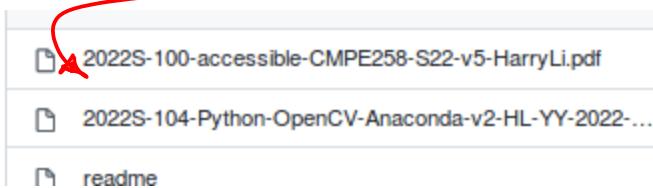


CMPE258 Spring2022

Feb1st. Organizational meeting.

1. Today's Topics "Greensheet"



Naming Convention Yr + Semester + ID
+ Name + Date

Contact Information: Email: hua.li@sjsu.edu

Text message to (650) 400-1116.

Office Hours: M.W. 4:30-5:30pm.

Zoom (link to be shared in the email)

Join from PC, Mac, Linux, iOS or Android: <https://sjsu.zoom.us/j/85616325978>

pwd=MzIRbDJXVHBDQ2gIU0RPM2tYc045Zz09

Password: 451032

On-Line materials on github

<https://github.com/hualili/opencv/tree/master/deep-learning-2020S>

<https://github.com/hualili/opencv/tree/master/deep-learning-2022s>

Also, CANVAS → mostly for Assignments and projects.

All Assignment / Projects are posted on Both github & SJSU CANVAS.

Lecture Material consists of P.P.T. posted on github, and Lecture Notes (White-Board Written Notes)

Core Emphases of the Class: Deep Convolutional

Neural Networks, And Their Application in Image Analysis, Video Analysis.

1. Text Book:

<https://github.com/hualili/opencv/blob/master/IP120-AI-DL/2018F/2018F-6-DeepLearningCh02.pdf>

2. Computer Vision Book By Horn

as a reference for Convolution & Image Segmentation, Contours Analysis (Binary Image).

3. OpenCV Reference Book (and Edition) together with Online Document (OpenCV).

Note: OpenGL (GL: Graphics Library) is just for Reference purpose, no need for this class. (but maybe helpful for the future research).

Unity is game Development Platform, interactive 3D Graphics Design platform.

Programming Languages:

1 Python. 3.6 or 3.7

2 C/C++ Feb. 8th.

Homework (Due A week from today)
No Submission. Submit A Screen

Capture that shows OpenCV installed successfully, with jpg or png file with Naming of the file as follows:

FirstName_LastName-SID-OpenCV.jpg

This Homework will be posted on CANVAS, Submission is on CANVAS.

Homework, Installation of Tensor Flow, Due 2 weeks
Feb 15th.

Submission: Screen Capture that shows the installation is successful. Submission on CANVAS.

Submit jpg, png file with the Naming convention as follows:

FirstName_LastName-SID-TF.jpg

Note: Optional, for Edge AI

Commenting, Consider using NVidia Jetson NAND (4GB) version.

5% Bonus

Grading:

Homework, projects : 30%

5% 25%

Project 1. Computer Vision for Preprocessing, plus Deep Convolutional Neural Nets
To give Real Time Detection Result of Last 4 Digits of a Student ID.

10%

Project 2. "Semester Long" project, with technical requirements (List)
Team project. 4 person Team.

Each person has clear definition of the tasks (Programming/Coding) And Balanced Contribution.

Final PPT, Demo Presentation
15%.

Midterm Exam: 30%.

Need to use your Laptop Computer, to Run/Execute code, modify the code.

Final 40%

Introduction

Topics: Neural Networks formulation
(Basic Building Blocks)
Digital Images/Videos.

Example: A Single Neuron Formulation
(Some kind Brain Cell)

Step 1. Summation function.



Fig 1.

" Σ " Summation function.

Note: $\sum_{k=1}^N x_k = x_1 + x_2 + \dots + x_N$

Step 2. Inputs

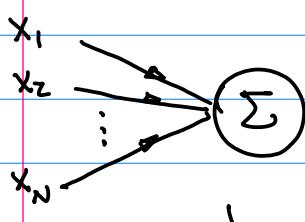


Fig 2.

Step 3. Weights (Knowledge)

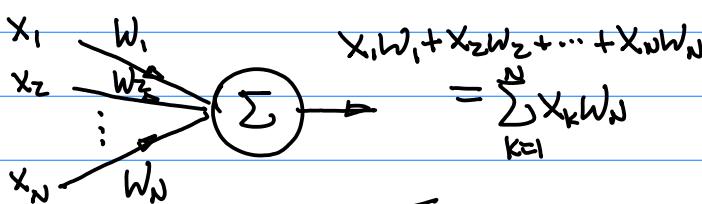


Fig 3.

$$\sum_{k=1}^N x_k w_k = x_1 w_1 + x_2 w_2 + \dots + x_N w_N \dots (1)$$

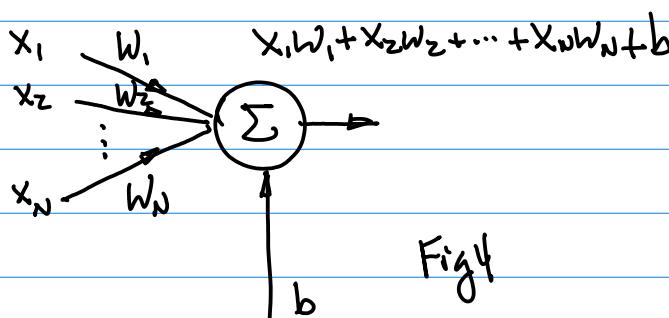


Fig 4

$$\sum_{k=1}^N x_k w_k = x_1 w_1 + x_2 w_2 + \dots + x_N w_N + b \quad \underline{\underline{}} \quad \dots (1b)$$

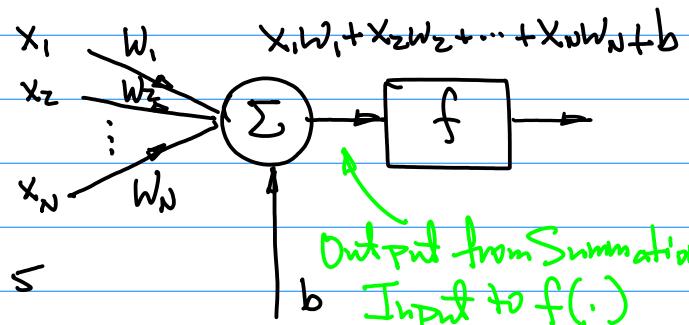


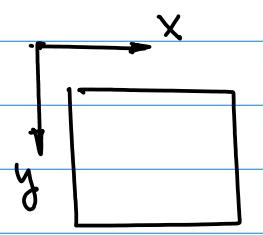
Fig 5

Note: Activation function f , denoted as $f(\cdot)$ (A function of Independent Variable " \cdot ", or A function of Input " \cdot ")

$$\begin{aligned} f(\cdot) &= f(x_1 w_1 + x_2 w_2 + \dots + x_N w_N + b) \\ &= f\left(\sum_{k=1}^N w_k x_k + b\right) \end{aligned} \quad \dots (2)$$

Summary: The output of a single is given by Eqn (2). Where Activation function $f(\cdot)$ can take different forms, it affects the Learning, Learning Speed.

Example: Digital Image, $I(x, y)$



$M \times N$

$I(x,y)$

Intensity, And/or

Color of An Image

(x, y) Location of A picture element, "pixel"

Example: Notations & Formulation

In Case of a Single pixel, (x, y) is the location of this pixel, I is Color/Intensity of the pixel

Note: For An Image $I(x,y)$

its features include Resolution $M \times N$

Pixel Depth:

bpp (Bit per pixel)

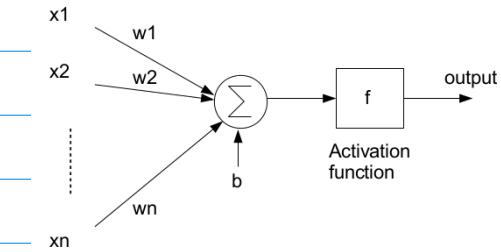


Fig.1

No. of pixels per Row

No. of Rows per frame

1. Notation for Input $x_i, i=1, 2, \dots, N$

$\{x_i | i=1, 2, \dots, N\} \dots (1)$

Vector form,

$(x_1, x_2, \dots, x_N) \dots (1b)$

$I(x,y)_{mn}$ OR $I(x,y)$

$\frac{1024 \times 768}{\text{pixels/Row}} = \text{Rows}$

For A color Image, A pixel depth very often is equal to 24 (bpp)

r, g, b Primitive color of red (r), green (g), blue (b) has 8 bits quantization level, e.g.

r: [0, 255], g: [0, 255], b: [0, 255].

Feb 8th (Tue)

Today's Topics : 1° Introduction, Basic Building Blocks, Math Formulation.
2° Sample Python Code for OpenCV.

Introduce Superscript j for Experiment j

Input $x_i^j, i=1, 2, \dots, N; j=1, 2, \dots, P$

Hence Eqn(1) Becomes

$\{x_i^j | i=1, 2, \dots, N; j=1, 2, \dots, P\}$

$(x_1^j, x_2^j, \dots, x_N^j)$ for Experiment j

2. Notation for Weight

$w_i, \text{ for } i=1, 2, \dots, N$

Hence,

$(w_1, w_2, \dots, w_N) \dots (2)$

$\downarrow W$

3. Inputs & weights

$$x_i \quad w_i$$

$$w_i x_i \text{ for } i=1, 2, \dots, N$$

$$(x_1, x_2, \dots, x_N) \cdot (w_1, w_2, \dots, w_N)$$

$$= w_1 x_1 + w_2 x_2 + \dots + w_N x_N = \sum_{i=1}^N w_i x_i$$

$$(w_1, w_2, \dots, w_N) \cdot (x_1, x_2, \dots, x_N)$$

$$= w_1 x_1 + w_2 x_2 + \dots + w_N x_N = \sum_{i=1}^N w_i x_i$$

4. Transfer Function, Denoted as

$$\tilde{h} \text{ OR } f(\cdot)$$

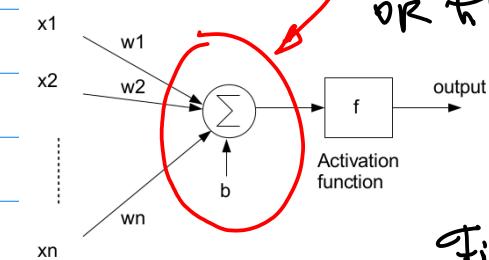


Fig. 2

$$h = \sum_{i=1}^N w_i x_i + b = w \cdot \vec{x} + b$$

... (3)

↑
Offset ("Bias")

\tilde{h} , or $h(\cdot)$, or Eqn (3), or
 $h(w_i; b)$, or $h(w_i)$

5. Activation Function: f

Acts like a switch, ON/OFF &

Attenuate the Output

$$f, f\left(\sum_{i=1}^N w_i x_i + b\right), \text{ or}$$

$$f(h(w_i; b)), \text{ or } f(h(\cdot))$$

The output of A Neuron is denoted as

$$y, \text{ and } y = f\left(\sum_{i=1}^N w_i x_i + b\right) \\ = f(h(w_i; b)) \quad \dots (4)$$

5. Outputs for A Neural Network =

$$y_k, \text{ for } k=1, 2, \dots, Q$$

if $Q=1$, y_1 , also better to just y , for a single neuron as y

Question: What output? whose output?

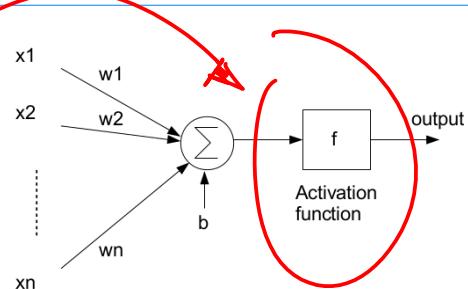
Output from the Network (or Network)

$$\tilde{y} \text{ (Tilde)}$$

Ground Truth is denoted as y

6. Loss function, Objective Function,

OR difference



"Supervised Learning"

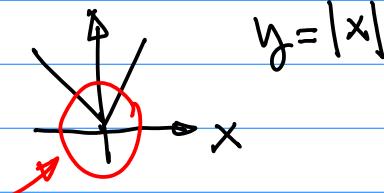
For Experiment j (multiple experiments)

$\tilde{y}_j^k - y_j^k$, $j=1, 2, \dots, p$... (5)
 Output from the NN (Neuron or Network) Ground Truth for the experiment
 for $j=1$, $\tilde{y}_1^k - y_1^k$
 $j=2$, $\tilde{y}_2^k - y_2^k$
 :
 $j=p$, $\tilde{y}_p^k - y_p^k$

Therefore, put all these differences (Loss) together

$$(\tilde{y}_1^k - y_1^k) + (\tilde{y}_2^k - y_2^k) + \dots + (\tilde{y}_p^k - y_p^k) \\ = \sum_{k=1}^p (\tilde{y}_k^k - y_k^k) \quad \dots (b)$$

Note:



To deal with the issue of Absolute Value of a function, Let's square it.

Hence Eqn (b) becomes

$$\sum_{k=1}^p (\tilde{y}_k^k - y_k^k)^2 \quad \dots (bb)$$

7. Objective function

$$L \triangleq \sum_{k=1}^p (\tilde{y}_k^k - y_k^k)^2 \quad \dots (7)$$

OR,

$$L(w_i) \triangleq \sum_{k=1}^p (\tilde{y}_k^k - y_k^k) \quad \dots (7b)$$

$$= \sum_{k=1}^p (\tilde{y}_k^k (h(w_i)) - y_k^k) \quad \dots (7c)$$

8. To generalize the result in Eqn

(7c) for multiple output, we have the following formulation.

From Eqn (7)

$$\sum_{k=1}^p (\tilde{y}_{k_1}^k - y_{k_1}^k)^2$$

\uparrow \uparrow
 k_1 : Output k_1 for multiple output

$$\tilde{y}_j^k, j=1, 2, \dots, k_2, k_2+1, \dots, M$$

We can count All outputs loss.

$$\sum_{k=1}^p (\tilde{y}_{j_1}^k - y_{j_1}^k)^2, \sum_{k=1}^p (\tilde{y}_{j_2}^k - y_{j_2}^k)^2, \dots \\ \dots \sum_{k=1}^p (\tilde{y}_{j_M}^k - y_{j_M}^k)^2$$

Hence

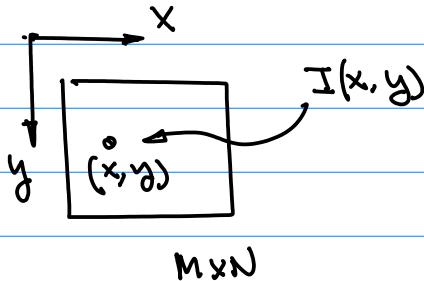
$$\sum_{k_2=1}^m \sum_{k_1=1}^p (\tilde{y}_{j_{k_2}}^{k_1} - y_{j_{k_2}}^{k_1})^2 \quad \dots (8)$$

CMPE258 Feb 8, 22

$$L(w_i) \triangleq \frac{1}{2} \sum_{h_2=1}^m \sum_{k_2=1}^p (\hat{y}_{j_{h_2}}^{k_2} - y_{j_{h_2}}^{k_2})^2$$

... (8b)

Example: Digital Image $I(x, y)$



Installation of OpenCV → Python OR
(C/C++, But
Python is Better)

Different Packages
for ML/DL, may need different version
of Python, And different packages

Anaconda is a well developed,
Adopted tool for Package management.

1. Check github Class Reference

2022S-104a-Python-OpenCV-Anaconda-v2-HL-YY-2022...

Note: For installation of Anaconda,
Check github document,
2022S-104-~

Once Anaconda is installed, then
Let's take a look the configuration for

Tensorflow & OpenCV Environment

Step 1.

Running O

Step 1. Create configuration
file .yml for CPU and GPU
version

conda-cpu.yml

```
1 name: yolov4-cpu
2
3 dependencies:
4   - python==3.7
5   - pip
6   - matplotlib
7   - opencv
8   - pip:
9     - opencv-python==4.1.1.26
10    - lxml
11    - tqdm
12    - tensorflow==2.3.0rc0
13    - absl-py
14    - easydict
15    - pillow
```



Name of the environment

17 lines (16 sloc) | 269 Bytes

```
1 name: yolov4-gpu
2
3 dependencies:
4   - python==3.7
5   - pip
6   - matplotlib
7   - opencv
8   - cudnn
9   - cudatoolkit==10.1.243
10  - pip:
11    - tensorflow-gpu==2.3.0rc0
12    - opencv-python==4.1.1.26
13    - lxml
14    - tqdm
15    - absl-py
16    - easydict
17    - pillow
```

Step 2. Create the environment

Step 2. Configure/create the environment in the folder you will run your openCV program by

Sconda env create -f conda-cpu.yml #for CPU
Sconda env create -f conda-gpu.yml #for GPU

file names to be used

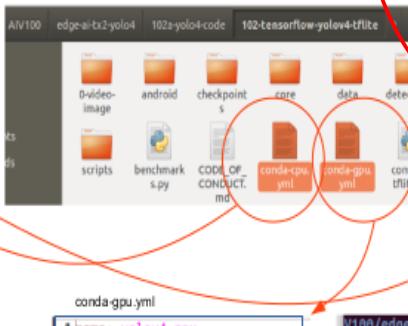
Step 3 Activate the environment

Step 4. Run Your Python Code (OpenCV or T.F.)

Running OpenCV with Conda Environment

Step 1. Create configuration file .yml for CPU and GPU version

```
conda-cpu.yml
1 name: yolov4-cpu
2
3 dependencies:
4   - python==3.7
5   - pip
6   - matplotlib
7   - opencv
8   - pip:
9     - opencv-python==4.1.1.26
10  - lxml
11  - tqdm
12  - tensorflow==2.3.0rc0
13  - absl-py
14  - easydict
15  - pillow
```



```
conda-cpu.yml
1 name: yolov4-cpu
2
3 dependencies:
4   - python==3.7
5   - pip
6   - matplotlib
7   - opencv
8   - pip:
9     - opencv-python==4.1.1.26
10  - bml
11  - lqdm
12  - tensorflow==2.3.0rc0
13  - absl-py
14  - easydict
15  - pillow
```

Name of the environment to be created

```
conda-gpu.yml
1 name: yolov4-gpu
2
3 dependencies:
4   - python==3.7
5   - pip
6   - matplotlib
7   - opencv
8   - cudnn
9   - cudatoolkit==10.1.243
10  - pip:
11    - tensorflow-gpu==2.3.0rc0
12    - opencv-python==4.1.1.26
13    - lxml
14    - tqdm
15    - absl-py
16    - pillow
```

Step 2. Configure/create the environment in the folder you will run your openCV program by

```
$ conda env create -f conda-cpu.yml      #for CPU
$ conda env create -f conda-gpu.yml      #for GPU
```

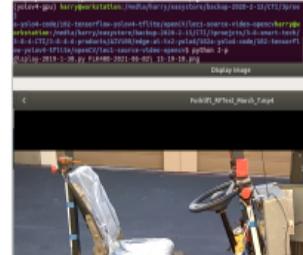
Step 3. Activate the environment with the name that you have created by

```
$ conda activate yolov4-cpu #for CPU
$ conda activate yolov4-gpu #for GPU
```

Name of the environment in the .yml file

Step 4. Now run your Python OpenCV program

\$ python myOpenCV.py image.png



Note: Install Anaconda → Create the environment for T.F. & OpenCV

Execute OpenCV → Activate the environment
Sample Code (from the class github)



a. Program Header

```
"""
pdisplay.py
Demo read and display image
"""
```

```
5 import sys
6 import cv2
7 import numpy as np
```

```
8
9 #main(sys.argv[1:])
10 window_name = 'Display Image'
```

```
11
12 imageName = sys.argv[1] #get file name from command line
13
14 src = cv2.imread(imageName, cv2.IMREAD_COLOR)
```

```
15
16 if src is None:
17     print ('Error opening image!')
18     print ('Usage: pdisplay.py image_name\n')
```

Name, Program Name, Version, Status(Deliver, Release), Date, Note

b. cv2 OpenCV

Caption of the window

CMPE258 Feb 8, 22

2nd Ref on the github)

9

2022S-103a-notation-neuro-loss-function-2022-2-8-1.pdf

```
22 while True: (3)  
23     cv2.imshow(window_name, src)  
24  
25     c = cv2.waitKey(500)  
26     if c == 27: #ESC  
27         break  
28  
29     ind += 1
```

Keyboard input to exit.

Note: These Python functions are registed (memorize them!)

- (1) import cv2 (2) cv2.imread()
- (3) cv2.imshow().

Homework (Due A week from today)
Feb 15

- 1° Installation of OpenCV.
- 2° Installation of Anaconda.
- 3° Use your Smartphone to take photo, and Save it for OpenCV Program
- 4° Write A Python Program (Ref. Code from the class github is ok)

to display:

- a. Your Name + SID (4 Digits)
- b. Your Smartphone picture.

- 5° Submission: One pdf file And Zipped. On CANVAS .

Feb 15

- Ref: 1° 2022S-103a-Notation
- 2° 2022S-103C

VIII. MINIMIZE THE LOSS FUNCTION

$$\frac{\partial L}{\partial w_{i,k}} = \frac{\partial}{\partial w_{i,k}} \frac{1}{2} \sum_{j=1}^P \sum_{i=1}^M (\hat{y}_i^j - y_i^j)^2 \quad (24)$$

Now, consider to Optimize Neural Network

Performance By minimizing the Loss function.

Mathematical Background

Derivative(s) → Partial Derivatives

a.



c. Gradient

↑

A special kind of Gradient: Descent

To Reduce Loss

"Steepest" ~

↑

Training A Neural Network

↑

Generalize the concept:
"Learning" (Supervised learning)

Derivatives.

Definition $\frac{df(x)}{dx} \triangleq \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$... (1)

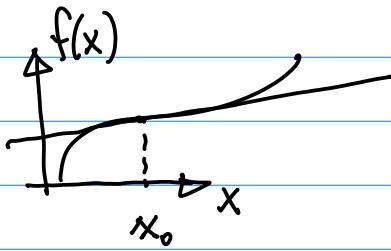
Note Eqn(1) is Based on "Forward Difference".

$f(x + \Delta x)$

forward of x



2022S-103a-notation-neuro-loss-function-2022-2-8.pdf



if $f'(x) = \frac{df}{dx} > 0$, then $f(x)$ at $x=x_0$ will increase

if $f'(x) < 0$, then $f(x)$ will decrease

if $f'(x) = 0$, then $f(x)$ is unchanged.

For a function $f(x_1, x_2)$, the partial

Derivative of $f(x_1, x_2)$

$$\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \dots (z)$$

$$\frac{\partial f(x_1, x_2)}{\partial x_1} \stackrel{def}{=} \lim_{\Delta x_1 \rightarrow 0} \frac{f(x_1 + \Delta x_1, x_2) - f(x_1, x_2)}{\Delta x_1} \dots (za)$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} \stackrel{def}{=} \lim_{\Delta x_2 \rightarrow 0} \frac{f(x_1, x_2 + \Delta x_2) - f(x_1, x_2)}{\Delta x_2} \dots (zb)$$

$$\frac{\partial f}{\partial x_1} > 0, f \uparrow \text{as } x_1 \uparrow$$

$$\frac{\partial f}{\partial x_2} > 0, f \uparrow \text{as } x_2 \uparrow$$

Define gradient of a function $f(x_1, x_2)$

(Note: Connection to Neural Network,

And a Single Neuron. inputs x_1, x_2, \dots, x_n

And Weights w_1, w_2, \dots, w_N

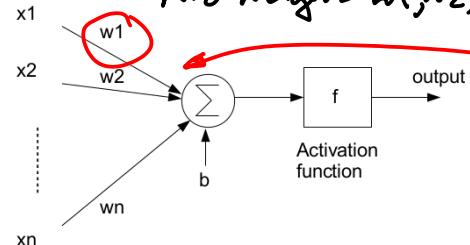


Fig 1.

Defined a vector

$$\begin{pmatrix} \frac{\partial f(x_1, x_2)}{\partial x_1} \\ \frac{\partial f(x_1, x_2)}{\partial x_2} \end{pmatrix} \text{ or } \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix}$$

We can use github Lecture ... (3)

Notes to expand Eqn (3) to N-Dimensional Case

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \dots (3b)$$

For A Single Neuron, we have gradient function as follows.

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial w_N} \end{bmatrix} \dots (3c)$$

Guideline for the Development of "Training" Technique :

Negative Gradient gives the steepest Descent of a Loss function f .

Note: Negative gradient gives the Best way to adjust w_i in a Neural Net

To make the Neural Net to Reach to Optimal Solution.
e.g. minimized loss function.

The Approach to Verify the Above Technique:

Generalized loss function, as f , or $f(\cdot)$, or $f(x_1, x_2, \dots, x_n)$, or

$f(\underline{w})$, or $f(w_1, w_2, \dots, w_n)$, or

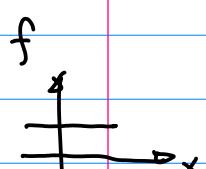
$f(w)$

Simple/Generalized Description to connect independent Variable x_1, x_2, \dots, x_n (or w_1, w_2, \dots, w_n) to the function f (loss function)

Taylor Expansion for $f(x)$

$$f(x) = f(x_0) + \frac{f'(x)}{1!}(x-x_0) + \frac{f''(x)}{2!}(x-x_0)^2 + \dots + \frac{f^{(k)}(x)}{k!}(x-x_0)^k + \dots \quad (4)$$

= Constant + 2nd (Linear function) +



$$\frac{f'(x)}{1!}(x-x_0)$$

Coefficient, Slop. $a(x-x_0)$
 $y = ax+b$ Linear function

$$\frac{f''(x_0)}{2!}(x-x_0)^2 \quad \text{Quadratic Term.}$$

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x-x_0) + T_n(x)$$

$$f(x) \approx f(x_0) + \frac{f'(x_0)}{1!}(x-x_0)$$

Linear function

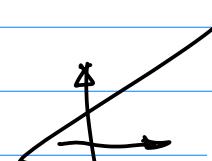
$$f(x) = f(x_0) + \frac{f'(x_0)}{1!} \Delta x \dots (5)$$

We expand function f (Loss) into Multi-dimensional Case, e.g. x_1, x_2, \dots (or w_1, w_2, \dots).

Choose work on 2D Case. e.g.

$$f(x_1, x_2) = f(x_{10}, x_{20}) + \frac{\partial f}{\partial x_1}(x-x_{10}) + \frac{\partial f}{\partial x_2}(x-x_{20}) + \dots \dots (6)$$

$$f(x_1, x_2) \approx f(x_{10}, x_{20}) + \frac{\partial f}{\partial x_1}(x-x_{10}) + \frac{\partial f}{\partial x_2}(x-x_{20}) \dots (7)$$



$$f(x_1, x_2) \approx f(a, b) + \frac{\partial f}{\partial x_1}(x_1 - a) + \frac{\partial f}{\partial x_2}(x_2 - b) \quad (6)$$

Question, if we change independent Variable x_1 and x_2 by the negative gradient of f , then the function f is decreased or increased?

Based on the Theory, we should have loss function f decreased in an optimized way, e.g. "Steepest Descent".

$$(x_1^{k+1}, x_2^{k+1}) = (x_1^k, x_2^k) + [-\eta(\nabla f)^T] \quad (5)$$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix}$$

$$\begin{cases} x_1^{k+1} = x_1^k + (-\eta) \frac{\partial f}{\partial x_1} & \dots (8a) \\ x_2^{k+1} = x_2^k + (-\eta) \frac{\partial f}{\partial x_2} & \dots (8b) \end{cases}$$

If we use Eqn.(8a), (8b) to update the independent Variables x_1, x_2 (or weights w_1, w_2), then you are sure to be able to find loss function f at this step $k+1$, is smaller than itself, e.g. f function at Step K.

$$f(x_1, x_2) - f(a, b) = (x_1 - a, x_2 - b) \nabla f \quad (10)$$

The Above equation, eqn(10), is Taylor Expansion

Note: from the Analysis below, we have:

$$f(x_1, x_2) - f(a, b) = (\Delta x_1, \Delta x_2) \nabla f = -\left(f_{x_1}^2 + f_{x_2}^2\right) \quad (11)$$

$$f_{x_1} = \frac{\partial f}{\partial x_1}$$

$$f_{x_1}^2 = \left(\frac{\partial f}{\partial x_1}\right)^2 \geq 0$$

$$\text{Similarly } f_{x_2} = \frac{\partial f}{\partial x_2}, \quad f_{x_2}^2 = \left(\frac{\partial f}{\partial x_2}\right)^2 \geq 0$$

$$\text{Therefore } -\left[\left(\frac{\partial f}{\partial x_1}\right)^2 + \left(\frac{\partial f}{\partial x_2}\right)^2\right] \leq 0$$

Hence

$$f(x_1, x_2) - f(a, b) \leq 0$$

$$f(x_1, x_2) \leq f(a, b)$$

$$x_1 - a = -\eta \frac{\partial f}{\partial x_1}$$

$$x_1^{k+1} - x_1^k = -\eta \frac{\partial f}{\partial x_1}$$

$$f(x_1, x_2) - f(a, b) = \frac{\partial f}{\partial x_1}(x_1 - a)$$

$$+ \frac{\partial f}{\partial x_2}(x_2 - b)$$

$$x_1^k = a, x_2^k = b \quad f(a, b)$$

$$f(x_1, x_2) \approx f(x_1^k, x_2^k) + \frac{\partial f}{\partial x_1}(x_1 - x_1^k) + \frac{\partial f}{\partial x_2}(x_2 - x_2^k) \quad \text{Eqn(7)}$$

PP.11

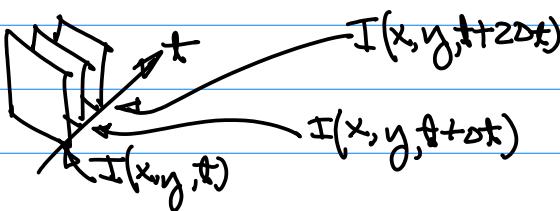
OpenCV

Note: One of the tools for Python

Programming for this class
is PyCharm.

Create A New Project \rightarrow Configuration
of the environment
& packages Needed

Example: OpenCV for Video.



Exercise Use your Smartphone to
Capture a video clip which
has the following feature:

1. White Background, Such as
Printer Paper or White Board,
then, write your last 4 Digits
on this white Background Paper/Board.

2. Be sure to use a marker to give
adequate contrast and width for
each mark. (1080P or 1080P Resolution)

3. Save the video (may have 20 to
Seconds video clip) for the future
use By Next Lecture.

Note: we will discuss Computer
Vision preprocessing techniques
in the next lecture.

Feb 22nd (Tue)

Today's Topics:

1. Theoretical Background on
Deep learning.
2. Preprocessing Techniques
for Handwritten Student ID
Recognition.

Single Neuron \rightarrow Loss function

Build multi-
Layer feed forward
NN \rightarrow Gradient Descent
Technique

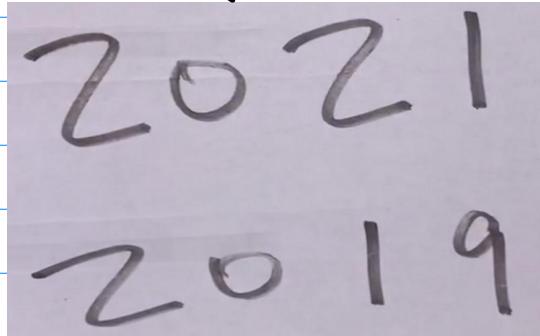
Back Prop. \rightarrow Deep Neural Network
 \rightarrow Convolutional NN
CNN]

Example: Pre-processing
Technique for handwritten Digits
Recognition.

2022S-107a-video-digits1.mp4

2022S-107b-video-digits2-outdoor.mp4

Capture A Single frame from the videos



- a.
- Objectives: To process video, identify ROI (Region of Interest), then
b. Remove Any Random Noise,
c. Enhance the quality of an image,
then feed it into Deep learning Neural Network.

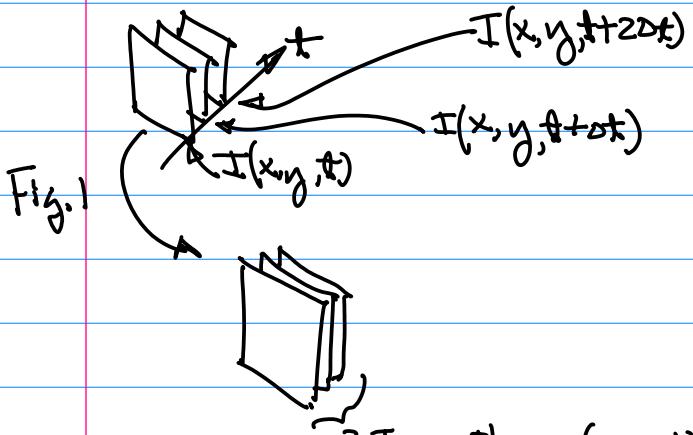
Example: Pre-processing Technique

Binary Image Processing.

Note: 1. A Color image, r, g, b

Red Green Blue

primitive colors, r, g, b : 8 bits per pixel.
24 Bits Color image



Tiled depth, Color Depth: 8 bits for Each Color plane

$$r \in [0, 255], g \in [0, 255], b \in [0, 255]$$

2. Gray Scale Image.

$I(x, y, t)$, drop the time t,
 $I(x, y)$ 8 bit gray scale image

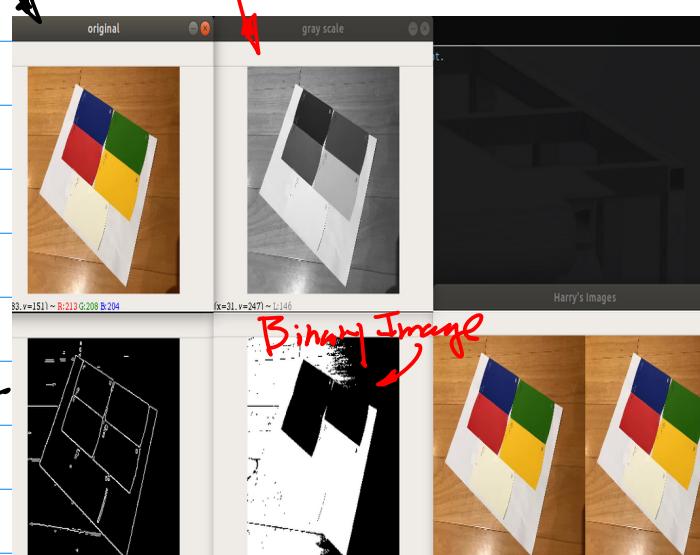
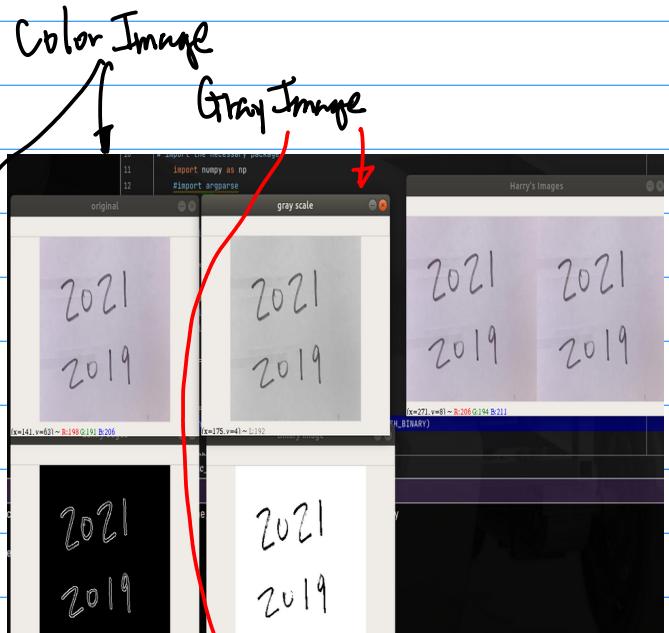
$$I(x, y) \in [0, 255],$$

$$I(x, y) = \frac{1}{3} [r(x, y) + g(x, y) + b(x, y)]$$

Provides a gray conversion.

Sometimes, Weighted Average

$$I(x, y) = \frac{1}{3} [d_1 r(x, y) + d_2 g(x, y) + d_3 b(x, y)] \quad \text{--- (1a)}$$



3. Binarize the gray scale

Image (Color Image Can
be binarized as well)

Binary image is denoted as

$$B(x, y) = \begin{cases} C_1 & \text{if } I(x, y) \geq \text{Threshold} \\ C_2 & \text{otherwise} \end{cases}$$

if $C_1 = 255$, $C_2 = 0$

Threshold is defined by user per
his/her processing need.

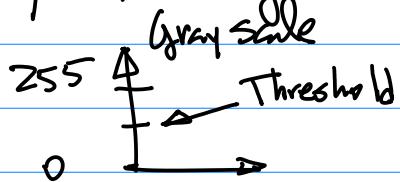


Fig.3.

Ref: For Binary Image Processing

[2022S-108-Intro-Binary.pdf](#)

Task: To Identify the Regions of
Interest (ROI)

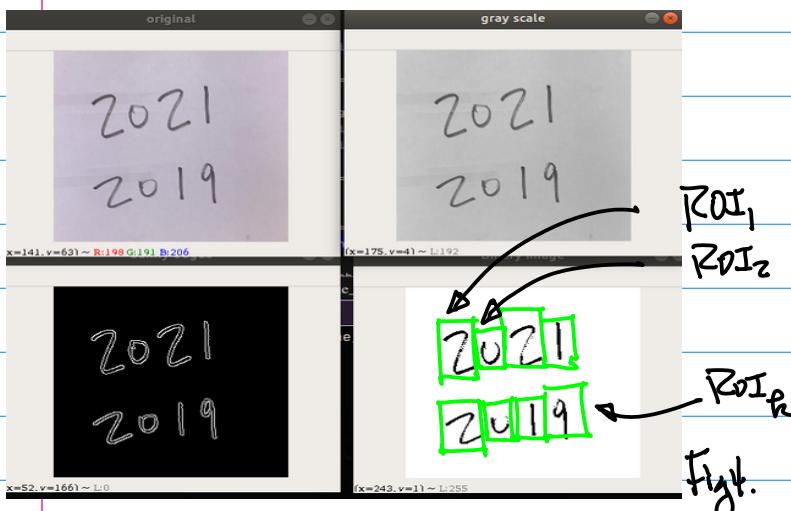


Fig.4.

Note: the Aspect Ratio for Each ROI

Can be different.

Define Aspect Ratio of ROI:

Given a ROI shown in Fig.5

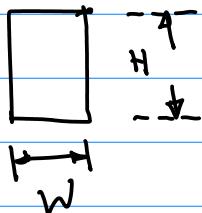


Fig.5

With Height H, Width W, the
Aspect Ratio α is defined as follows

$$\alpha = W/H \quad \dots \quad (3)$$

Suppose $W = 35$ (pixels), $H = 120$ (pixels)

$$\therefore \alpha = \frac{35}{120}$$

Note: Color image, gray scale image,
Binary image

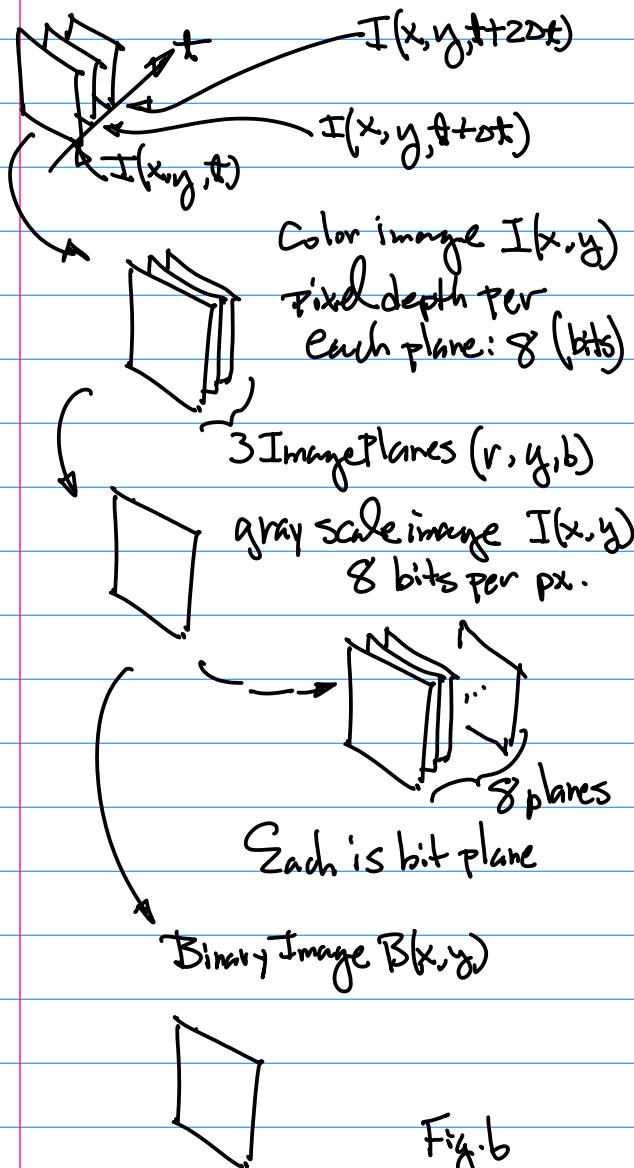


Fig.b

Question: Python code for Tensor Representation

of Each Image Types? e.g.

Color, gray scale, and Binary.

Example: Binarize a gray scale image

[2022S-108c-example-binary.pdf](#)

1) Given a digital image $f(x, y)$ find Binary Image Based on threshold $T = 135$.

So

Based on

$T = 135$

and

	x	y	$f(x, y)$
70	70	80	159
75	85	140	121
90	210	101	90
211	209	115	85

$$B(x, y) = \begin{cases} 255 & \text{if } f(x, y) \geq T \\ 0 & \text{otherwise} \end{cases}$$

(Origin of the Coordinate System)
(0,0)

Fig.1.

	K_2	y
K_1	0 0 0 255	0
x	0 0 255 0	0 255 0 0
	255 255 0 0	

However, for hard calculation, we use (1,1) instead.

K_1 : Index for Each Row,

K_2 : Index for Each Col.

Sample Code in OpenCV Python

[2022S-107d-display-binary.py](#)

```

20     image = cv2.imread(img, cv2.IMREAD_COLOR)
26
28     image = cv2.resize(image, (256, 256))
28     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

```

Note: 1. Binarization function, function's arguments

`ret, thresh1 = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)`

Input Image, threshold, intensity Value, Binarization Keyword

Example: Algorithms for Binary Image Processing.

2022S-108-Intro-Binary.pdf

Note 1. Binary or Gray Scale image,

$$I(x, y), \quad \begin{matrix} x = 0, 1, 2, \dots, N-1 \\ y = 0, 1, 2, \dots, M-1 \end{matrix}$$

No. of Row \uparrow for Col.

$M \times N$
Col. (px) per Row \uparrow Row.

2. Since Digital Image is Collection of Discrete Data, So Rewrite integration-Based formula By Summation.

From Eqn(1) of the handout, we have

$$\iint_{\Omega} B(x, y) dx dy$$

Ω : An Image, like the one below

70	70	80	159
75	85	140	124
90	210	101	90
211	209	115	85

$B(x, y)$: A Binary Image, like

we have

0	0	0	255
0	0	255	0
0	255	0	0
255	255	0	0

$\iint_{\Omega} B(x, y) dx dy$ $\dots (1)$
Summation w.r.t all x's

$\iint_{\Omega} B(x, y) dx dy$ $\dots (2)$
Summation w.r.t all y's

Pattern Recognition For Binary Images

The tool box for pattern recognition for binary images

- 1. Size
- 2. Moments

\bar{x}
 \bar{y}
 \bar{x}^2
 \bar{y}^2 etc.

- 3. Perimeter
- 4. Orientation
- 5. Compositions of the above
Perimeter and moments: vector
- 6. Invariant operators
size invariant
orientation invariant
illumination invariant

Harry Li, PhD, SJSU

Biologically inspired techniques

- Rule 1. Proximity
- Rule 2. Similarity
- Rule 3. Closure
- Rule 4. Good continuation
- Rule 5. Symmetry
- Rule 6. Simplicity

Note: 'Proximity' usage for clean up binary image and remove noise, as well as growing boundary points per 'good continuation' rule to form a better edge map.

Note: Similarity defines a interesting question, how to describe one object is similar, or somewhat similar to others, neural network and fuzzy logic may help.

Shape Descriptor

Invariant "Shape Descriptor"

In this class, we would like to develop

Techniques, Tools Described on the left Side of the PPT.

2022S-108b-BinaryImageFormula.pdf

Example:

$$\bar{x} = \frac{\iint_{\Omega} x \times B(x, y) dx dy}{\iint_{\Omega} B(x, y) dx dy} \dots (1)$$

Where, the Area of $B(x, y)$ is defined as

$$A = \iint_{\Omega} B(x, y) dx dy \dots (2)$$

Therefore, we have

$$\sum_{y=0}^{M-1} \sum_{x=0}^{N-1} B(x, y) \approx \iint B(x, y) dx dy \quad \dots (4)$$

$$\frac{1}{255} \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} B(x, y) = \text{Area}$$

Find

$$\sum_{y=0}^{M-1} \sum_{x=0}^{N-1} B(x, y)$$

for the given Binary Image $B(x, y)$

Below,

We have

		K_2	
		0	255
K_1	0	0	0
	255	0	0
255	0	0	0

Sol. From Eqn (4), we have

$$\sum_{y=0}^{M-1} \sum_{x=0}^{N-1} B(x, y)$$

Since image example is very small, we start index from 1, not 0.

Hence,

$$\sum_{y=1}^{M-1} \sum_{x=1}^{N-1} B(x, y) = \sum_{y=1}^4 \sum_{x=1}^4 B(x, y)$$

$$= \sum_{y=1}^4 \left(\sum_{x=1}^4 B(x, y) \right)$$

$$= \sum_{y=1}^4 (B(1, y) + B(2, y) + B(3, y) + B(4, y))$$

$$= B(1, 1) + B(2, 1) + B(3, 1) + B(4, 1) +$$

$$B(1, 2) + B(2, 2) + B(3, 2) + B(4, 2) +$$

$$B(1, 3) + B(2, 3) + B(3, 3) + B(4, 3) +$$

$$B(1, 4) + B(2, 4) + B(3, 4) + B(4, 4)$$

$$= (0+0+0+255)$$

$$0+0+255+0$$

$$0+255+0+0$$

$$255+255+0+0)$$

$$= 5 \times 255$$

After multiplying $\frac{1}{255}$, we have

$$\sum_{y} \sum_{x} B(x, y) = 5$$

Exercise 1: Form 4-Person Team,

elect a Coordinator of the Team.

to prepare for Semester Long Project

Send me email by next week with

the team information.

Exercise 2: Use Existing Code Sample

to write a program to display your Video Clips by next week.

March 1st.

Today's Topics: Preprocessing

for Digits Recognition:

2. Neural Networks (Feed Forward).

Homework Assignment Due A week from Today: March 8.

[2022S-102b-homework2-building-blocks-gradient-2022-3...](#)

Summation, A cumulative Loss

$$L_{total} = \frac{1}{2} \sum_{j=1}^P (\tilde{y}^j - y^j)^2$$

Ground Truth

$$y = f\left(\sum_{i=1}^N w_i x_i = W \cdot X + b\right) = f(h(w_i, b))$$

$$N=3, x_1, x_2, x_3, w_1, w_2, w_3$$

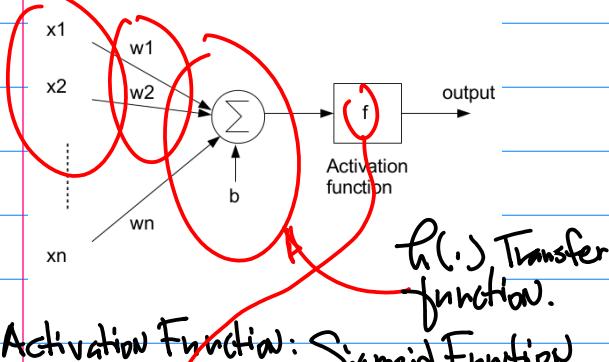
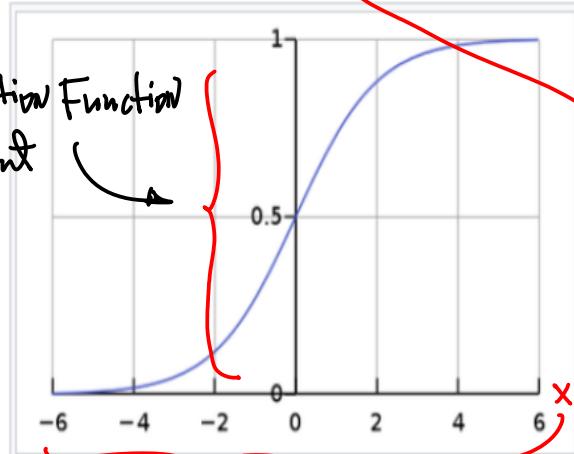


Fig 1.

Activation Function Output

Input to the Activation function, e.g. $f(w_i^j, b)$'s output

(4.3) Suppose there are 3 experiments (training) are performed, with the known ground truth as $(11.3, 0.2, 1)$. Use the equation in Figure 3, find the total loss. You can use abstract output symbol such as y^1, y^2 and y^3 in your result provided you have evaluated y^1 output based on the given parameters in this assignment.

$$\tilde{y}^{(1)} = 1.75 \text{ (for Example)}$$

$$\tilde{y}^{(1)} - y^{(1)} = 1.75 - y^{(1)}$$

$$\tilde{y}^{(2)} - y^{(2)}, \dots$$

$$\frac{\partial L}{\partial w_{i,k}} = \frac{\partial}{\partial w_{i,k}} \frac{1}{2} \sum_{j=1}^P \sum_{i=1}^M (\tilde{y}^j - y^j)^2$$

M=1, Then

ith Output

$$\frac{\partial}{\partial w_{i,k}} \frac{1}{2} \sum_{j=1}^P (\tilde{y}^j - y^j)^2 \Big|_{\tilde{y}^j = f(h(w_{i,k}, b))}$$

where

$$\tilde{y}^j = f(h(w_{i,k}, b)) = f\left(\sum_{i=1}^3 w_i x_i + b\right)$$

Therefore

$$\frac{\partial}{\partial w_{i,k}} \frac{1}{2} \sum_{j=1}^P (\tilde{y}^j - y^j)^2 =$$

$$\frac{1}{2} \sum_{j=1}^P (f\left(\sum_{i=1}^3 w_i x_i + b\right) - y^j)$$

$$\frac{1}{L} \sum_{i=1}^L (f\left(\sum_{j=1}^3 w_j x_j + b\right) - y_i)$$

w_1, w_2, w_3 ; x_1, x_2, x_3 ;
 b .

then construct feedforward NN.

Ground Truth

Example: Collecting data for training

Name	Weight (lb)	Height (in)	Gender
Alice	133	65	F
Bob	160	72	M
Charlie	152	70	M
Diana	120	60	F

The Loss function

The weights for Each
Neuron w_i to Replace
 x_i

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \dots \\ \frac{\partial f}{\partial x_i} \\ \dots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Fig.2

Signs	M_{ij}	M_{pq}
V1	133	65
V2	160	72
V3	152	70
V4	120	60

Note: Input vector (Training Data) (x_1, x_2)
2 Dimensional Vector.

7. Submit your work in one PDF file, then zip it. Use the following file
firstName_lastName_SID(last-4-digits)_cmpe258_CondaOpenCV.pdf.

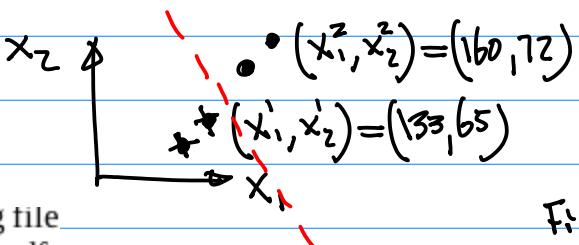
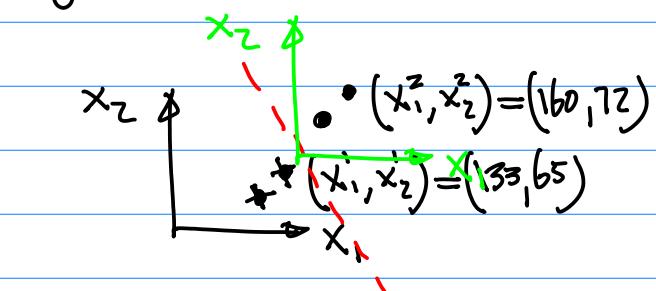
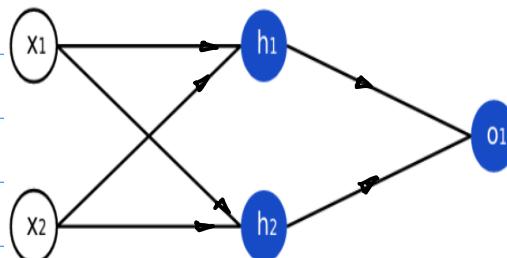


Fig.3

Basic Building Block
"B3"

Example: Build Feedforward Neural Network (Multi-Layer NN)

2022S-103c-NN-Intro-Python-v5.pdf



Define "Loss function", Error function

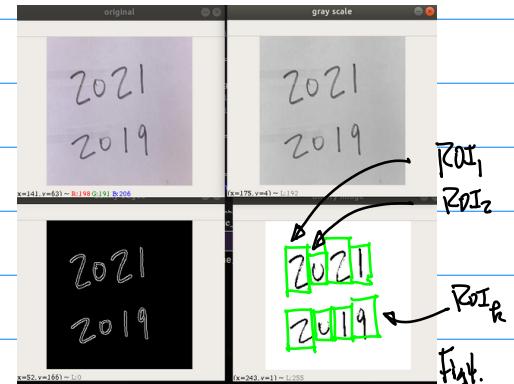
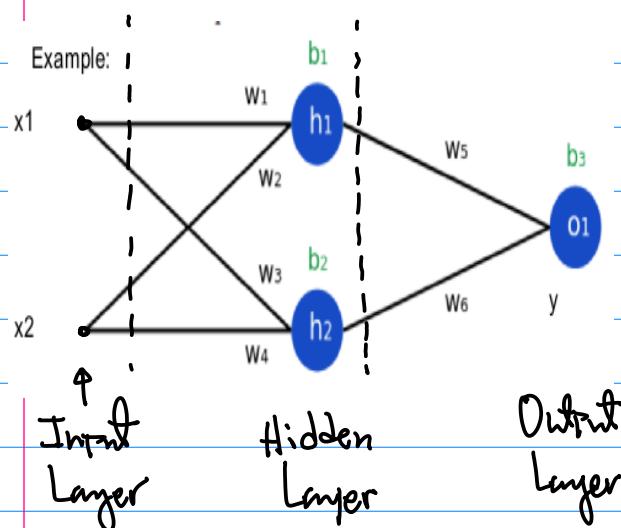
$$MSE = \frac{1}{n} \sum_{i=1}^n (y_{true} - y_{pred})^2$$

↑
Mean

Python Code for Basic Building Blocks,

Note: The enumeration for W_i

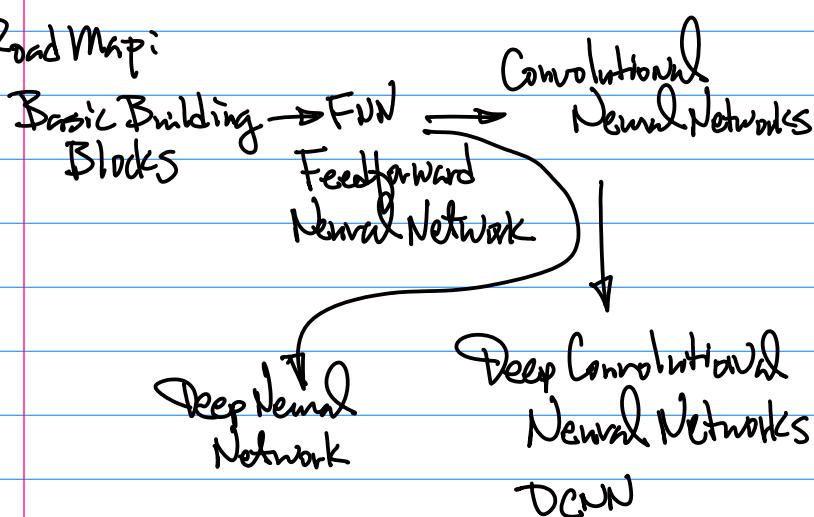
Each Handwritten Digit.



Requirements for this Python Code:

- 1° To be able to make connection Between the code and the Actual Neurons, Neural Network Architecture;
- 2° To modify the code to handle more neurons in the Hidden Layer if needed;
- 3° Run the code,

Road Map:

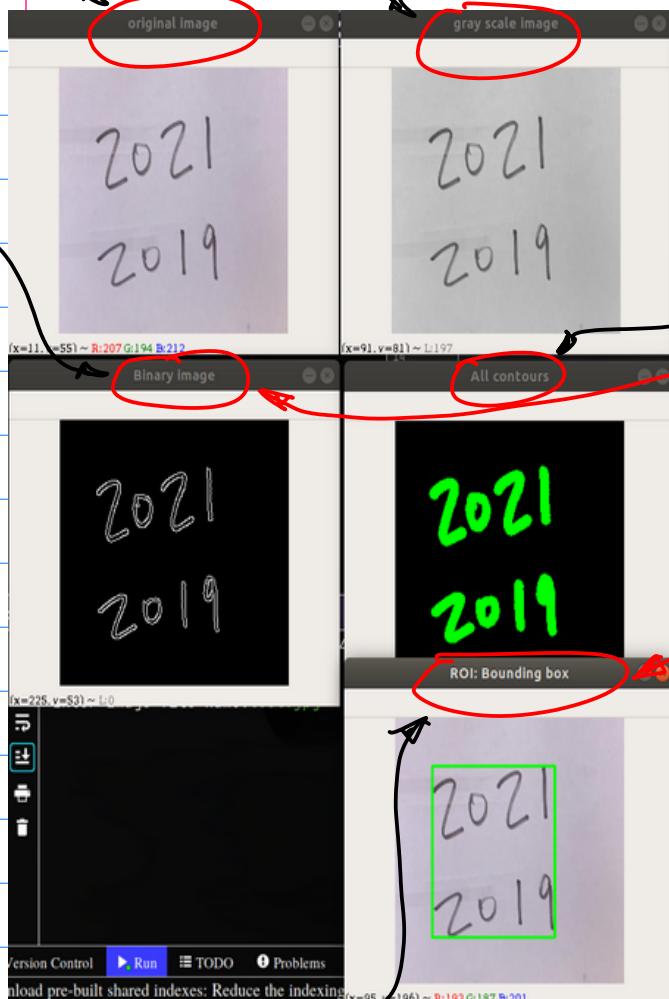


Example: Task to Develop Preprocessing Techniques(s) to define ROI. (to localize

Cmpe258 March 1st, 22

Note: a. Color Video \rightarrow Color Image, b.

Convert Color Image to gray
Scale



c. Edge map

Canny Edge Detector

Input Image, Resize it

```
18     img = cv2.imread(image, cv2.IMREAD_COLOR)
19     img = cv2.resize(img, (256, 256))
20     cv2.imshow('original image', img)
```

Convert the image to gray scale image.

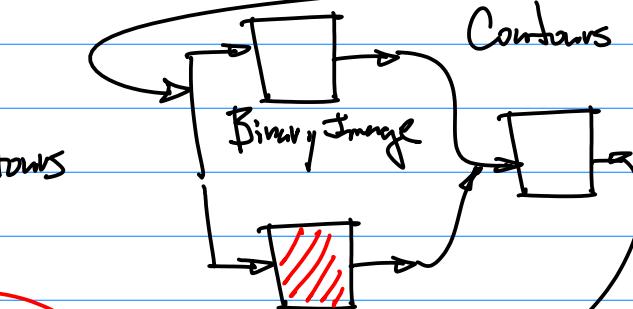
```
22     imggray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
23     cv2.imshow('gray scale image', imggray)
```

Note: Memorize the Code !

Color Video Image Gray Image



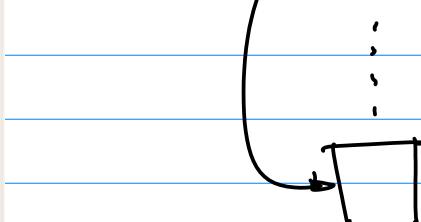
d. Contours



Edge map

Rectangle(Bounding Box)

Creation



Continued, Edge Detection, Canny Detector

26

thresh = cv2.Canny(imggray, 100, 200)

Produced Edge

map, which is a binary image.

$B(x, y)$.

Threshold1 Threshold2

Find Contours

```
29 contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```



Result

Source image Defines Contour Type

Draw color contour on color image

```
32 thresh1 = cv2.cvtColor(thresh, cv2.COLOR_GRAY2BGR)
```

```
33 cv2.drawContours(thresh1, contours, -1, (0, 255, 0), 5) # all contours
```

Compute(x, y) Top left corner of the Bounding Box, w: width, h: height

```
37 x, y, w, h = cv2.boundingRect(thresh)
```

```
38 cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
```

```
39 cv2.imshow('ROI: Bounding box', img)
```

