

CMPE258  
Spring 2023

Jan. 26 (Thu)

Organizational Meeting  
for Deep Learning Class.

Syllabus, "GreenSheet".

Note: 1° Syllabus is posted on the Class  
github. Also SJSU CANVAS

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

2023S-100-accessible-CMPE258-S23-v7-H...

**San José State University**  
**College of Engineering**  
**Computer Engineering Department**  
**CMPE258-Section 1 Deep Learning**  
**S2023**

**Course and Contact Information**

**Instructor:** Hua Harry Li, Ph.D.

**Office Location:** Engineering Building, Room 267A

**Telephone:** Mobile (650) 400-1116 Text message only

**Email:** [hua.li@sjsu.edu](mailto:hua.li@sjsu.edu)

**Office Hours:** MW 4:30 - 5:30 PM;

On-line with Zoom

Join Zoom Meeting

<https://us04web.zoom.us/j/98416076832>

pwd=U1A3aEk1TnV4bjNLQk5CQkw0dDk4UT09 Meeting ID: 984

160 7683 Passcode: 121092

**Class Days/Time:** Tuesdays, Thursdays 4:30 - 5:45 PM

**Classroom:** Zoom (link to be shared in the SJSU email)

Note: 4° Office Hours. On Zoom.

The office hours are good for the  
entire school semester, e.g.  
from the 1st day of the class till the  
last day of lecture.

Note: 3° Class on Zoom. Video  
Cam Activation for the  
class is required. Have  
your Video Cam Ready By Next Session

3° Attendance Requirement: Attend Lecture  
ON-Line is required.

CMPE258

Spring 2023

Note: 5. Class github. CANVAS is the only source for All Submissions, including Homeworks, Projects, Exam Papers etc. No

#### Faculty Web Page and MYSJSU Messaging (Optional)

Copies of the course materials such as the syllabus, major assignment handouts, etc. can be found on line at SJSU CANVAS, the same material is also provided at the following yahoo group, see URL below: e-mail

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

Submission is Accepted.

Office hours zoom link: Join Zoom Meeting [https://us04web.zoom.us/j/9841607683?](https://us04web.zoom.us/j/9841607683?pwd=U1A3aEk1TnV4bjNLQk5CQkw0dDk4UT09)

pwd=U1A3aEk1TnV4bjNLQk5CQkw0dDk4UT09 Meeting ID: 984 160 7683 Passcode: 121092

#### Course Description

Note: Pre-requisite CMPE 255 OR CMPE 257 is required.

Deep neural networks and their applications to various problems, e.g., speech recognition, image segmentation, detection and recognition of temporal and spatial patterns, and natural language processing. Covers underlying theory, the range of applications to which it has been applied, and learning from very large data sets.

Prerequisite: CMPE 255 or CMPE 257 or instructor consent. Computer Engineering and Software Engineering majors only.

#### Course Learning Outcomes (CLO)

Note: Book Listed below is a good reference source.

#### Required Texts/Readings

##### Textbook

- Deep Learning with Python, 1st Edition, by François Chollet, ISBN-13: 978-1617294433, ISBN-10: 9781617294433, <https://github.com/hualili/opencv/blob/master/IP120-AI-DL/2018F/2018F-6-DeepLearningCh02.pdf>
- Robot Vision by B.K. P. Horn, the MIT press, ISBN 0-262-08159-8, or 0-07-030349-5 (McGraw Hill).
- Reference textbook Learning OpenCV, Computer Vision with the OpenCV Library by Bradski and Kaebler, O'Reilly Publisher, ISBN 978-0-596-51613-0, 2011.

for Reference Only.

Practical / HandBook

#### Other Readings

1. OpenCV on line reference: <http://docs.opencv.org/index.html>
2. OpenGL on line reference (OpenGL programming guide): [ftp://ftp.sgi.com/opengl/contrib/kschwarz/OPEN\\_GL/REFERENCE/OGL\\_PG/ogIPG.pdf](ftp://ftp.sgi.com/opengl/contrib/kschwarz/OPEN_GL/REFERENCE/OGL_PG/ogIPG.pdf)
3. My lecture notes <https://github.com/hualili/opencv/tree/master/IP120-AI-DL/2018F> and <https://github.com/hualili/opencv/tree/master/deep-learning-2020S>

for the future Reference

References from the lecture Note

2022F-101-cmpe258-note-part2-2022-12-6...

Keyword "note"

#### Other equipment / material requirements

1. Python.
2. Or you may choose C++ as an option.
3. OpenCV.
4. Tensorflow Keras API.
5. Optional embedded board for assignment and projects: Nvidia Jetson NANO

Pycharm "Tool", Development/Debugging/Testing with Colab, Jupyter Note Book etc. All Line Tools are O.K. However, for the project Submission,

Stand-Alone Python Code for Deployment is Required. Also this

#### Course Requirements and Assignments

SJSU classes are designed such that in order to be successful, it is expected that student of forty-five hours for each unit of credit (normally three hours per unit per week), including participating in course activities, completing assignments, and so on. More details about

In the exams, the Deployable,  
Stand Alone Code is Required.

2 projects, Team/Semester-Long

Project

Final Presentation  
(Last week of the Semester).

### Grading Policy

Quiz, Homework, Projects	30%
Midterm Examination	30%
Final Examination	40%

CMPE258 Deep Learning, S2022.

On-Line.

0-59	F
60-69	D
70-79	C
80-89	B
90-100	A

### Classroom Protocol

Note: Homework Submission.  
One week from Today.

Will post the Homework  
on CANVAS.

### From University Policy F15-7:

#### 1.0 DEFINITIONS OF ACADEMIC DISHONESTY

##### 1.1 CHEATING

San José State University defines cheating as the act of obtaining credit, attempting to obtain credit, or assisting others to obtain credit for academic work through the use of any dishonest, deceptive, or fraudulent means. Cheating includes:

- 1.1.1. Copying, in part or in whole, from another's test or other evaluation instrument, including homework assignments, worksheets, lab reports, essays, summaries, and quizzes;
- 1.1.2. Submitting work previously graded in another course without prior approval by the

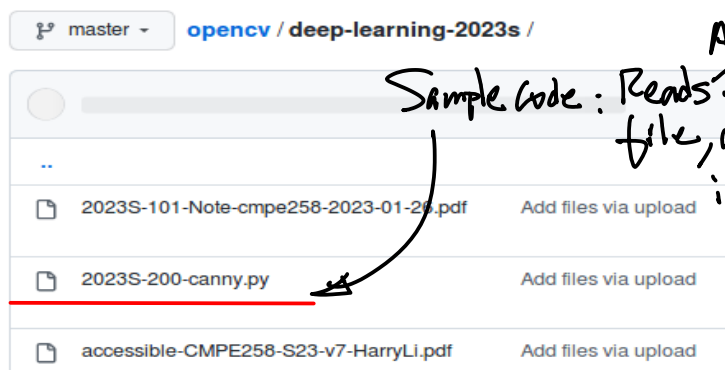
Jan 31 (Thu).

[opencv / deep-learning-2022s / 2022F-103-NN-Intro-Python-v5-2022-8-25.pdf](#)

1. Class Ref

2. Honesty Pledge Due  
this Thursday. Opt.

3. OpenCV & Anaconda  
Installation Due A week  
from Today. 1 Pt.

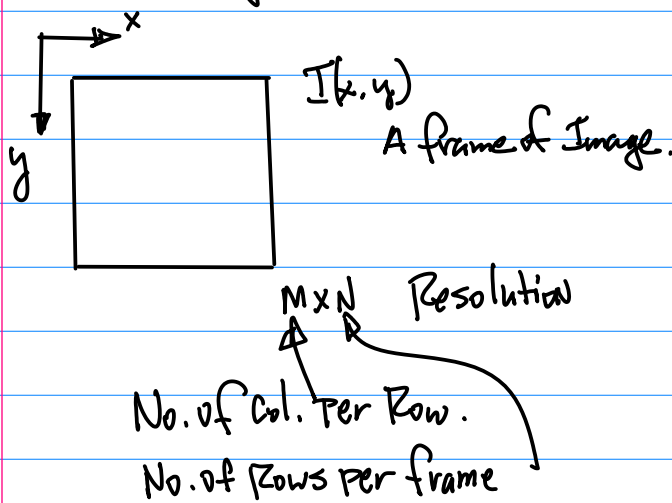
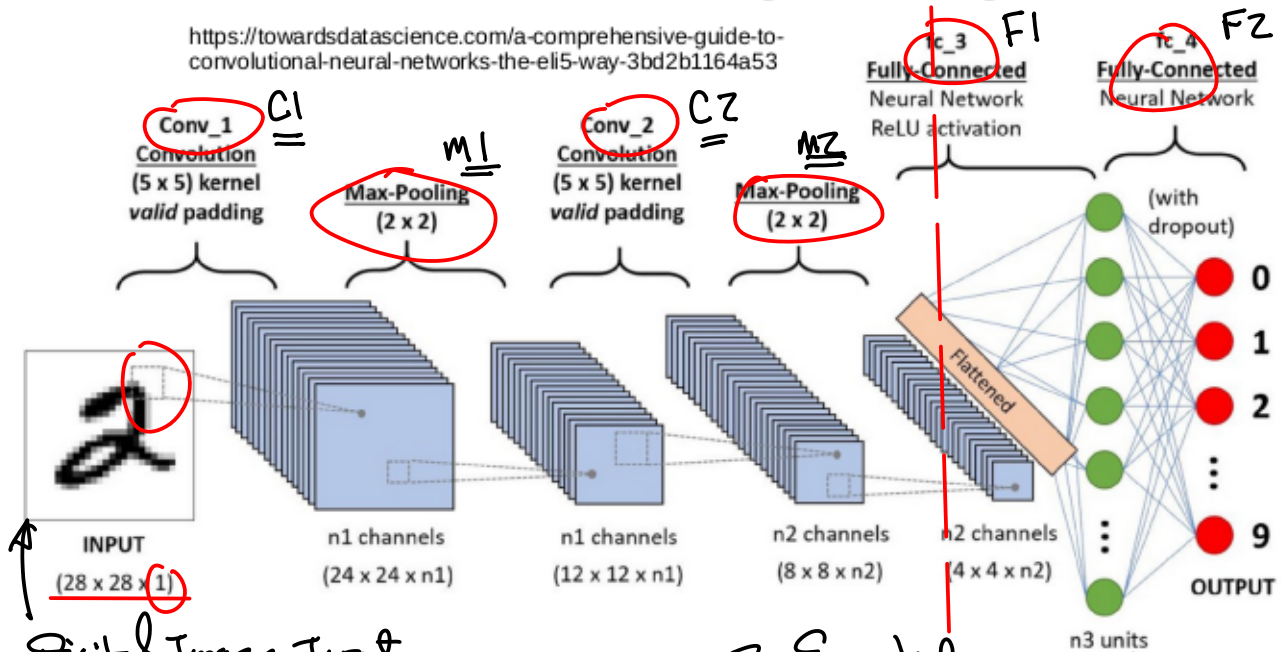


Sample Code: Reads Image  
file, and displays  
it.

Example:

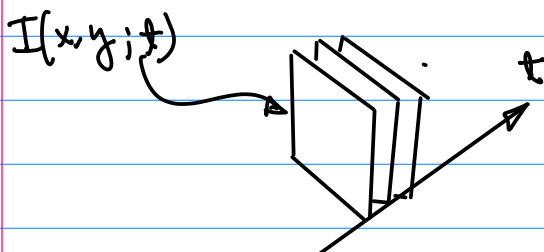
## Illustration of A CNN for Digits Recognition

<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>



$1920 \times 1280$   
M N

Pixel: picture element  
for a Video clip, OR Video Stream.

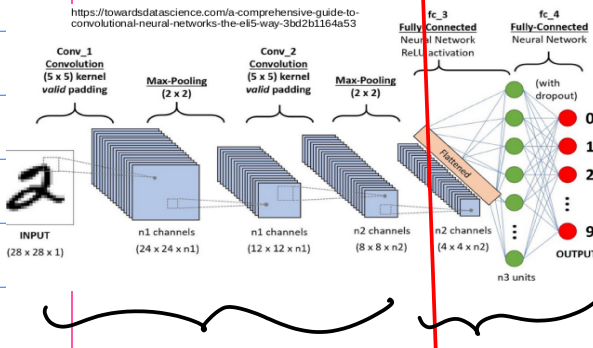


3. the "square" indicated on the input Image plane shows an Arbitrary Location of a Convolution Kernel.

4. "Conv-1"  $\rightarrow C \rightarrow C1$   
Simplified Notation. Adding An Index to Label the Convolution Block

"Max-Pooling"  $\rightarrow M \rightarrow M1$

Hence, the Architecture for the Above CNN (Convolutional Neural Network) is Can be described as  $C1M1C2M2$



Feature Extraction Decision Making.

Fully Connected (Feed Forward) Dense NN  $\rightarrow F$

$F1 \rightarrow$

And then the 2nd Block:  $F2$ .

Conclusion: The Architecture is defined as  $C1M1C2M2F1F2$

Feb 2nd (Thu).

Note: 1<sup>o</sup> Attendance, please use

Chat message to text me privately

Your First-Last Name, and 4-Digits STD.

2<sup>o</sup> Introduction & Team Formation.

Ref: from the class github.

1<sup>o</sup> [2022F-103b-NN-Intro-Python-v5-2022-8-25.pdf](#)  $\leftarrow$  PPT

2<sup>o</sup> Lecture Notes (Last Semester)

[2022F-101-cmpe258-note-2022-11-1.pdf](#)

3<sup>o</sup> White paper on Single Neuron Formulation

[2022S-103a-notation-neuro-loss-function-2022-2-8.pdf](#)

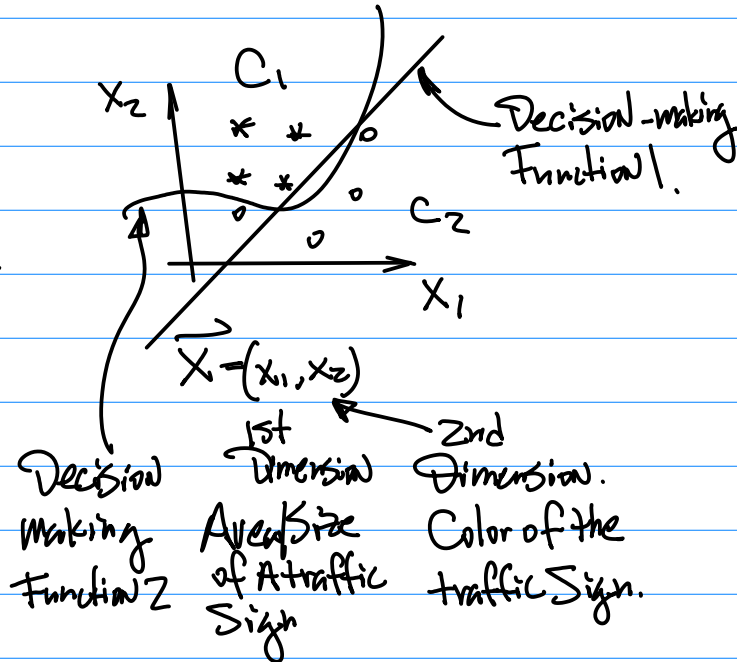
Example: Continuation of the Architecture

Question: How to Extract more features in general?

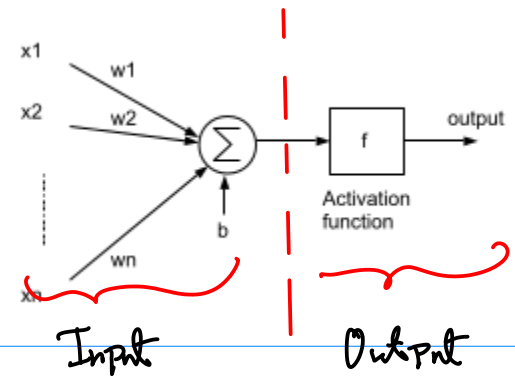
$C1M1$  Architecture v.s.  $C1M1C2M2$ ?

In general we can increase the Number of Convolutional Blocks. for Example:

$C1M1C2M2C3M3$ .



Consider A Single Neuron Below :



Note: 1<sup>o</sup> Input/Excitation

$$X = (x_1, x_2, \dots, x_n) \dots (1)$$

2<sup>o</sup> Weights, Links to Allow  $X$  to Connect to A Neuron.

$$W = (w_1, w_2, \dots, w_n) \dots (2)$$

$$w_i \in [0, 1]$$



### 3° Combining All the Inputs

$$x_1 w_1 + x_2 w_2 + \dots + x_n w_n \dots (3)$$

Notation for a neuron input  $x_i$ ,  $i = 1, 2, \dots, N$  is written as

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

and its vector form is

$$(x_1, x_2, \dots, x_N) \quad (2)$$

or simply denoted as  $X$ .

Now, introduce a superscript  $j$  for experiment  $j$ . The input is  $x_i^j$ , and  $i = 1, 2, \dots, N$  and  $j = 1, 2, \dots, P$ .

$$\{x_i^j | i = 1, 2, \dots, N; j = 1, 2, \dots, P\} \quad (3)$$

#### III. NOTATION FOR WEIGHTS

Notation for a weight  $w_i$ ,  $i = 1, 2, \dots, N$  is written as

$$\{w_i | i = 1, 2, \dots, N\} \quad (4)$$

and its vector form is

$$(w_1, w_2, \dots, w_N) \quad (5)$$

or simply denoted as  $W$ .

And

$$x_1 w_1 + x_2 w_2 + \dots + x_n w_n = \sum_{i=1}^N x_i w_i \dots (3-1)$$

And

$$\sum_{i=1}^N w_i x_i = w_1 x_1 + w_2 x_2 + \dots + w_N x_N \quad (9)$$

Or simply in a short hand vector form notation:

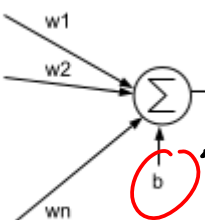
$$\sum_{i=1}^N w_i x_i = W \cdot X.$$

#### V. A TRANSFER FUNCTION

A transfer function is defined as

$$h = \sum_{i=1}^N w_i x_i = W \cdot X + b \quad (11)$$

$x_1$   
 $x_2$   
 $\vdots$   
 $x_n$



Adding A "bias" (Threshold) to Regulate/Control the Input, then the Response of the Neuron.

$$\sum_{i=1}^N x_i w_i \text{ v.s. } \sum_{i=1}^N x_i w_i + b \quad \text{offset}$$

#### 4. Define A Transfer function

$$h = \sum_{i=1}^N x_i w_i + b \text{ OR } \dots (4)$$

$$= \sum_{i=1}^{N+1} x_i w_i, \text{ where } x_{N+1} = b.$$

$$\text{And } w_{N+1} = 1 \dots (4-1)$$

$$h \rightarrow h(x_i, w_i) \rightarrow h(X, W; b)$$

#### 5. Activation Function.

Denoted as "f"

$$y = f \dots (5)$$

Output f

$$f(x_i, w_i) \dots (5-1)$$

$$f(h(x_i, w_i)) \text{ OR } \dots (5-2)$$

$$f(h(x_i, w_i; b)) \dots (5-3)$$

Vector Dot-Product  
( $x_1, x_2$ ) · ( $w_1, w_2$ )  
(10)

$$= x_1 w_1 + x_2 w_2$$

$$y = f\left(\sum_{i=1}^N w_i x_i = W \cdot X + b\right). \quad (17)$$

where  $y$  is the output of the neuron, and the activation function can be rewritten as

$$y = f\left(\sum_{i=1}^N w_i x_i = W \cdot X + b\right) = f(h(w_i, b)). \quad (18)$$

Or simply written as

$$y = f(h(\cdot)) = f(h(w_i, b)). \quad (19)$$

#### 6. Output from a Neuron $y$

Output from a Neural Network from an Experiment.

Suppose it is from the experiment  $j$   
 $j=1, 2, \dots, M$   
 $\hat{y}_j \leftarrow$  Superscript.

Feb 7 (Tue).

Note: 1<sup>st</sup> github Ref.

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

2022F-103b-NN-Intro-Python-v5-2022-8-25.pdf

2022F-101-cmpe258-note-2022-11-1.pdf

White paper

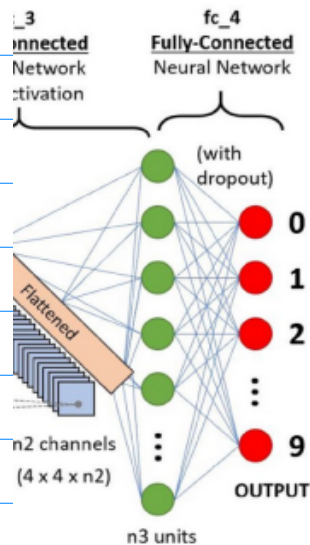
PPT.

Notes

Example:

Note: 1. Output  $y$

$$y_i = f_i(h(\cdot)) = f_i(h(w_i, b))$$



$y_i$

$i=0, 1, 2, \dots$

$\hat{y}_i$  output @  
Node  $i$

Fig. 1

Extend the Notation  $\hat{y}_i$  to  
allow us to describe an experiment  
 $j$ , for  $j=1, 2, \dots, N$

$$\hat{y}_i^j \mid_{j=1}^N = \hat{y}_i^1$$

Now, consider the performance of  
a Convolution Neural Network  
Comparison of the Network Output  
 $\hat{y}$  to the "Ground Truth"

$y$

$\hat{y}/y$  OR  $\hat{y}-y$  for  $A$   
= Single  
Output

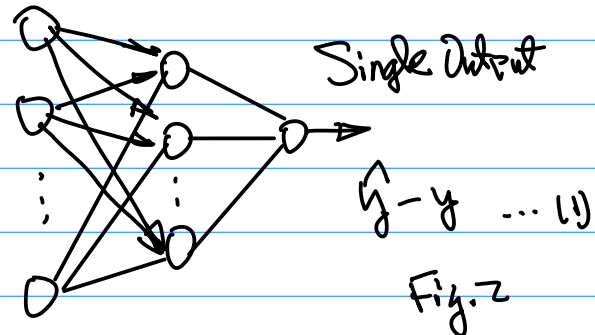


Fig. 2

For more than one Nodes, in Fig. 1.

$$\hat{y}_i - y_i \dots (1-b)$$

If for multiple experiments, then

$$\hat{y}_i^j - y_i^j$$

at Node  $i$ , Experiment/Training  $j$ , we  
are Comparing the Network Output to  
-h | e ground Truth.

Note: This Technique, e.g., Comparison  
of  $\hat{y}_i^j - y_i^j$  is a Supervised Learning.

Now, To measure the performance for all the Nodes, we have

$$\sum_{i=1}^N (\hat{y}_i^j - y_i^j) \dots (2)$$

Note  $\hat{y}_i^j - y_i^j \geq 0$  OR  $\hat{y}_i^j - y_i^j < 0$

From Eqn (2),

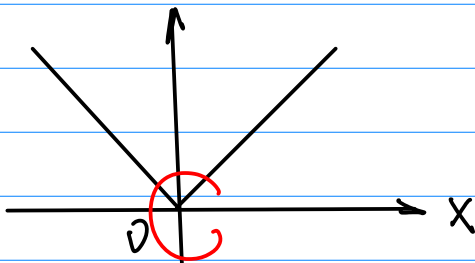
$$\sum_{i=1}^N (\hat{y}_i^j - y_i^j) = (\hat{y}_1^j - y_1^j) + (\hat{y}_2^j - y_2^j) + (\hat{y}_3^j - y_3^j) + \dots + (\hat{y}_N^j - y_N^j)$$

To Address this Problem, we can use Squared Value, Such as

$$\sum_{i=1}^N (\hat{y}_i^j - y_i^j)^2 \dots (3)$$

OR, Absolute Value. Consider the "behavior" of Absolute function,

$$f = |x| \dots (4)$$



Now, to Evaluate the performance for All Experiments,

$$\sum_{j=1}^M \sum_{i=1}^N (\hat{y}_i^j - y_i^j)^2 \dots (4)$$

Eqn (4) define A Loss function.

$$L = \sum_{j=1}^M \sum_{i=1}^N (\hat{y}_i^j - y_i^j)^2$$

Now, from the white paper, we have

$$L_{total} = \frac{1}{2} \sum_{j=1}^P (\hat{y}^j - y^j)^2 \dots (23)$$

for a single Output Node.

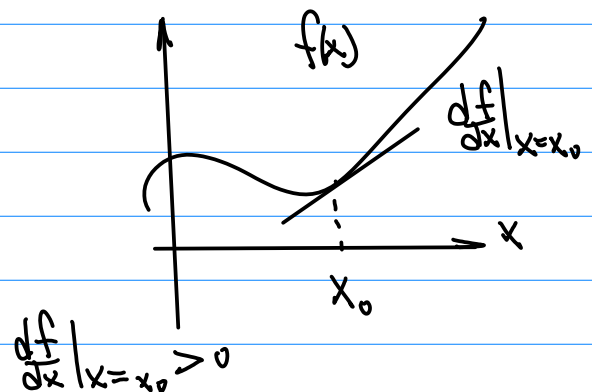
Coefficient " $\frac{1}{2}$ " is constant to allow a better/Simpler manipulations in Gradient Descent Analysis.

Consider the Improvement of A Neural Network Performance. See Eqn (24) from the white Paper.

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

Background:

Derivative of a function  $y = f(x)$ .







Feb 14 (Tue).

Homework: Installation of T.F.

Version 2.0 or higher. For the future projects & Homework, Source Code Submission is required with a Stand-Alone Python Code.

Submission of the Installation on CANVAS, in A week, Feb. 21 (Tue).

1° Screen Capture of the Installation.

2° Run the NN Python Code. with minor modification of Stopping Condition Based on the Loss function.

2022S-103c-#nn\_sample\_2022.py

Base Line Reference code on the github.

Today's Ref. Gradient Descent

2022S-105c-#20-2021S-4gradient-descent-v2-final-2021-2-8.pdf

Example: From the white paper, we have a function denoted as

$f(x_1, x_2, \dots, x_n)$  N-Dimensional

then, define partial derivative as follows

$$\frac{\partial f}{\partial x_i} = \lim_{\delta x_i \rightarrow 0} \frac{f(x_1, \dots, x_i + \delta x_i, \dots, x_n) - f(x_1, \dots, x_i, \dots, x_n)}{\delta x_i} \quad (1)$$

$\frac{\partial f}{\partial x_i}$  Derivative w.r.t  $x_i$  (with respect to)

where  $i=1, 2, \dots, N$ .

$f_{x_i}$

from Eqn (1),

$$\frac{\partial f}{\partial x_i} \triangleq \lim_{\delta x_i \rightarrow 0} \frac{\Delta f_{x_i}}{\delta x_i} = \lim_{\delta x_i \rightarrow 0}$$

$$\frac{f(x_1, x_2, \dots, x_i + \delta x_i, \dots, x_n) - f(x_1, x_2, \dots, x_n)}{\delta x_i}$$

Now, define a gradient

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

... (2)

N Dimensional Vector.

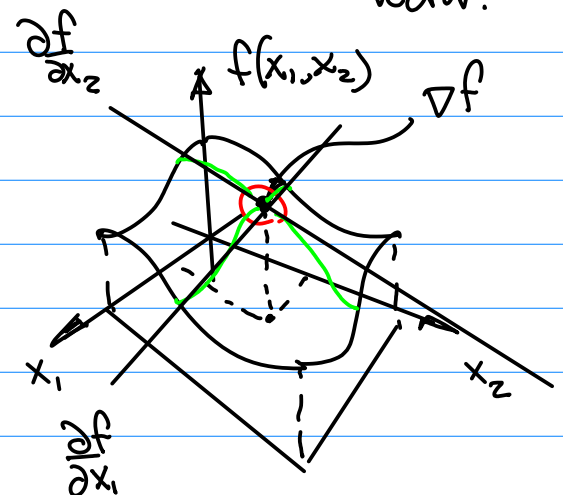


Fig.1

Note, Gradient at the illustrated point provides a better view of the function Behavior.

In Eqn(b), we have

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

Motivation: Use the gradient function to measure the behavior of the Loss function, And to update the Weights ( $\{x_i | i=1, \dots, N\}$ ) in such a way to reduce/minimize the loss function along its fastest descent direction. (Steepest)

The Technique to achieve this is Based on Taylor Expansion Series.

Consider  $f(x)$ , we can write it in a Standardized way.

$$f(x) = f(x_0) + \frac{df}{dx}(x-x_0) + \frac{1}{2!} \frac{d^2f}{dx^2}(x-x_0)^2 + \frac{1}{3!} \frac{d^3f}{dx^3}(x-x_0)^3 + \dots + R_n(x_0) \quad \dots (3)$$

Higher Order terms.

Basic Building Blocks

$f(x_0)$  : Constant,  $K$

$\frac{df}{dx} \Big|_{x=x_0} (x-x_0)$  :  $A(x-x_0) \rightarrow$

$y = ax + b$

Linear Function.

$\frac{d^2f}{dx^2} \Big|_{x=x_0} (x-x_0)^2$  :  $A(x-x_0)^2 \rightarrow$

$y = ax^2 + bx + c$

Remark 1: If function  $f(x)$  has derivatives upto Order  $K$ , then, it can always

be written as a sum of the Basic Building Block, e.g. a Constant term, a Linear term, a quadratic term, etc. Based on the Taylor Series.

Now, Find A Root of the Function.

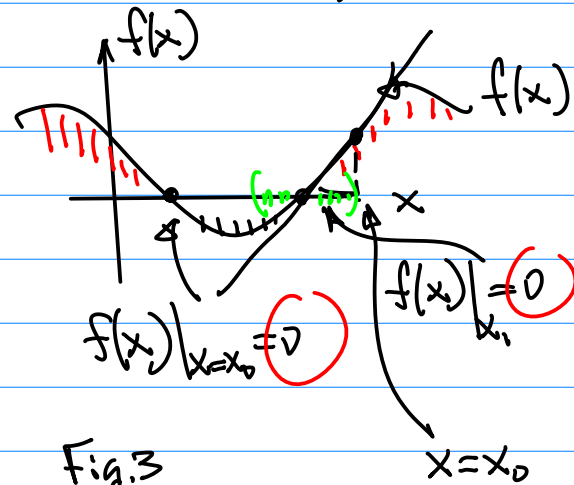


Fig.3

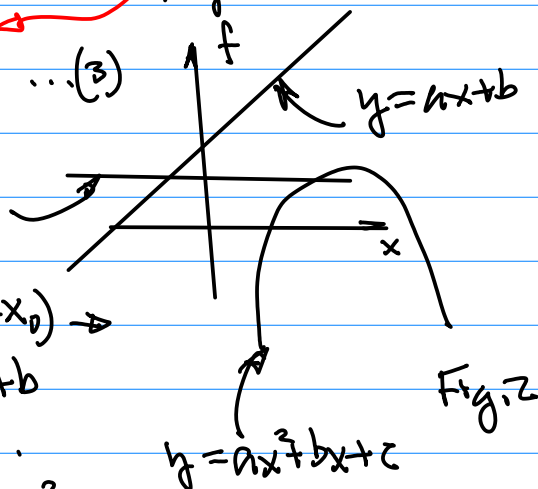


Fig.2

$$f(x) = f(x_0) + \frac{df}{dx}(x-x_0) + \left(\frac{1}{2!} \frac{d^2f}{dx^2}(x-x_0)^2 + \frac{1}{3!} \frac{d^3f}{dx^3}(x-x_0)^3 + \dots + R_n(x_0)\right)$$

Removal of  $R_n(x_0)$

Approximate

$$f(x) \approx f(x_0) + \frac{df}{dx}(x-x_0) + \left(\frac{1}{2!} \frac{d^2f}{dx^2}(x-x_0)^2 + \frac{1}{3!} \frac{d^3f}{dx^3}(x-x_0)^3 + \dots - \frac{1}{(n-1)!} \frac{d^{n-1}f}{dx^{n-1}}(x-x_0)^{n-1}\right)$$

$$f(x) \approx f(x_0) + \frac{df}{dx}(x-x_0)$$

$$f(x) - f(x_0) \approx \frac{df}{dx}(x-x_0)$$

$$\frac{f(x) - f(x_0)}{\frac{df}{dx}} = \underbrace{x - x_0}_{\text{Step } k, \text{ Step } k+1}$$

$$f(x, y) = f(x_0, y_0) + \frac{1}{1!} \frac{\partial f}{\partial x_1}(x-x_0) + \frac{1}{1!} \frac{\partial f}{\partial x_2}(y-y_0) + \dots \quad \dots (1)$$

just the Linear term,

$$f(x, y) \approx f(x_0, y_0) + \frac{1}{1!} \frac{\partial f}{\partial x_1}(x-x_0) + \frac{1}{1!} \frac{\partial f}{\partial x_2}(y-y_0)$$

$$= f(x_0, y_0) + \frac{\partial f}{\partial x_1}(x-x_0) + \frac{\partial f}{\partial x_2}(y-y_0)$$

Let  $x_0 = a, y_0 = b$ . Hence,

$$f(x, y) \approx f(a, b) + f_{x_1}(x-a) + f_{x_2}(y-b) \quad \dots (2)$$

from the definition of a gradient, we have

$$\nabla f(x, y) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} f_x \\ f_y \end{pmatrix} \quad \dots (3)$$

Feb 16 (Thu).

Example: Continuation. Now Consider N-Dimensional Case for Taylor Expansion.

$f(x_1, x_2, \dots, x_n)$  take  $n=2$ .  
Without loss of generality.

$f(x_1, x_2)$  Then,

$$f(x, y) - f(a, b) \approx (x-a, y-b) \cdot \begin{pmatrix} f_x \\ f_y \end{pmatrix} \quad \dots (3-b)$$

From the white paper, let's make

$$\Delta x_1 = x_1 - a = -f_{x_1}, \Delta x_2 = x_2 - b = -f_{x_2} \quad (11)$$

Therefore, substitute  $\Delta x, \Delta y$  ( $\Delta x_1, \Delta x_2$ ) into Eqn (3-b)

$$f(x, y) - f(a, b) \approx (-f_x, -f_y) \cdot \begin{pmatrix} f_x \\ f_y \end{pmatrix}$$

$f(x,y) - f(a,b) \approx -(f_x^2 + f_y^2)$  Discussion on Activation function  $f(\cdot)$

Since

$$f_x^2 + f_y^2 \geq 0$$

we have

$$-(f_x^2 + f_y^2) \leq 0 \quad \dots (4)$$

which leads to

$$f(x,y) - f(a,b) \leq 0$$

or

$$f(x,y) \leq f(a,b)$$

Note:

... (5)

$f(x,y) \approx f(a,b) + f_x(x-a) + f_y(y-b)$

$f(x,y)$  to be minimized

$(x,y)$  Next Step

$x-a, y-b$  Current Step  $\Delta x, \Delta y$

$f(a,b)$  Current Loss

To minimize  $f(x,y)$ , we will have  
Choose a better way to update the  
next step  $(x,y)$ , hence, the Negative  
gradient.

$$f(x_1, x_2) - f(a, b) = (\Delta x_1, \Delta x_2) \nabla f = -(f_{x_1}^2 + f_{x_2}^2) \quad (12)$$

Leads to the conclusion of minimization  
of the loss function By using Negative  
gradient.

Example:

Example 1. Given

$$y = f(x_1, x_2) = x_1^2 + x_1 x_2$$

Find its gradient as follows [1]:

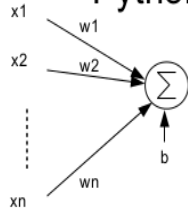
$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 + x_2 \\ x_1 \end{pmatrix}$$

$$\frac{\partial f}{\partial x_1} = \frac{\partial}{\partial x_1} (x_1^2 + x_1 x_2) = 2x_1 + x_2$$

$$\frac{\partial f}{\partial x_2} = \frac{\partial}{\partial x_2} (x_1^2 + x_1 x_2) = 0 + x_1$$

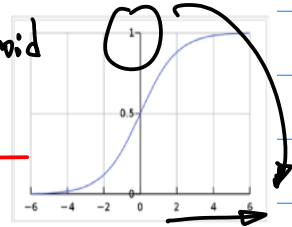
## Python Code For A Single Neuron

<https://victorzhou.com/blog/intro-to-neural-networks/>



① Sigmoid

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



Note: 1. define activation function

```
def sigmoid(x):
    # Our activation function: f(x) = 1 / (1 + e^(-x))
    return 1 / (1 + np.exp(-x))
```

2. define a single neuron

```
class Neuron:
    def __init__(self, weights, bias):
        self.weights = weights
        self.bias = bias

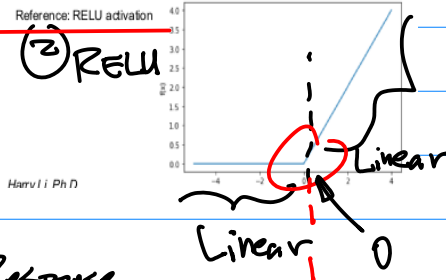
    def feedforward(self, inputs):
        # Weight inputs, add bias, then use the activation function
        total = np.dot(self.weights, inputs) + self.bias
        return sigmoid(total)
```

A sigmoid function is a mathematical function having a characteristic "S"-shaped curve

[https://en.wikipedia.org/wiki/Sigmoid\\_function](https://en.wikipedia.org/wiki/Sigmoid_function)

Reference: for further discussion on Sigmoid  
<https://deeplai.org/machine-learning-glossary-and-terms/sigmoid-function>

② RELU



Evaluate the Response  
of A Activation function.

Suppose we have  $h(w_i, x_i; b) = 0.5$   
Find Response of the Sigmoid, and the  
RELU

Sol Since the Sigmoid  $S(x) = \frac{1}{1 + e^x}$

We have  $x = h(w_i, x_i; b) = 0.5$  ... (6)

so

$$S(0.5) = \frac{1}{1 + e^{-x}} \bigg|_{x=0.5} = \frac{1}{1 + e^{-0.5}}$$

And for the RELU,

$$f(x) = \begin{cases} ax & \text{for } x \geq 0 \\ 0 & \text{o/w} \end{cases} \quad \dots (7)$$

$$f(x) = f(x) \Big|_{x=h(w_i, x_i; b)} = 0.5 \quad f(0.5)$$

$$= a \times 0.5 = 0.5a$$

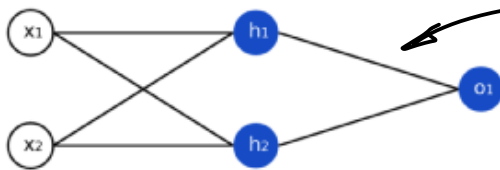
Example: Python Code for Simple  
Multi-Layer Feed forward NN.

Note: 1° Derivative, gradient Computation  
are Carried out by Finite  
Difference Approach, But in  
this Sample code, we are using  
a close form solution. see  
below

```
16 def deriv_sigmoid(x):
17     # Derivative of sigmoid: f'(x) = f(x) * (1 - f(x))
18     fx = sigmoid(x)
19     return fx * (1 - fx)
```

2° Write Python Code for NN like the following

Input Layer      Hidden Layer      Output Layer



```
def __init__(self):
    weights = np.array([0, 1])
    bias = 0

    # The Neuron class here is from the previous section
    self.h1 = Neuron(weights, bias)
    self.h2 = Neuron(weights, bias)
    self.o1 = Neuron(weights, bias)

    def feedforward(self, x):
        out_h1 = self.h1.feedforward(x)
        out_h2 = self.h2.feedforward(x)

        # The inputs for o1 are the outputs from h1 and h2
        out_o1 = self.o1.feedforward(np.array([out_h1, out_h2]))

        return out_o1

network = OurNeuralNetwork()
x = np.array([2, 3])
print(network.feedforward(x))
```

3° The code for MSE (mean  
Square Error) Loss function.

```
21 def mse_loss(y_true, y_pred):
22     # y_true and y_pred are numpy arrays of the same length.
23     return ((y_true - y_pred) ** 2).mean()
```

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

$$\begin{aligned} & y_{true} - y_{pred} \\ & \downarrow \\ & (y_{true} - y_{pred})^2 \end{aligned}$$

$$\sum_{i=1}^n (y_{true} - y_{pred})^2 \longrightarrow \text{Exp} \left[ \sum_{i=1}^n (y_{true} - y_{pred})^2 \right]$$

The Mean Square Error Function as a  
very often it is also defined as objective  
minimize the loss function becomes ob  
as shown below step by step.



Homework: Due A week from Today.  
Feb. 23 (Thu).

CANVAS.

- 1<sup>o</sup> Download the Sample code from the github. Modify and Run the code

2022S-103c-#nn\_sample\_2022.py

Check the convergence under different Learning Rate.

- 2<sup>o</sup> 2022S-102b-homework2-building-blocks-gradient-2022-3-1.pdf

CMPE258  
A Single Neuron Basic Building Blocks and Gradient Descent Function  
Homework  
HL

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

Figure 1.

1. Given the equation in Figure 1, design by drawing a single neural, for  $N=3$ , and  $w_1=0.3$ ,  $w_2=0.9$ ,  $w_3=0.83$ , suppose the bias  $b = 0.1$ .
2. Based on the equation in Figure 1, explain what is the function  $h(\cdot)$ , based on the parameters in Question 1 (above), for  $x_1=0.1$ ,  $x_2=14$ ,  $x_3=-7.5$ , find  $h$ ?
3. Suppose we choose the following function for activation function  $f$ , find the output of the neuron based on the equation in Figure 1, with the parameters in Question 1 and 2.

- 3<sup>o</sup> Update your OpenCV Code to Display Video from a Live Web CAM and from a file.

Feb 21st (Tue).

Note: 1<sup>o</sup> Team Formation update;

2<sup>o</sup> CANVAS Homework.

To Be continued: OpenCV Video Clip.

Today's Topic: Deep Convolutional Neural Network.

Example: Consider A Design of Hand Written Digits Recognition System.

Ref: 1<sup>o</sup> PP.3. Architecture of CNN

2022F-103b-NN-Intro-Python-v5-2022-8-25.pdf

2<sup>o</sup> Python Code

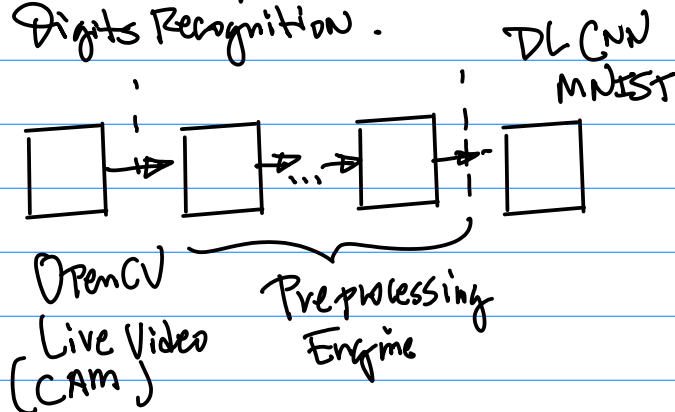
2022F-105a-#6mnist-numerals-ch02 (copy).py

Design A System with the following Requirements:

- 1<sup>o</sup> Use Laptop Web CAM to Capture Live Video ;  
Use A marker to write 7 Digits "ID" ON a Blank (White) Paper ;
- 2<sup>o</sup> Design Vision Preprocessing Engine (Algorithm), takes the video input then performs various different processing

tasks, then to provide input to the CNN (Deep Learning Engine).

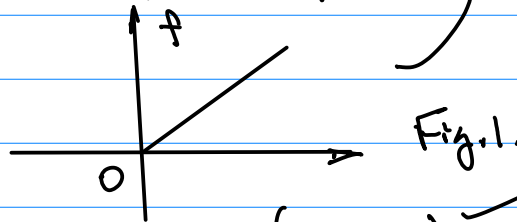
- 3<sup>o</sup> Compile & Build Pre-released Deep Learning Engine, e.g., MNIST to Perform Hand Written Digits Recognition.



Note: The Sample Code for MNIST Architecture: the Implementation of

```
31 network = models.Sequential()
32 network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28)))
33 network.add(layers.Dense(10, activation='softmax'))
```

Note: Activation Function: RELU.



2<sup>o</sup> input\_shape = (28 \* 28,)

Where  $(x, y)$  is the upper left position of the Bounding Box ;

We also define a Bounding Box as R.O.I. (Region of interests).

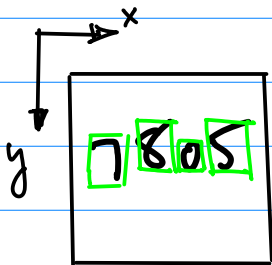
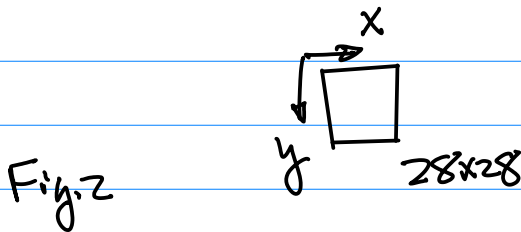
R.O.I. Denoted As

$$R_i(x, y, w, h, \bar{x}, \bar{y}) \dots (i)$$

W: Width

h: Height

$(\bar{x}, \bar{y})$ : Center of the R.O.I.



$I(x, y)$   
A frame of Image.

1080P Resolution; OR  
720P Resolution.  
M x N Resolution Resolution.  
1920 x 1280

No. of Col. Per Row. M N  
No of Rows per frame

Question 3: To Be Able to Extract the digit Inside R.O.I. Even if there are some Artifacts Around it. (Such as Background Color Variation, different Objects superimposed on top of the Digit, etc.).

Note: Define An Index to further Describe the Shape of the R.O.I.

$$\text{Aspect Ratio} = W/h \dots (3)$$

Question 1: Perform Segmentation to isolate each individual Digit

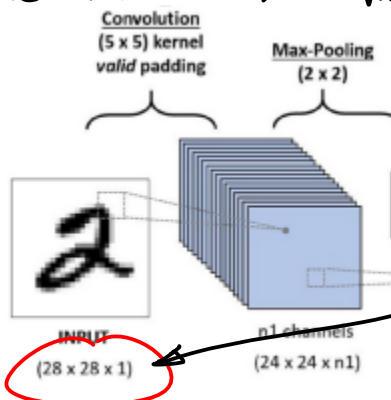
Question 2: place Bounding Boxes on to Each of Every Digits.

Denote Each Bound Box as

$$B_i(x, y) \text{ where } i = 1, 2, \dots, K$$

... (1)

Note: From Line 32 Input-shape:



$$A.R. \text{ Input Image} = w/h = 28/28 = 1$$

Requirements: The preprocessing Algorithm will have to:

1<sup>o</sup> Make its Output for Each R.O.I.

to meet 28x28x1 Resolution

Requirement

Channel, Gray Scale Image.

2<sup>o</sup> As the Output Image

Size is reduced to 28x28, we

have to make sure the R.O.I.

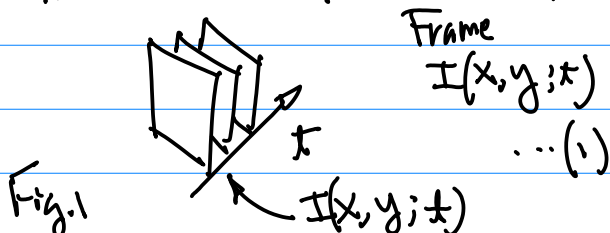
Aspect Ratio is preserved.

Feb 23 (Thu).

Example:

Question: 1<sup>o</sup> Perform Segmentation to isolate each individual Digit

Given A Video Clip  $\rightarrow$  A Image



At the moment  $t = t_0$ ,  
Write a Simplified

$$I(x, y) \dots (1-b)$$

$\downarrow$  Color Image

$$I(x, y) = (r(x, y), g(x, y), b(x, y))$$

red      green      blue

$r(x, y) \in [0, 255]$  (2<sup>8</sup> = 256, Color Plane where 8: No. of Bit per pixel per the primitive

Color), Same for  $g(x, y)$ , and  $b(x, y)$ .

$\downarrow$  Convert the Color image to a gray scale image  $I_g(x, y)$

$$I_g(x, y) = \alpha_1 r(x, y) + \alpha_2 g(x, y) + \alpha_3 b(x, y) \dots (3)$$

Treat each color plane equally.

$$\alpha_1 = \alpha_2 = \alpha_3 = 1/3$$

$$I_g(x, y) \text{ or simply } I(x, y)$$

Option 1 for Segmentation: Binarization (Thresholding Based)

$$B(x, y) = \begin{cases} 255 & I(x, y) \geq T \\ 0 & \text{o/w} \end{cases} \dots (4)$$

2022S-102b-homework3-2dConvolution-binary-202...

2022S-109-contour-intro-2022-2-28.pdf

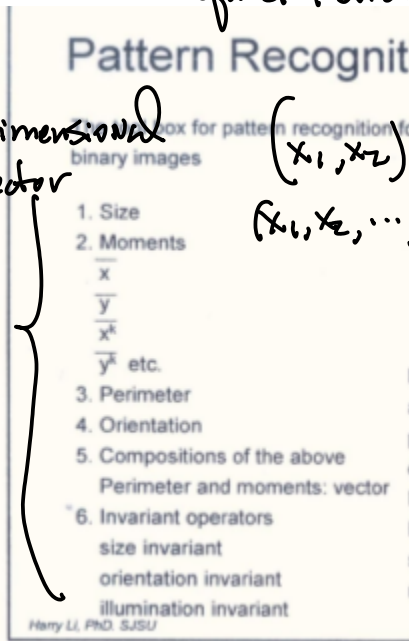
2022S-108-Intro-Binary.pdf

2022S-108b-BinaryImageFormula.pdf

2022S-108c-example-binary.pdf

Note: 1<sup>o</sup> Techniques / Algorithms  
quick Review

Multi-Dimensional  
Feature Vector



$(x_1, x_2)$  for 2D.  
 $(x_1, x_2, \dots, x_n)$  for  
N-Dimensional.

retrieved an contours without any hierarchical relationships.  
Harry Li, Ph.D. <http://opencvexamples.blogspot.com>

## Compute Contours Fe

[https://docs.opencv.org/3.1.0/dd/d49/tutorial\\_py\\_contour\\_features.html](https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html)

### 1. Moments

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread('star.jpg', 0)
5 ret, thresh = cv2.threshold(img, 127, 255, 0)
6 contours, hierarchy = cv2.findContours(thresh, 1, 2)
7
8 cnt = contours[0]
9 M = cv2.moments(cnt)
10 print M
```

### 5. Convex Hull

checks a contour  
corrects it  
hull = cv2.conv

### 6. Checking Contour

k = cv2.isConto

### 7.a. Straight B

1 x,y,w,h = cv2.b  
2 cv2.rectangle(i

### 7.b. Rotated R

1 rect = cv2.min  
2 box = cv2.box  
3 box = np.int0(i

### 2. Contour

area = cv2.contourArea(cnt)

### 3. Contour Perimeter

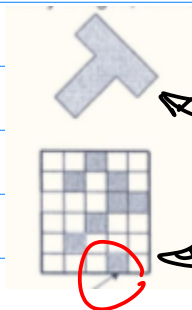


3<sup>o</sup> Consider the following Techniques

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

Note 2<sup>o</sup> Illustration of  $I(x,y)$  v.s.  
 $B(x,y)$



$I(x,y)$

$B(x,y)$

Fig. 2

Example: Given A gray scale image  $I(x,y)$  Below, And Threshold  $T=20$ ; find (1) Binary Image  $B(x,y)$ ; (2) Find its Size (Area)

Note: Hand Calculation Together with Math Formulations will be required.

Image Preprocessing  
Binary Images

Binary Images (Part 1)  
HL (Basics) 1.10

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

	x			
y	105	110	11	13
	105	120	8	7
	105	157	2	2
	255	255	10	20

Fig. 3

So! From Eqn (4), and the image  $I(x, y)$ , we have

$$B(x, y) = \begin{cases} 255 & I(x, y) \geq T = 20 \\ 0 & \text{o/w.} \end{cases}$$

105	110	11	13
105	120	8	7
105	157	2	2
255	255	10	20

Fig (4a)

 $I(x, y)$ 

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

Fig (4b)

 $B(x, y)$ 

$$A[B(x, y)] = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} B(x, y) \quad \dots (b)$$

Hence, the Area of  $B(x, y)$ :

$$A = \sum_{y=0}^3 \sum_{x=0}^3 B(x, y)$$

$$= \sum_{y=0}^3 (B(0, y) + B(1, y) + B(2, y) + B(3, y))$$

$$= B(0, 0) + B(1, 0) + B(2, 0) + B(3, 0) + \\ B(0, 1) + B(1, 1) + B(2, 1) + B(3, 1) + \\ B(0, 2) + B(1, 2) + B(2, 2) + B(3, 2) + \\ B(0, 3) + B(1, 3) + B(2, 3) + B(3, 3)$$

$$= 255 \times 9 \rightarrow \text{Normalize it}$$

$$A_{\text{Norm}} = A / 255 = 9$$

Start from the top left corner  $(0, 0)$ , "Scan" from the Left to Right one pixel at time; Then top-Down one row at time.

Now, Compute the Area.

$$A \triangleq \iint_{\Omega} B(x, y) dx dy \quad \dots (5)$$

$\Omega = B(x, y)$  Binary Image plane

For Digital Image, we have