

y

1 Hidden
Layer

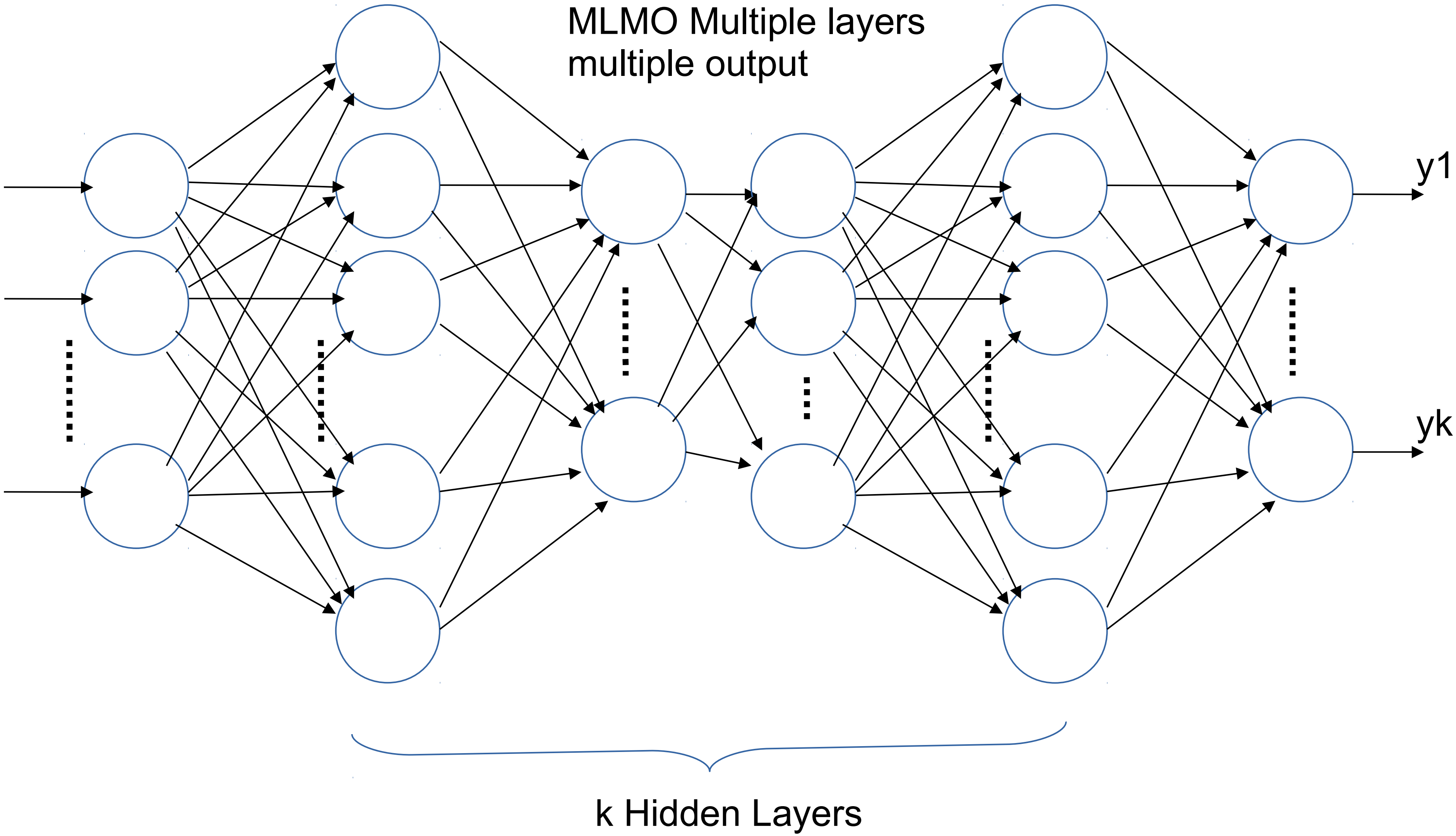
1 Hidden Layer

y1

yk

Multiple layer
multiple output
MLMO

Road Map to LeNet (3)



Harry Li, Ph.D.

Python Implementation with numpy

1

```
#------------------------------------------------------------------------------#
# Program: 106-pytest86-edgefinder4.py;                                        #
# Coded by: Tony Xu;              Date: Nov. 3rd, 2017;                        #
# Network Designed by: Harry Li, Ph.D. CTI One Corp.                           #
# Version: x01.0;                                                              #
# Status:  Tested.                                                             #
# Note: Set output image pixel value 100, -100 and 0, as 1,#
#       -1 and 0, so the output is 2 nodes, and this model                     #
#       converges.                                                             #
#------------------------------------------------------------------------------#
```

Always starts your program with a well structured header like the one.

2

```
import numpy as np
learning_rate = 0.001
momentum = 0.9
```

Define learning rate and momentum. See the updating formula next page

```
epochs = 10000 # Number of iterations
stopError = 0.001
#define layout of network.
inputLayerSize, hiddenLayerSize, outputLayerSize = 27, 200, 2
```

3

Define MLMO archiecture

4

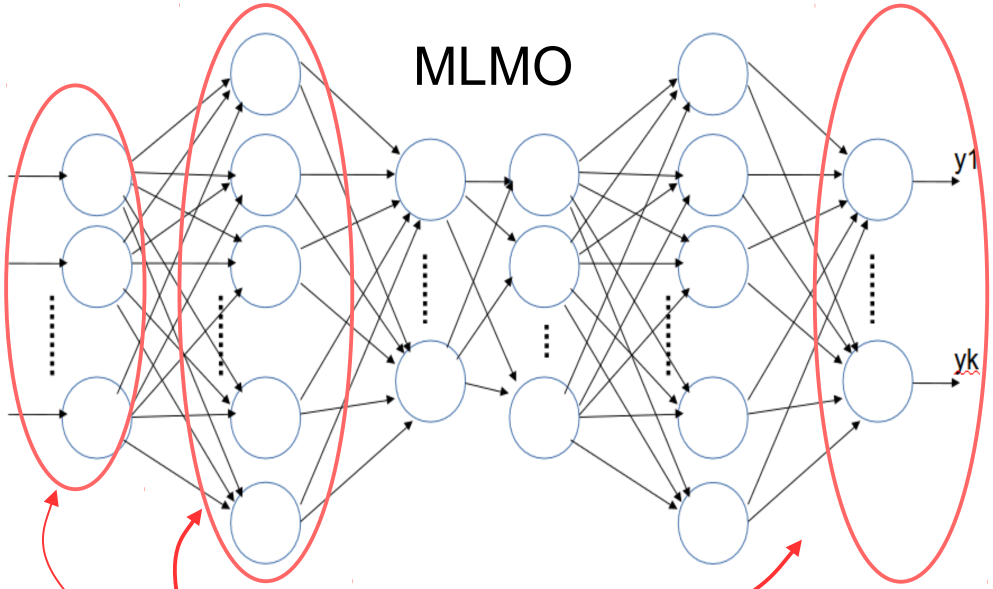
```
imageSize = 4
kernelSize = 3
expandSize = 6   #imageSize + 2 * (kernelSize/2)
```

Python Code

2

NumPy is a library for the Python programming language: (1) adding support for large, multi-dimensional arrays and matrices, and (2) large collection of high-level mathematical functions. BSD-new license Stable release: 1.13.3 (September 2017); Initial release: As Numeric, 1995; as NumPy, 2006.

https://docs.scipy.org/doc/numpy-1.13.0/index.html Numpy C-API
https://docs.scipy.org/doc/numpy-1.13.0/reference/c-api.html



3

Define MLMO architecture as:

```
inputLayerSize, hiddenLayerSize, outputLayerSize = 27, 200, 2
```

Harry Li, Ph.D.

Connecting to Our Image Data

4

```
imageSize = 4
kernelSize = 3
expandSize = 6   #imageSize + 2 * (kernelSize-
1)/2
```

x

y

I_1(x,y)

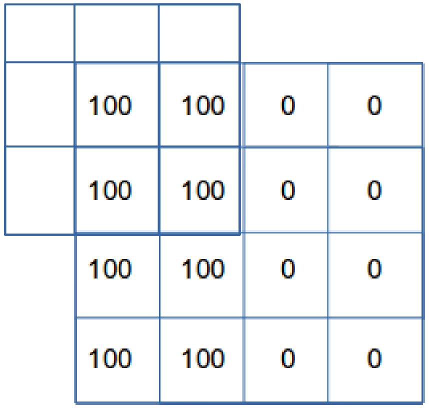
100	100	0	0
100	100	0	0
100	100	0	0
100	100	0	0

Image

k(x,y)

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

Kernel



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

Expand for convolution

Harry Li, Ph.D.

Python Code Defines Test Image Data

5

```
inputImagelist  = [
            [100,100,0,0,    100,100,0,0,    100,100,0,0,     100,100,0,0],
            [100,100,100,0,  100,100,0,0,    100,100,0,0,     100,100,0,0],
            [100,100,100,0,  100,100,100,0,  100,100,0,0,     100,100,0,0],
            [100,100,100,0,  100,100,100,0,  100,100,100,0,   100,100,0,0],
            [100,100,100,0,  100,100,100,0,  100,100,100,0,   100,100,100,0],
            [0,100,100,0,    100,100,100,0,  100,100,100,0,   100,100,100,0],
            [0,100,100,0,    0,100,100,0,    100,100,100,0,   100,100,100,0],
            [0,100,100,0,    0,100,100,0,    0,100, 100, 0,   100,100,100,0],
            [0,100,100,0,    0,100,100,0,    0,100, 100, 0,   0,  100,100,0]
            ]
desiredValues = [
           [0,-1,-1,0,  0,-1,-1,0, 0,-1,-1,0,  0,-1,-1,0],
           [0,0,-1,-1,  0,-1,-1,0, 0,-1,-1,0,  0,-1,-1,0],
           [0,0,-1,-1,  0,0,-1,-1, 0,-1,-1,0,  0,-1,-1,0],
           [0,0,-1,-1,  0,0,-1,-1, 0,0,-1,-1,  0,-1,-1,0],
           [0,0,-1,-1,  0,0,-1,-1, 0,0,-1,-1,  0,0,-1,-1],
           [1,1,-1,-1,  0,0,-1,-1, 0,0,-1,-1,  0,0,-1,-1],
           [1,1,-1,-1,  1,1,-1,-1, 0,0,-1,-1,  0,0,-1,-1],
           [1,1,-1,-1,  1,1,-1,-1, 1,1,-1,-1,  0,0,-1,-1],
           [1,1,-1,-1,  1,1,-1,-1, 1,1,-1,-1,  1,1,-1,-1]
           ]
```

x

y

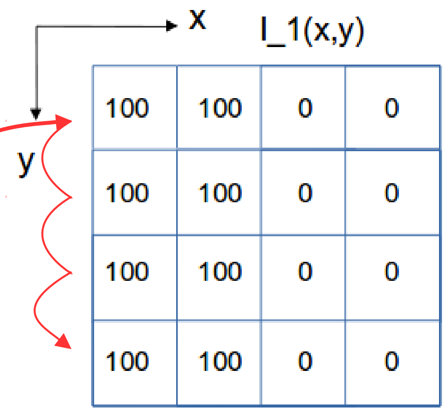
I_1(x,y)

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

Python Code Function Definition

```
def getsubImage(image, px, py):
    myElements = []
    startPosition = expandSize * py + px
    for i in range(kernelSize):
        for j in range(kernelSize):
            myElements.append(image[startPosition + i * expandSize + j])
    return myElements
```

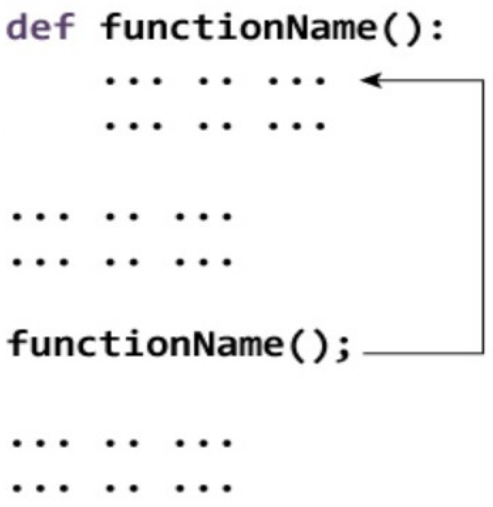


100	100	0	0
100	100	0	0
100	100	0	0
100	100	0	0

Syntax of Function Definition for Python

```
def function_name(parameters):
        """docstring"""
        statement(s)
```

https://www.programiz.com/python-programming/function



```
def functionName():
    ... .. ...
    ... .. ...

... .. ...
... .. ...

functionName();

... .. ...
... .. ...
```

Note: (1) def Keyword marks the start of function header.

(2) Parameters (arguments) if any;

(3) A colon (:) to mark the end of function header.

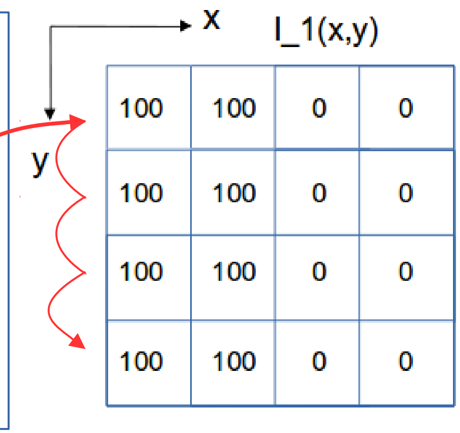
(4) Optional documentation string (docstring) to describe the function.

(5) Valid python statements make up the function body with same indentation level (usually 4 spaces).

(6) An optional return statement to return a value from the function.

Python Code “for” Loop

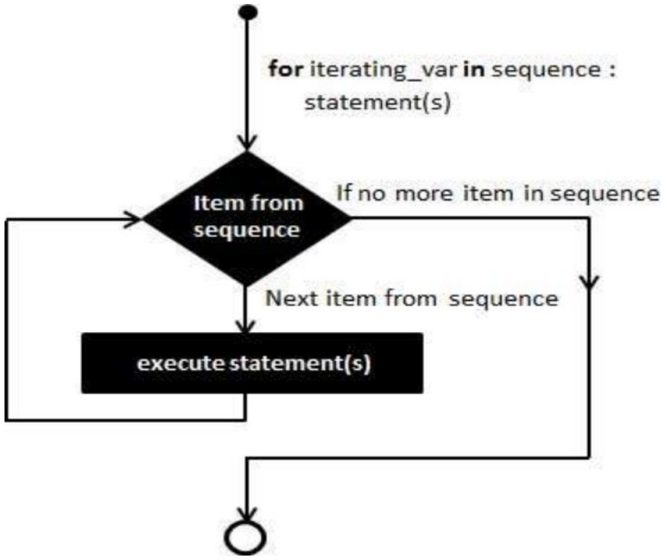
```
def getsubImage(image, px, py):
    myElements = []
    startPosition = expandSize * py + px
    for i in range(kernelSize):
        for j in range(kernelSize):
            myElements.append(image[startPosition + i * expandSize + j])
    return myElements
```



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

Syntax

```
for iterating_var in sequence:
   statements(s)
```



Note:
(1) A sequence, e.g., an expression list is evaluated. Then, the first item in the sequence is assigned to the iterating variable.
(2) then the statements block is executed.
(3) Each item in the list is assigned to iterating_var, and the statement(s) block is executed until the entire sequence is exhausted.

https://www.tutorialspoint.com/python/python_for_loop.htm

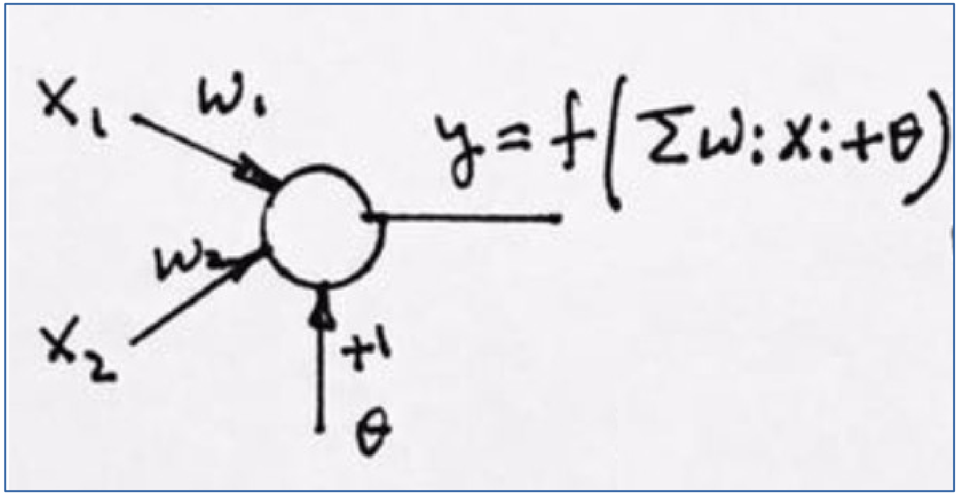
Python Code Define Function

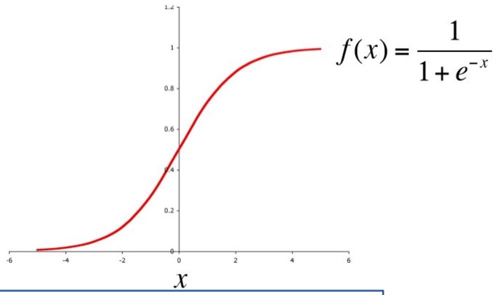
```
def expandImage(image):
    operationalInput = []
    inputImage = []
    #expand right and left intensity column
    for i in range(imageSize):
        for j in range(imageSize):
            if j == 0:
                operationalInput.append(image[i * imageSize + j])
            if j == imageSize - 1:
                operationalInput.append(image[i * imageSize + j])
            operationalInput.append(image[i * imageSize + j])

    #expand first intensity row
    for i in range(expandSize):
        inputImage.append(operationalInput[i])
    #keep middle rows
    for i in range(imageSize):
        for j in range(expandSize):
            inputImage.append(operationalInput[i * expandSize + j])
    #add last row
    for i in range(expandSize):
        inputImage.append(operationalInput[(imageSize-1) * expandSize + i])
    return inputImage
```

https://www.tutorialspoint.com/python/python_for_loop.htm

Python Code for Sigmoid Function

```
#activation function sigmoid
def sigmoid (x):
        return 1/(1 + np.exp(-x))

# derivative of sigmoid
def sigmoid_derivative(x):
        return x * (1 - x)
```



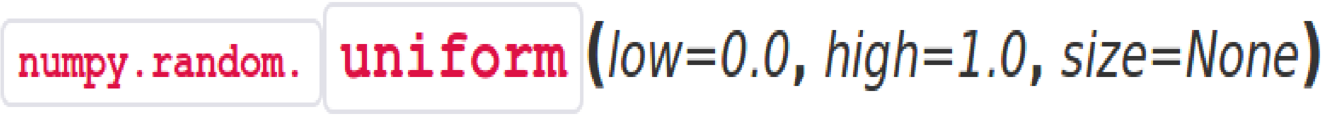


Python Code for Weights Init

```
#init weights and old weights
Weightsi_h = np.random.uniform(-1, 1, size=(inputLayerSize, hiddenLayerSize))
lastdeltaWeightsi_h =np.zeros(shape=(inputLayerSize, hiddenLayerSize))

Weightsh_o = np.random.uniform(-1, 1, size=(hiddenLayerSize,outputLayerSize))
lastdeltaWeightsh_o = np.zeros(shape=(hiddenLayerSize,outputLayerSize))
```

np.random.uniform and np.zeros

```
#init weights and old weights
Weightsi_h = np.random.uniform(-1, 1, size=(inputLayerSize, hiddenLayerSize))
lastdeltaWeightsi_h =np.zeros(shape=(inputLayerSize, hiddenLayerSize))
```



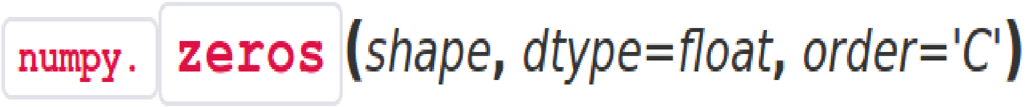
`numpy.random.uniform(low=0.0, high=1.0, size=None)`

Samples are uniformly distributed over the half-open interval [low, high) (includes low, but excludes high).

Note:

size=(inputLayerSize, hiddenLayerSize)

size : int or tuple of ints



`numpy.zeros(shape, dtype=float, order='C')`

Return a new array of given shape and type, filled with zeros.

shape : int or sequence of ints

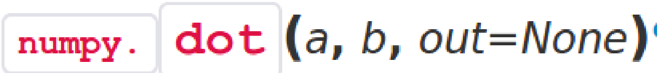
https://www.tutorialspoint.com/python/python_for_loop.htm

Back Prop

```
  for i in range(epochs):
    #forward path
    hiddenLayerOutput = sigmoid(np.dot(X, Weightsi_h))          6
    outputLayerOutput = sigmoid(np.dot(hiddenLayerOutput, Weightsh_o))
    #error (delta error) /cost /loss
    E = D - outputLayerOutput
    #determine if we have reached desired accuracy               7
    currentError = np.sum(np.square(E)) /float(len(X))
    if (currentError < stopError):
      print ("Reach to desired accuracy at epoch:", i)
      break
    #step 1, output layer's error
    dEatoutputLayer = E * sigmoid_derivative(outputLayerOutput)
    #step 2, determine delta weights between hidden layer and output layer
    dWeightsh_o = learning_rate * hiddenLayerOutput.T.dot(dEatoutputLayer) + momentum *
lastdeltaWeightsh_o
                                                                 8
    #save current to be old
    lastdeltaWeightsh_o = dWeightsh_o
    #step 3, caculate delta E for hidden layers
    dEathiddenLayer = dEatoutputLayer.dot(Weightsh_o.T) * sigmoid_derivative(hiddenLayerOutput)
    #determine delta weights between hidden layer and input layer
    dWeightsi_h = learning_rate * X.T.dot(dEathiddenLayer) + momentum * lastdeltaWeightsi_h
    lastdeltaWeightsi_h = dWeightsi_h
    #update output and hidden layer weights
    Weightsh_o +=  dWeightsh_o
    Weightsi_h +=  dWeightsi_h
```

Harry Li, Ph.D.   https://www.tutorialspoint.com/python/python_for_loop.htm

np.dot & np.sum & np.square

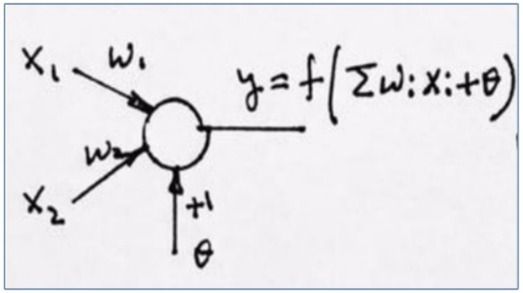


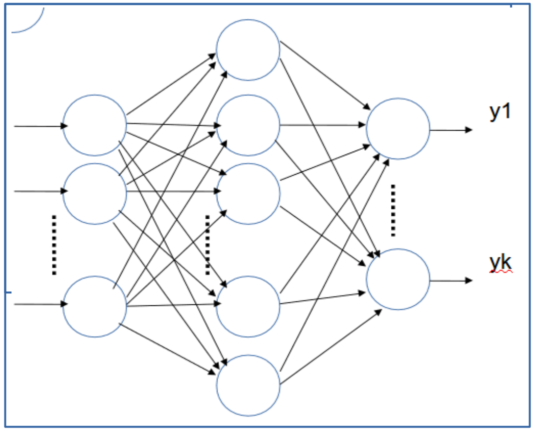
numpy.dot(a, b, out=None)

Dot product of two arrays.

For 2-D arrays it is equivalent to matrix multiplication, and for 1-D arrays to inner product of vectors (without complex conjugation). For N dimensions it is a sum product over the last axis of a and the second-to-last of b:

Example:
np.dot(X, Weightsi_h)





dWeightsh_o = learning_rate * hiddenLayerOutput.T.dot(dEatoutputLayer) + momentum * lastdeltaWeightsh_o

$$\boxed{W(t+1) = W(t) + \Delta W} \quad \ldots (3)$$

$$\boxed{\Delta W = -\eta * \partial E/\partial w} \quad \ldots (3.1)$$

https://www.tutorialspoint.com/python/python_for_loop.htm

Quiz (1)

1. what is NumPy? What language is it written for? What is the current release version?

2. How do you import numpy?

3. write one line of Python code to define MLMO feed-forward Neural Network architecture? Suppose there is one hidden layer and there are 100 nodes of the hidden layer? Input nodes = 20, and output nodes = 10?

4. How do you use def to define a function? Define a simple calculator function that is input arguments of 2 operand and 1 operator, such as addition, subtraction, multiplication, or division, and return the computed the result?

5. How is the for loop defined? Write a simple double for loop to read K by K image one pixel at time from left to right, top to bottom?

6. How would you call C function from python or from C call python?

7. How do you use np. to compute exp( ) function? Any other functions?

8. np.random.normal( ), np.zeros( ), and np.exp( ) etc.

Example: 2D Convolution (1)

x I_1(x,y)

y

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

x I_2(x,y)

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 O_1(x,y)

y

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

x O_2(x,y)

y

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

Verify this output by trained NN

Harry Li, Ph.D.

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

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_3(x,y)

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

x

y

O_4(x,y)

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

Verify this output by trained NN

Harry Li, Ph.D.

2D Convolution (3)

x

I_5(x,y)

y

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

x

O_5(x,y)

y

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

k(x,y)

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

x

I_6(x,y)

y

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

x

O_6(x,y)

y

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

Verify this output by trained NN

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

Verify this output by trained NN

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

x y O_10(x,y)

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

Verify this output by trained NN