

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/9841607683?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: 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 e-mail

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

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

<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=UIA3aEk1TnV4bjNLQk5CQkw0dDk4UT09)

pwd=UIA3aEk1TnV4bjNLQk5CQkw0dDk4UT09 Meeting ID: 984 160 7683 Passcode: 121092

#### Course Description

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-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 Kaehler, O'Reilly Publisher, ISBN 978-0-596-51613-0, 2011.

#### 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/oglPG.pdf](ftp://ftp.sgi.com/opengl/contrib/kschwarz/OPEN_GL/REFERENCE/OGL_PG/oglPG.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>

Practical / HandBook

for the future Reference

References from the lecture Note

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

Key word "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 Notebooks, Online Tools are OK. 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 students 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 created in another course without prior approval by the

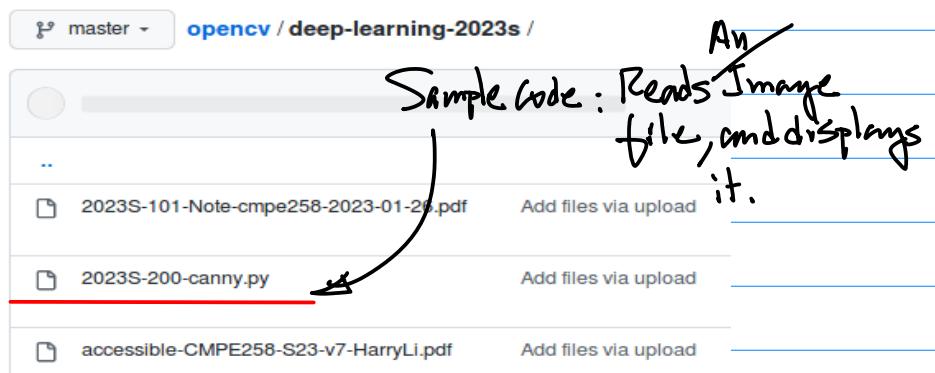
Jan 31 (Thu).

[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.



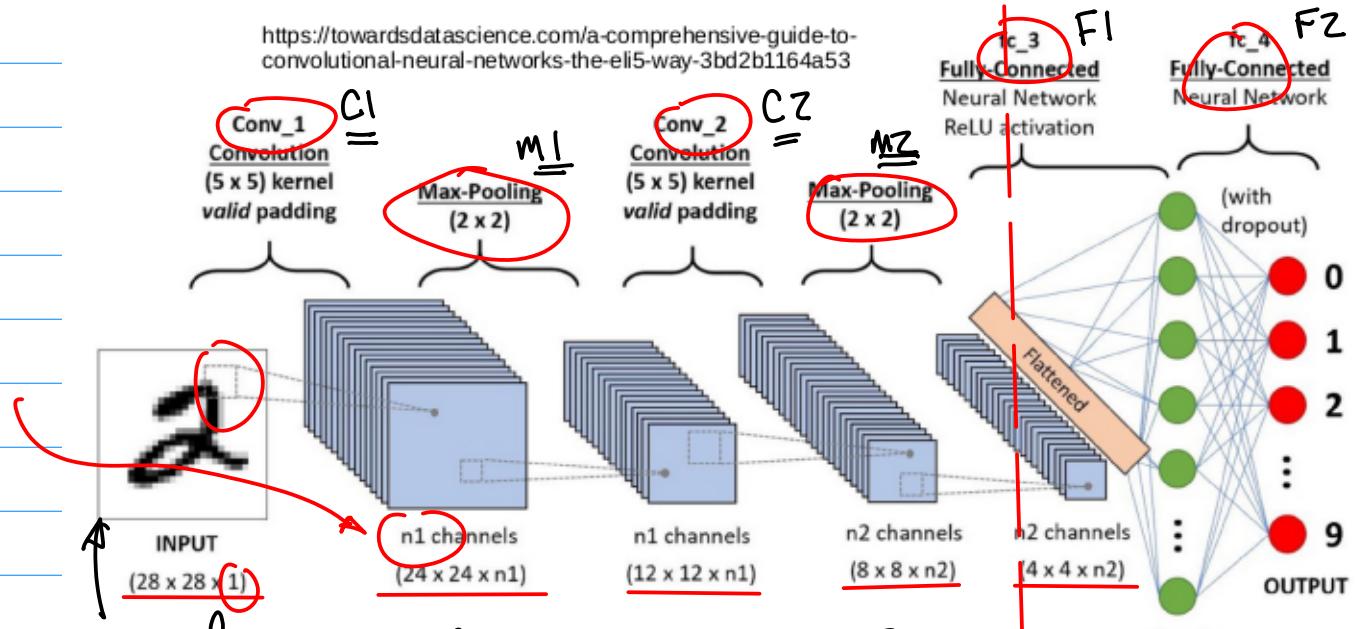
Example:

$$\begin{array}{|c|c|} \hline 2 & 127 \\ \hline 0 & 5 \\ \hline \end{array} \xrightarrow{\text{max-pooling}} \boxed{127} \quad \max\{2, 127, 0, 5\} = 127$$

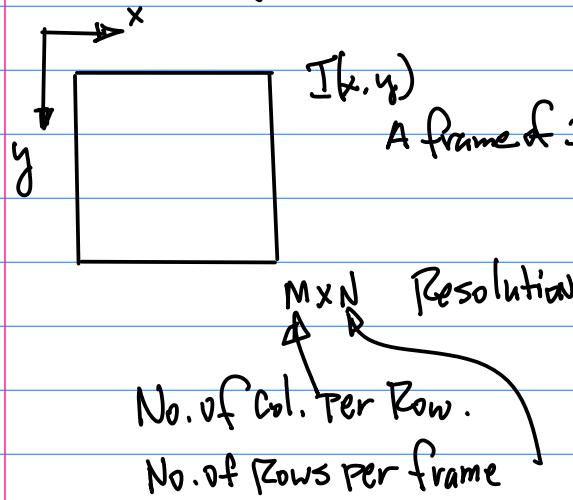
4)

## Illustration of A CNN for Digits Recognition

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



1. Digital Image Input.



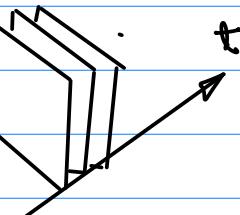
$1920 \times 1280$

$M \quad N$

Pixel: picture element

for a Video Clip, DR Video Stream.

$I(x, y, t)$



2. Example for  $28 \times 28 \times 1$ ,

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

8 bit Gray scale image

Brightest, White  
Darkest, Black

3. the "square" indicated on the input Image plane shows an arbitrary location of a convolution kernel.

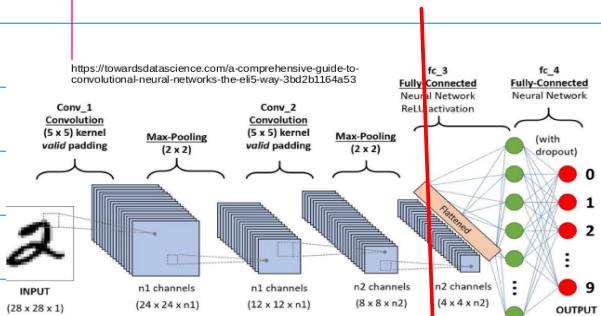
4.  $\text{Conv-1} \rightarrow C \rightarrow C_1$

Simplified Notation. Adding An Index to Label the Convolution Block

"Max-Pooling"  $\rightarrow M \rightarrow M_1$

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

$C_1 M_1 C_2 M_2$



Feature Extraction      Decision Making.

Fully Connected (Feed Forward) Dense NN  $\rightarrow$  F<sub>1</sub>

And then the 2nd Block : F<sub>2</sub>.

Conclusion: The Architecture is defined as C<sub>1</sub>M<sub>1</sub>C<sub>2</sub>M<sub>2</sub>F<sub>1</sub>F<sub>2</sub>

Feb 2nd (Thu).

Note: 1° Attendance, please use Chat message to text me privately

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

2° Introduction & Team Formation.

Ref: from the class github.

1° [2022F-103b-NN-Intro-Python-v5-2022-8-25.pdf](#)

2° [Lecture Notes \(Last Semester\)](#)

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

3° [Whitepaper on Single Neuron Formulation](#)

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

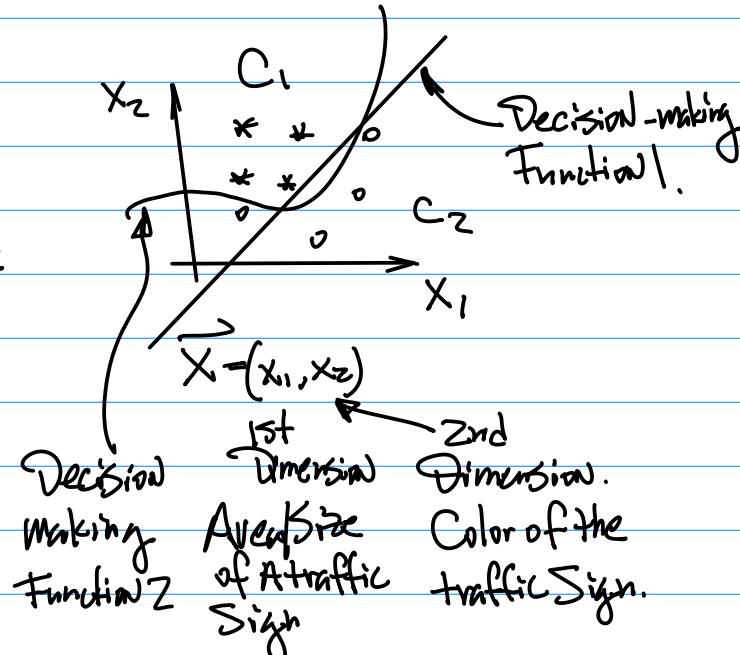
Example: Continuation of the Architecture

Question: How to Extract more features in general?

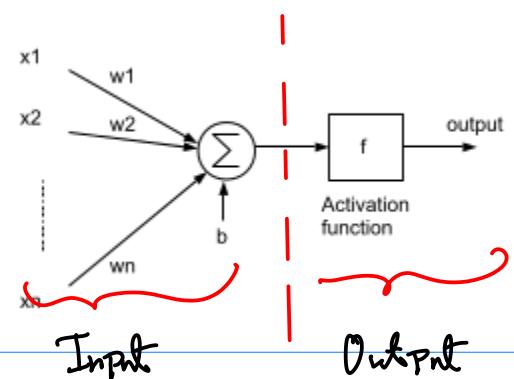
C<sub>1</sub>M<sub>1</sub> Architecture V.S. C<sub>1</sub>M<sub>1</sub>C<sub>2</sub>M<sub>2</sub>?

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

C<sub>1</sub>M<sub>1</sub>—C<sub>2</sub>M<sub>2</sub>—C<sub>3</sub>M<sub>3</sub>.



Consider A Single Neuron Below :



Note: 1° Input / Excitation

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

2° 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

$$x_1w_1 + x_2w_2 + \dots + x_Nw_N \dots (3)$$

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

$$x_1w_1 + x_2w_2 + \dots + x_Nw_N = \sum_{i=1}^N x_iw_i \dots (3-1)$$

And

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

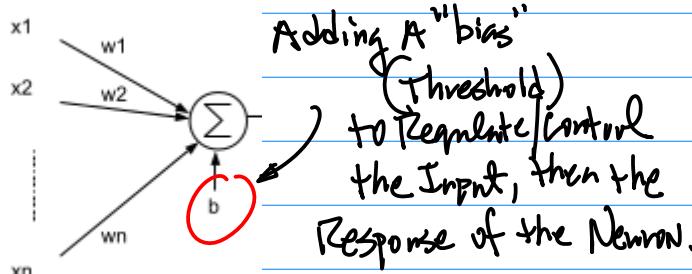
Or simply in a short hand vector form notation:

$$\sum_{i=1}^N w_i x_i = W \cdot X. \quad (10)$$

## V. A TRANSFER FUNCTION

A transfer function is defined as

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



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

## 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 ↓

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

Vector Dot Product

$$(x_1, x_2) \cdot (w_1, w_2) \quad (10)$$

$$= x_1w_1 + x_2w_2$$

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

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

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

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

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

Or simply written as

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

6. Output from a Neuron  $y$

Output from a Neuron if from an Experiment.

$$\hat{y}_j$$

Suppose it is from the experiment  $j$

$$j=1, 2, \dots, M$$

$$\hat{y}_j^i \leftarrow \text{Superscript.}$$

Feb 7 (Tue).

Note: 1<sup>o</sup> GitHub Ref.

White paper

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

[2022F-103b-NN-Intro-Python-v5-2022-8-25.pdf](#) PPT.

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

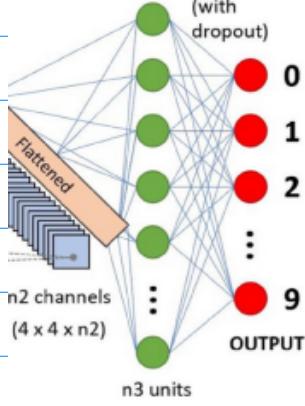
Example: Note 1. Output  $y_j$

$$y_j = f_j(h(\cdot)) = f_j(h(w_j, b))$$

3 connected Network activation

fc\_4 Fully-Connected Neural Network

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



$$\hat{y}_j^i \text{ output } @ \text{ Node } i$$

Fig. 1

Extend the Notation  $\hat{y}_j^i$  to

allow us to describe an experiment  $j$ , for  $j=1, 2, \dots, N$

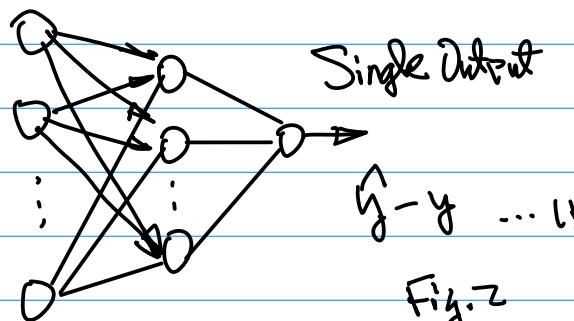
$$\hat{y}_j^i |_{j=1}^N = \hat{y}_i$$

Now, consider the performance of a Convolution Neural Network Comparison of the Network Output

$\hat{y}_j^i$  to the "Ground Truth"

$$y_j$$

$$\hat{y}_j^i / y_j \text{ or } \hat{y}_j^i - y_j \text{ for } A \\ = \text{Single Output}$$



For more than one Nodes, in Fig. 1.

$$\hat{y}_j^i - y_i \dots (l-b)$$

If for multiple experiments, then

$$\hat{y}_j^i - y_i$$

at Node  $i$ , Experiment / Training  $j$ , we are Comparing the Network Output to

- h i.e ground Truth.

Note: This Technique, e.g., Comparison if  $\hat{y}_j^i - y_i$  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),

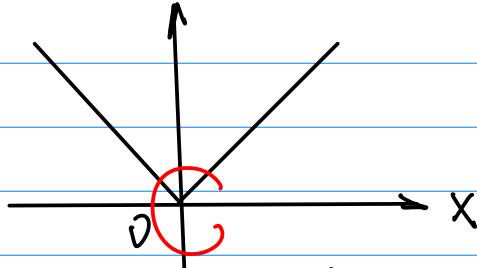
$$\begin{aligned} \sum_{i=1}^N (\hat{y}_i^j - y_i^j) &= (\hat{y}_1^j - y_1^j) + \\ &(\hat{y}_2^j - y_2^j) + (\hat{y}_3^j - y_3^j) + \dots + (\hat{y}_N^j - y_N^j) \end{aligned}$$

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) defines 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

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

for a Single Output Node.

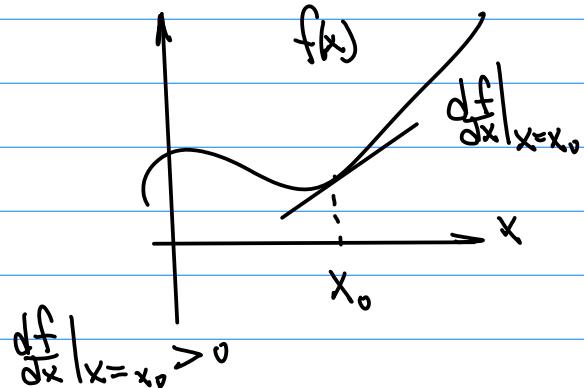
Coefficient  $\frac{1}{2}$  is constant to allow a better/Simpler Manipulations in Gradient Descent Analysis.

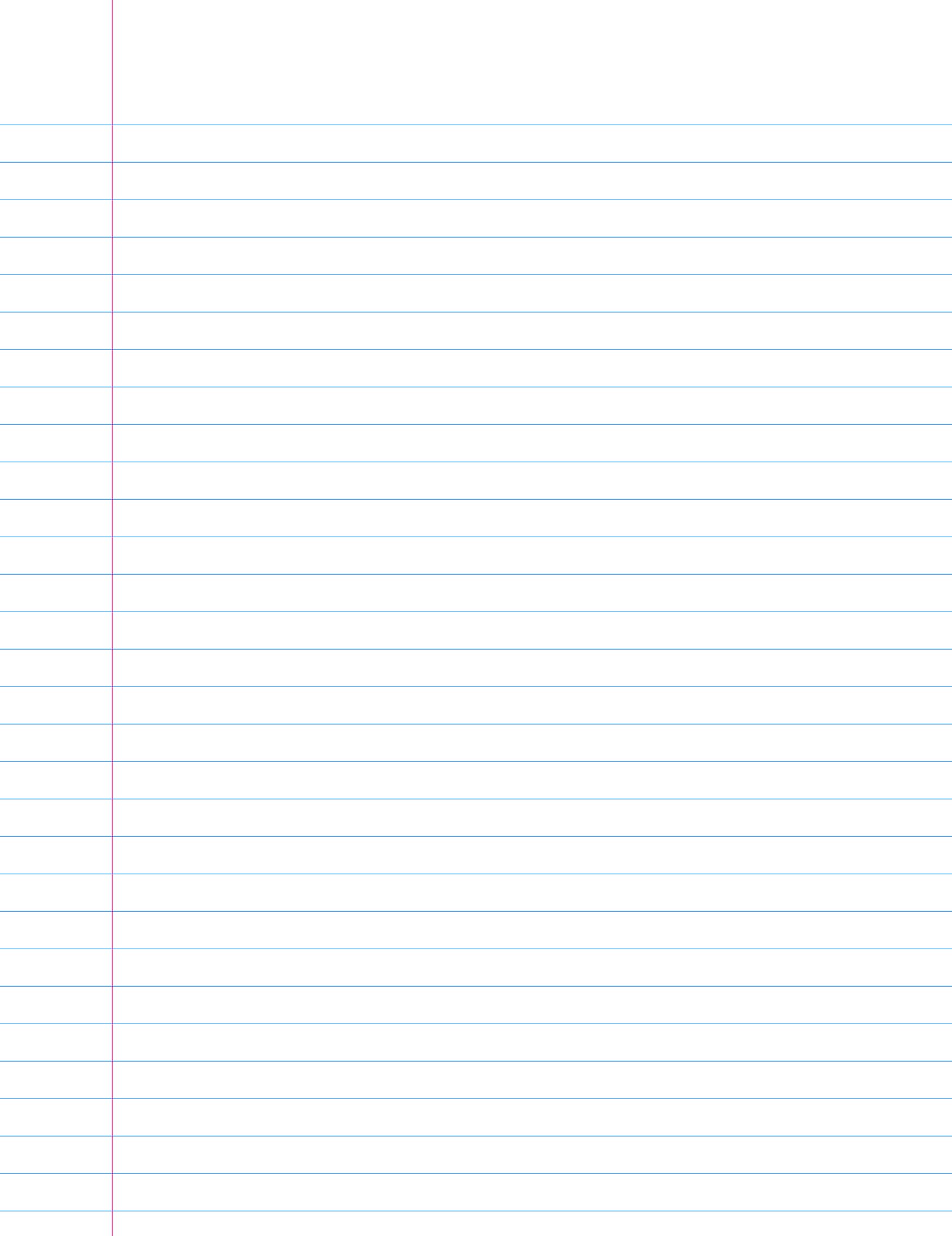
Consider the Improvement of A Neural Network Performance. See Eqn(24) from the white paper.

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

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

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

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

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

$f_{x_i}$ :  
from Eqn (1),

$$\frac{\partial f}{\partial x_i} \triangleq \lim_{\delta x_i \rightarrow 0} \frac{\Delta f_{x_i}}{\delta x_i} = \lim_{\delta x_i \rightarrow 0} \frac{f(x_1, x_2, \dots, x_i + \delta x_i, \dots, x_n) - f(x_1, x_2, \dots, x_n)}{\delta x_i}$$

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

Now, define a gradient

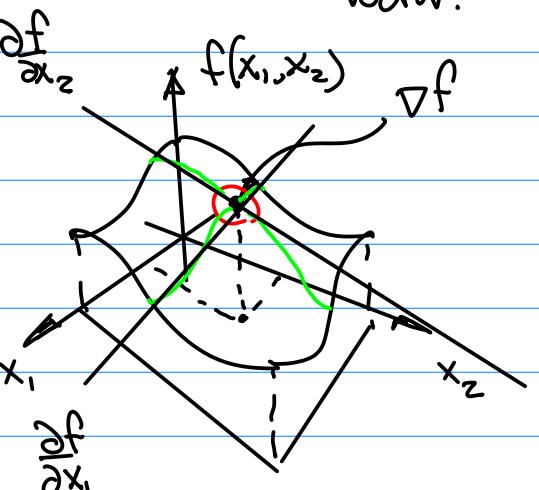
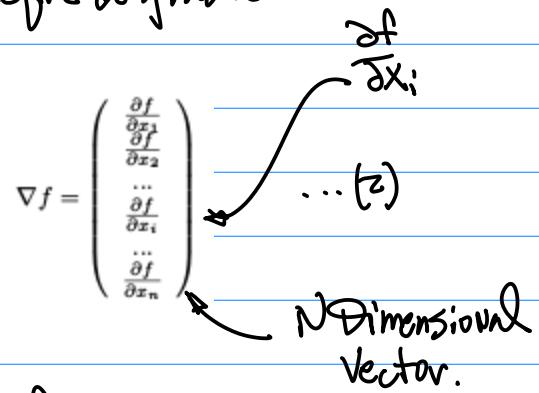


Fig.1

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

In Eqn(b), we have

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

**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.

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

Higher Order terms.

**Basic Building Blocks**

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

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

$$y = ax + b$$

Linear Function.

$\frac{df}{dx^2} \Big|_{x=x_0}$  :  $A(x-x_0)^2 \Rightarrow$

$$y = ax^2 + bx + c$$

**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.

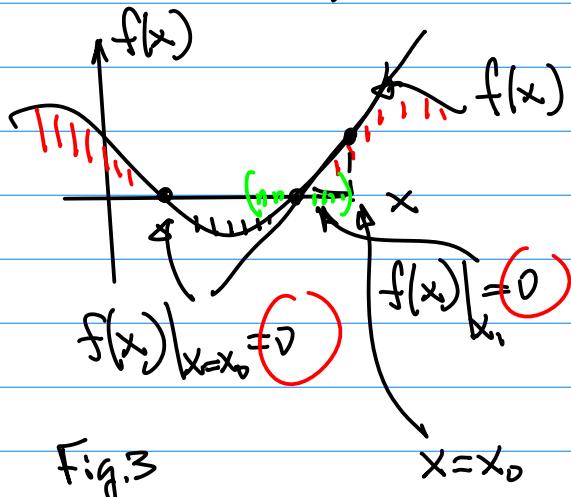


Fig.3

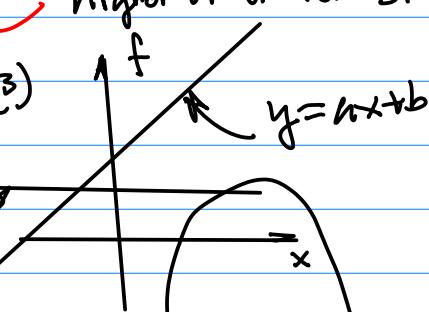


Fig.2

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

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

$$+ \frac{1}{1!} \frac{\partial f}{\partial x_2}(y-y_1) + \dots \quad \dots (1)$$

just the linear term,

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

Approximate

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

$$+ \frac{1}{1!} \frac{\partial f}{\partial x_2}(y-y_0)$$

$$- \frac{1}{(n-1)!} \frac{d^{n-1} f}{dx^{n-1}}(x-x_0)^{n-1} = f(x_0, y_0) + \frac{\partial f}{\partial x_1}(x-x_0) + \frac{\partial f}{\partial x_2}(y-y_0)$$

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

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

$$\begin{aligned} f(x) - f(x_0) &\approx \frac{df}{dx}(x-x_0) \\ \frac{f(x) - f(x_0)}{\frac{df}{dx}} &= x - x_0 \\ \text{Step } k & \\ \text{Step } k+1 & \end{aligned}$$

Feb 11 (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 the 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)

$$f(x, y) - f(a, b) \approx (-f_{x_1}, -f_{y_1}) \cdot \begin{pmatrix} f_x \\ f_y \end{pmatrix}$$

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

Since

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

we have

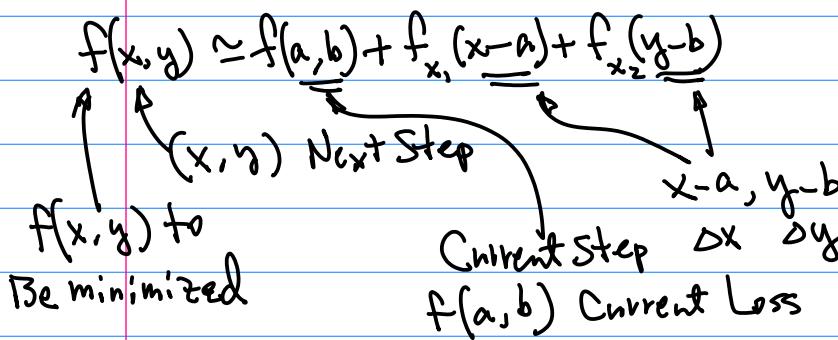
$$-(f_x^2 + f_y^2) \leq 0 \quad \dots (4)$$

which leads to

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

$$\text{Or} \quad f(x, y) \leq f(a, b)$$

Note:  $\dots (5)$



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:

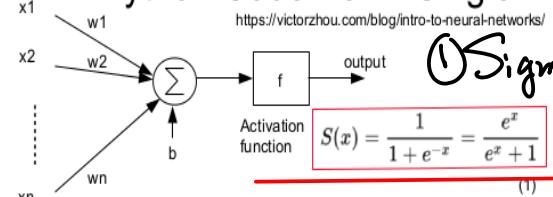
Example 1. Given

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

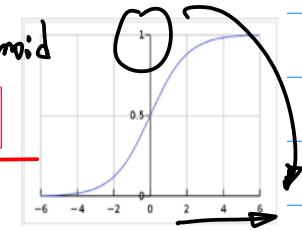
Find its gradient as follows [1]:

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

### Python Code For A Single Neuron



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



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

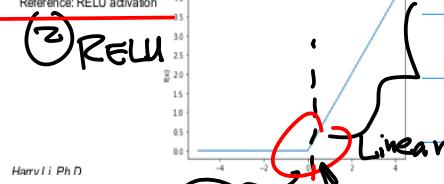
```
class Neuron:
    def __init__(self, weights, bias):
        self.weights = weights
        self.bias = bias

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

Reference: for further discussion on Sigmoid

https://deeplearningbook.org/machine-learning-glossary-and-terms/sigmoid-function

Reference: for RELU activation



Evaluate the Response of A Activation function,

$$\text{Suppose we have } h(w_i, x_i; b) = 0.5$$

Find Response of the Sigmoid, and the RELU

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

$$\text{We have } x = h(w_i, x_i; b) = 0.5 \quad \dots (6)$$

so

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

And for the RELU,

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

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

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

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

$$= ax 0.5 = 0.5a$$

Example: Python code for Simple

multi-Layer Feed forward NN.

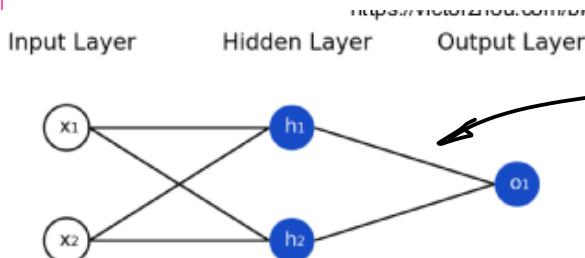
Note: 1° Derivative, gradient Computation  
are Carried out by Finite Difference Approach, But in

this Samplecode, 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



```
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()
```

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

$$y_{true} - y_{pred}$$

$$(y_{true} - y_{pred})^2$$

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

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

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

CANVAS.

- 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.

- Given the equation in Figure 1, design by drawing a single neural, for N=3, and w1=0.3, w2=0.9, w3=0.83, suppose the bias b = 0.1.
- Based on the equation in Figure 1, explain what is the function h(.), based on the parameters in Question 1 (above), for x1=0.1, x2=14, x3=7.5, find h=?
- 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:

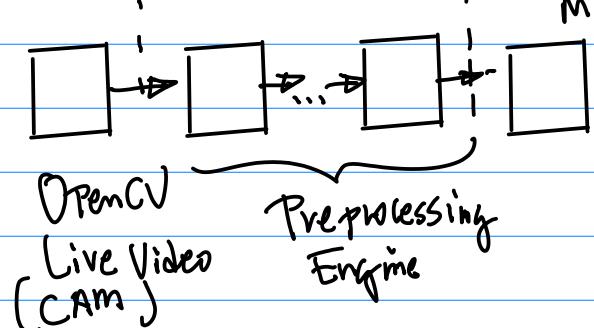
1<sup>o</sup> Use Laptop Web Cam to Capture Live Video ;  
Use A marker to write 7 Digits "ID" on a Blank (White) Paper ;

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 .

DL CNN  
MNIST



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.

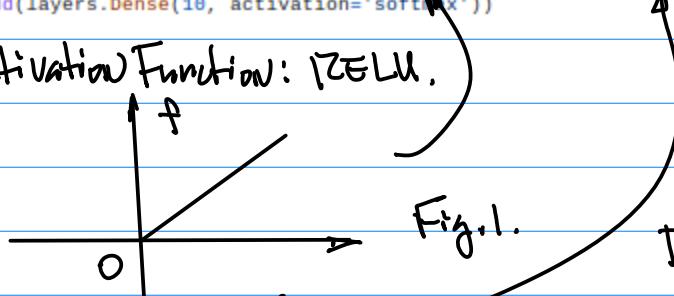


Fig.1.

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

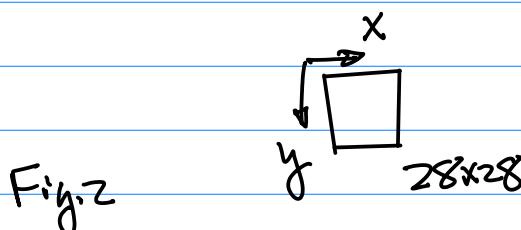
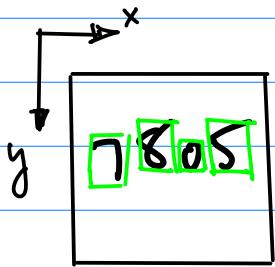


Fig.2



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

1080P Resolution; OR

720P

Resolution Resolution.  
 $1920 \times 1280$   
 $M \quad N$   
No. of Col. Per Row.  
No. of Rows per frame

Question 1: Perform Segmentation to isolate each individual Digit

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

Denote Each Bounding Box as

$B_i(x, y)$  where  $i=1, 2, \dots, K$   
... (1)

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

$R_i(x, y; w, h, \bar{x}, \bar{y}) \dots (2)$

w : Width

h : Height

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

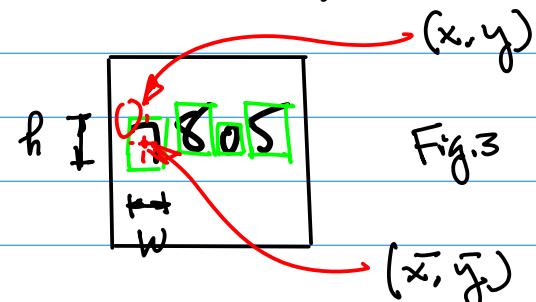


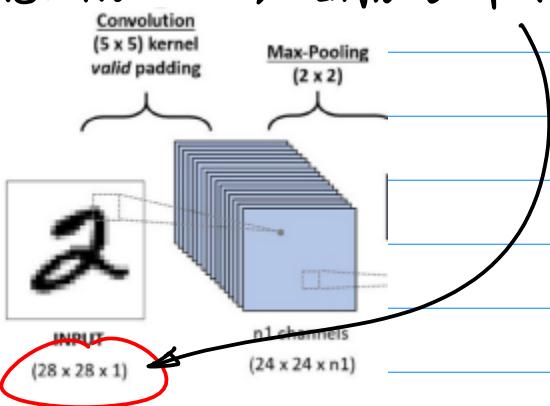
Fig.3

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.

Aspect Ratio  $\triangleq w/h \dots (3)$

Note: From line 32 Input shape:



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

Requirements: The preprocessing

Algorithm will have to:

1° Make its output for Each I.D.I.

to meet  $28 \times 28 \times 1$  Resolution

Requirement

Channel, Gray Scale  
Image.

2° As the Output Image

Size is reduced to  $28 \times 28$ , we

have to make same the R.O.I.

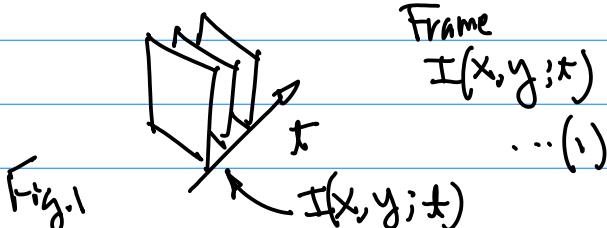
Aspect Ratio is preserved.

Feb 23 (Thu).

Example:

Question: 1° 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

$$I(x, y) = (r(x, y), g(x, y), b(x, y)) \dots (2)$$

red green blue

$r(x, y) \in [0, 255]$  ( $Z^8 = 256$ ,  
Color Plane where  $g$ : 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.

$$\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{3}$$

$I_g(x, y)$  or Simply  $I(x, y)$

(Option 1 for Segmentation: Binarization  
(Thresholding Based))

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

... (4)

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

- 2022S-108-Intro-Binary.pdf
- 2022S-108b-BinaryImageFormula.pdf
- 2022S-108c-example-binary.pdf

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

## Pattern Recognition

Multi-Dimensional  
Feature Vector

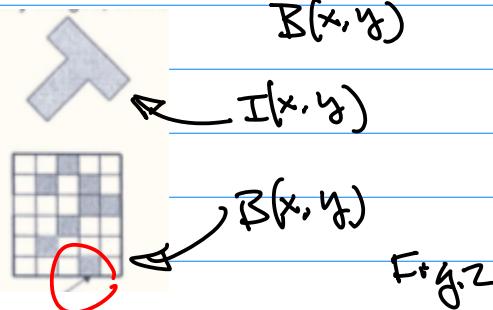
( $x_1, x_2$ ) for 2D.

( $x_1, x_2, \dots, x_n$ ) for N-Dimension.

- 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, Ph.D., SJSU

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



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  $T = 135$ .

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

Harry Li, Ph.D.

removes all contours without  
any hierarchical relationships.

http://opencvexamples.blogspot.com

## Compute Contours Features

[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

```

### 2. Contour Area

```
area = cv2.contourArea(cnt)
```



### 3. Contour Perimeter

### 5. Convex Hull

checks a curve  
corrects it  
hull = cv2.convexHull()

### 6. Checking Contours

k = cv2.isContourValid()

### 7.a. Straight Bounding Box

1 x,y,w,h = cv2.boundingRect(cnt)

### 7.b. Rotated Rectangle

1 rect = cv2.minAreaRect(cnt)
2 box = cv2.boxPoints(rect)
3 box = np.int0(box)

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

No  
an  
--

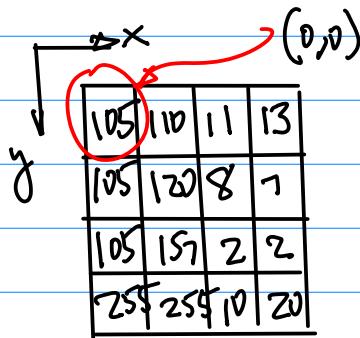
Example: Given a gray scale image  
 $I(x, y)$  below, find threshold  
 $T = 30$ ; find (1) Binary Image  
 $B(x, y)$ ; (2) Find its Size (Area)

		$\rightarrow x$
$\downarrow y$	105 110 11 13	
	105 120 8 7	
	105 157 2 2	
	255 255 10 20	

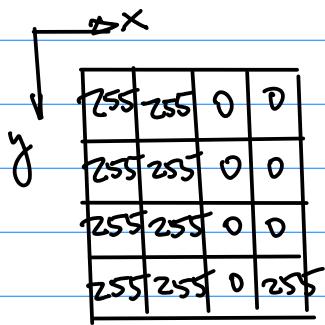
Fig. 3

$\Leftrightarrow$  From Eqn(4), and the image  $I(x, y)$ , we have

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



Fig(4a)  
 $I(x, y)$



Fig(4b)  
 $\beta(x, y)$

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.

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

$\Omega = \{(x, y) \mid x \in [0, 3], y \in [0, 3]\}$  Binary Image plane

For Digital Image, we have

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

Hence, the Area of  $\beta(x, y)$ :

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

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

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

$$\beta(0, 1) + \beta(1, 1) + \beta(2, 1) + \beta(3, 1) +$$

$$\beta(0, 2) + \beta(1, 2) + \beta(2, 2) + \beta(3, 2) +$$

$$\beta(0, 3) + \beta(1, 3) + \beta(2, 3) + \beta(3, 3) +$$

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

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

Feb 28 (Tue).

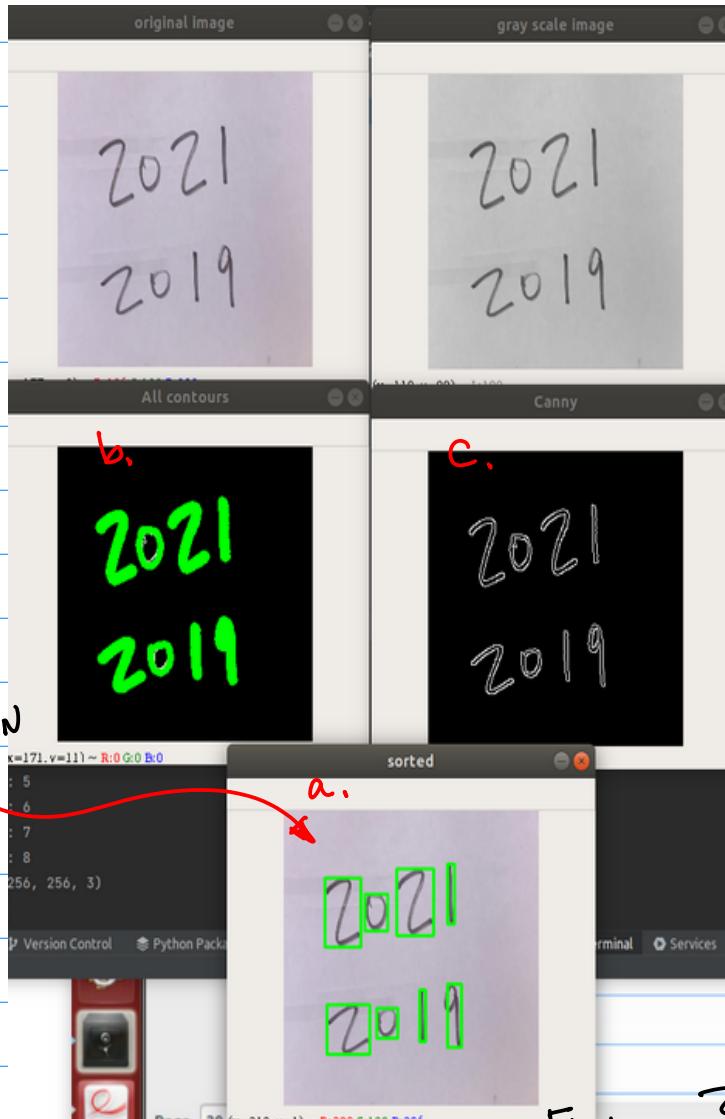
Note: 1° Update Team Project Next Tuesday;

2° Project for Handwritten Digit Recognition. Due March 16, 11:59 pm.

Requirements : (Details will be posted on CANVAS),

- a. Live Video Input from A web CAM;
- b. 4 Handwritten Digits on a white Printer Paper;

C. Display Key Steps for The processing Result.



d. R.O.I. for Each Digit. →  
Squared  $28 \times 28$  image (gray Scale) for the input for the MNIST CNN Engine.

e. DisplayCaption, placed the Captions Beneath Each R.O.I.

Please check my Reference code on the github

Example: Consider Edge Detection

Conclusion: Most adopted Edge Detectors.

1° CANNY Edge Detector

2° LOG (Laplace of Gaussian)  
Edge Detector.

Color → Color → Conversion  
Video from CAM Image to Gray Scale  
Image

CANNY  
Edge Detection

Note: 1° "imread( )" key word

```

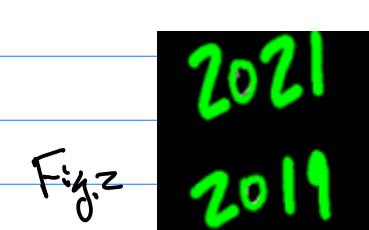
20     img = cv2.imread('test.jpg')
21     image = input('Enter image file name:')
22
23     img = cv2.imread(image, cv2.IMREAD_COLOR)
24     img = cv2.resize(img, (256, 256))
25     cv2.imshow('original image', img)
26
27     imggray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
28     cv2.imshow('gray scale image', imggray)
29
30     thresh = cv2.Canny(imggray, 100, 200)
31     cv2.imshow('Canny', thresh)

```

2° cv2.resize( )



One Possible  
Resized Image



Distortion, To Avoid the distortion,  
make sure to perform Resize with  
Equal Reduction for Both x- and y-  
dimension.

3<sup>o</sup> Line 25, CV2.imshow( )

4<sup>o</sup> Line 27, CV2.cvtColor( )

Naming Convention

lower case

5<sup>o</sup> Line 30, CV2.Canny( )

Source Image.

Lower Threshold

Higher Threshold.

(Lower Threshold ↓), picks up more  
(1st Threshold) finer details  
of the Objects  
Boundaries.

(Higher Threshold ↑), picks up Long  
(2nd Threshold) Continued  
Boundaries of  
given Objects.

Let's Consider the Theoretical  
Background

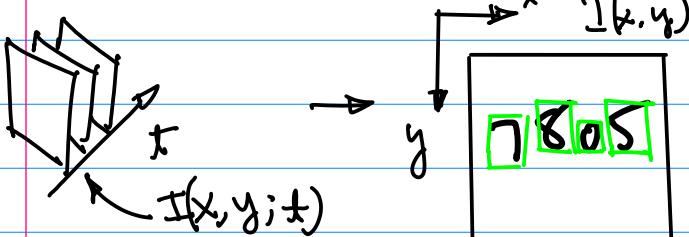


Fig.3

Edge Detection is  
based on the characteristics of  
Image Edge

Edges are the Abrupt Change of  
image intensity. if we take  
2D Fourier Transform, then the  
higher Frequency Components  
Corresponds to Edges,

Edge Detection therefore is Build  
Based Math. Operations which  
Can pick up the Changes of the  
intensity.

↓ Tool

Derivative :

$$\frac{df}{dx} = \lim_{\Delta x \rightarrow 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \quad \dots (1)$$

for Image Map.

$$\frac{df}{dx} = \lim_{\Delta x \rightarrow 0} \frac{f(x+\Delta x, y) - f(x, y)}{\Delta x} \quad \dots (2a)$$

$$\frac{df}{dy} = \lim_{\Delta y \rightarrow 0} \frac{f(x, y+\Delta y) - f(x, y)}{\Delta y} \quad \dots (2b)$$

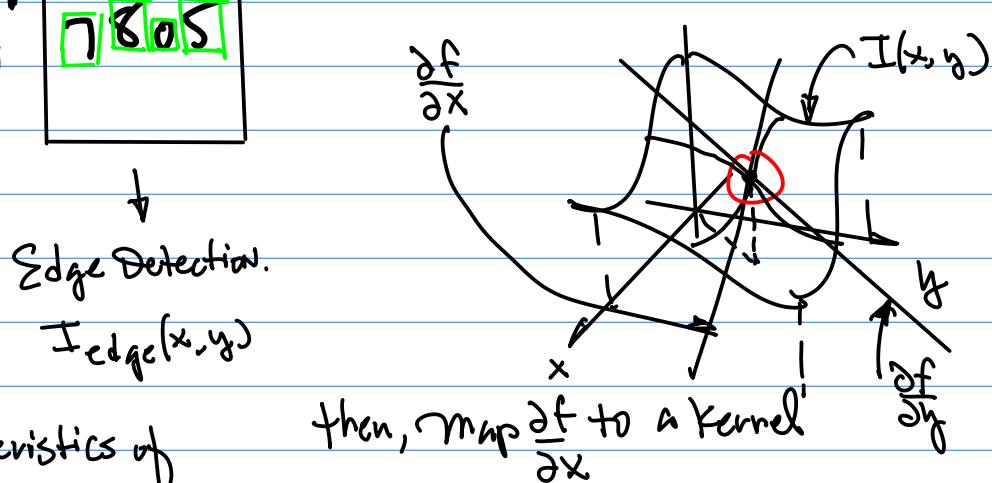


Fig.4.

Frequency Domain  $\rightarrow$  2D F.T.

(Fourier Transform)

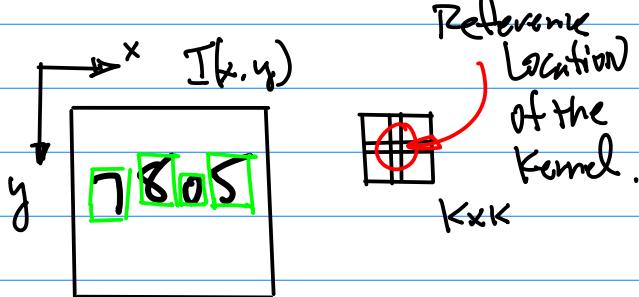
March 2nd (Th).

Note: Team Project Announcement.

Spatial Domain  $\rightarrow (x, y)$  as in

Fig. 4.

The "mapping" of the operator can be accomplished by using  $K \times K$  Kernel, e.g., Square Patch.



From Eqn(3a).

$$\frac{\partial f}{\partial x} = \lim_{\Delta x \rightarrow 0} \frac{f(x+\Delta x, y) - f(x, y)}{\Delta x}$$

$$\approx \frac{f(x+\Delta x, y) - f(x, y)}{\Delta x} \Big|_{\Delta x=1}$$

$$= \frac{f(x+1, y) - f(x, y)}{1}$$

$$= f(x+1, y) - f(x, y) \dots (3a)$$

Finite Difference Technique.

Eqn(3a). "Forward Difference"



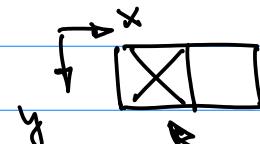
2023S-103-project-team-v2-2023-3-2.pdf

(2.2) Computer Vision algorithms for preprocessing may be needed in your design;  
 (2.3) Annotation and Training are plus in addition to just running the github code.

Must be a part of your Project

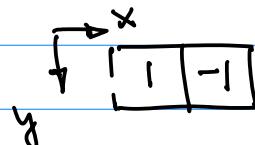
Example: Continuation

Create A Kernel



From Eqn(3a),

$$\begin{aligned} & f(x+1, y) - f(x, y) \\ & = 1 \cdot f(x+1, y) + (-1) \cdot f(x, y) \end{aligned}$$

Map them to the  $K \times 1$  Patch.

Let's Now take a look at "Backwards Difference"

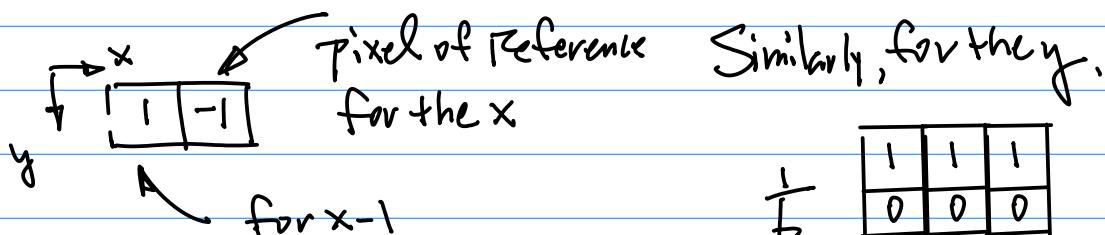
$$\frac{\partial f}{\partial x} \approx \frac{f(x, y) - f(x-1, y)}{1} = f(x, y) - f(x-1, y)$$

VR, equivalently,

$$\frac{\partial f}{\partial x} \approx \frac{f(x-1, y) - f(x, y)}{1} = f(x-1, y) - f(x, y)$$

Choose the 2nd one ... (4)

Hence, its kernel is



Add 2 Kernels together.

$$\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \quad \left\{ + \right\} \rightarrow \frac{1}{2}$$

for  $y$ .

Now, quick Review.

Color Image  $\rightarrow$  Grayscale Image

Edge Detection  $\rightarrow$  Binarization  
↓ Canny  
Contour Analysis

$$\begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \quad \dots (5)$$

So, the kernel is  $3 \times 1$  for  $\frac{\partial f}{\partial x}$ .

Similarly for  $y$ , we have the kernel for the central difference

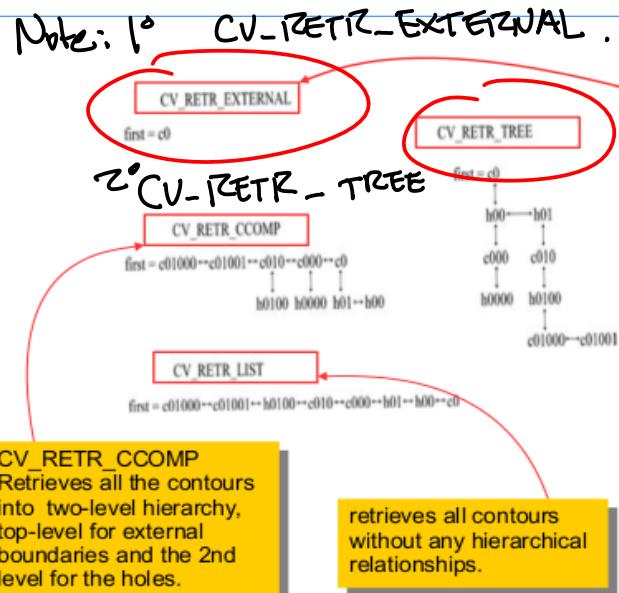
$$\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \quad \dots (b)$$

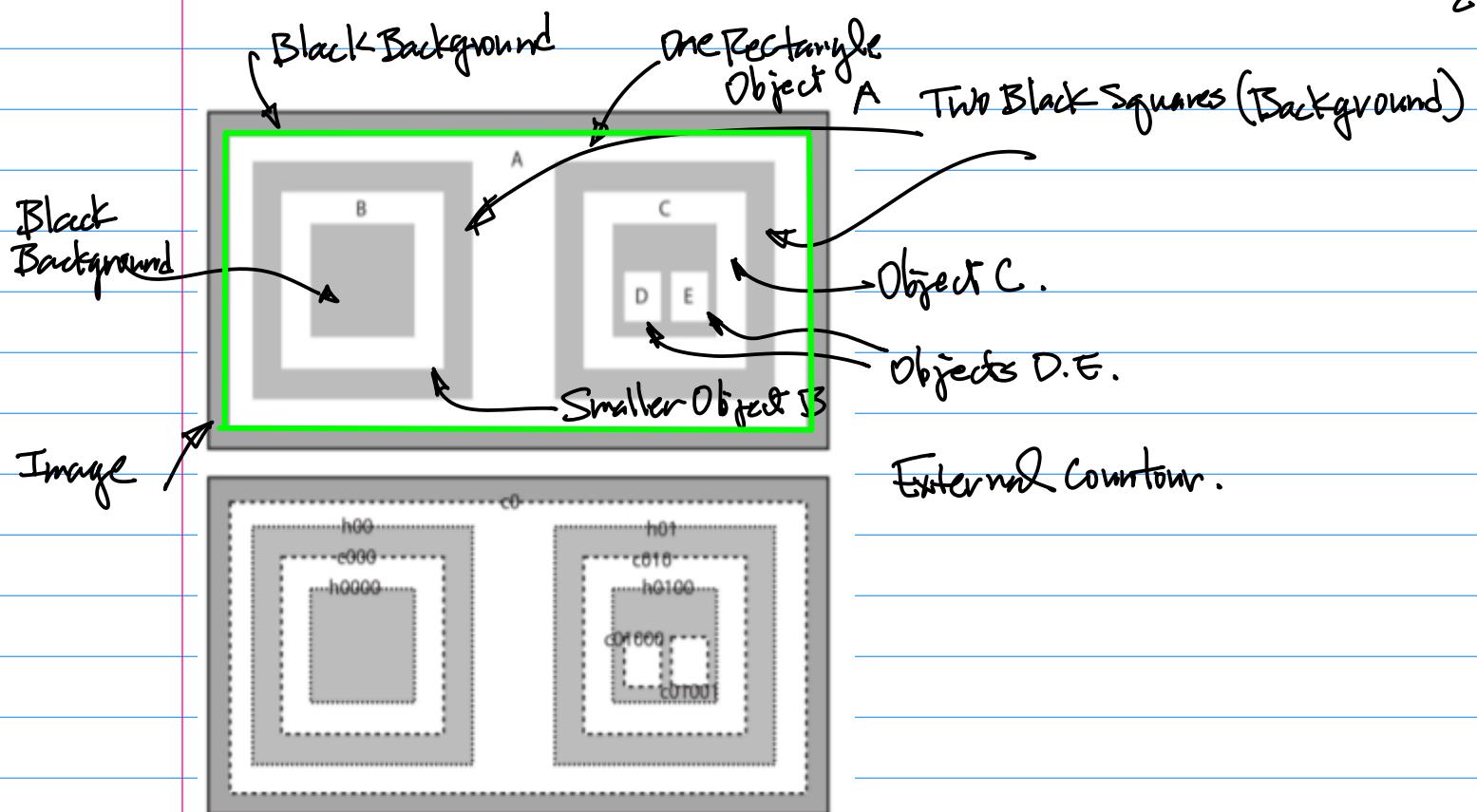
Let's Count one row above and one beneath, so

$$\frac{1}{2} \times \frac{1}{3} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \rightarrow \frac{1}{6} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \dots (7)$$

for the  $x$ .

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





## Tree Contours.

- ✓ One Object, "C" Contour, φ Index for the first ~.

first = c0  
↓  
h00 → h01 ✓ Two Holes, "H", φ Index for the first ~.  
↓  
c000 ✓ c010 ✓ Segmental Order, φφ, φl, ...  
↓  
h0000 ✓ h0100 ✓ Index from its Parent  
↓  
c01000 → c01001 ✓ 1st H. 2nd H.

Note: 1<sup>o</sup> please memorize this API function.

# OpenCV Sample Code

Cv2.findContours( )

→ a. Source image

## b External Contour.

### c. Key Word.

2022S-109c-v2-localization-contours.py

```
33 contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
34 # Draw contour
35 thresh1 = cv2.cvtColor(thresh, cv2.COLOR_GRAY2BGR)
36 cv2.drawContours(thresh1, contours, -1, (0, 255, 0), 5) # all contours
37 cv2.imshow('All contours', thresh1)
```

Zº. Cv2.drawContours(

a. Target image (Source image) to draw the Contours on.

b. External Contour  
found in the previous  
Step.

B, G, R.  
S | ZSS  
Bright Green

Note: 1 week from today.

Quiz/Homework 2 pts.

Note: Go through all the contour components.

```

40 for i in range(len(contours)):
41     [x, y, w, h] = cv2.boundingRect(contours[i])
42     # Filter out smaller noise regions
43     if (w > 5) & (h > 25):
44         print("i:", i)
45         cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
46 cv2.imshow('sorted', img)

```

Location,

w: width; h: height.

March 7th (Tuesday).

Note: Quiz (1 pt) On-Line.

1° Scope: Math. Formulation  
of Homework Assignment  
on Python Program.

2° Format: Hand Calculation  
plus coding.

3° Need Bring Printer Paper.

Write your answer on the

Paper, Take a photo (photos)

↳ Screen Capture of the Code  
execution. → Personalized  
Identifier.

↳ Handwritten Answer → photos.

a). pdf ← jpg ← png

b) Screen Capture(s) → pdf  
jpg, png

c) Combine the multiple pdf's  
to a single pdf.

First-Last-4Digits-Cmpe258.zip

cv2.boundingRect( )

6° Submission to CANVAS.  
~15 minutes.

Midterm Exam: 23rd Thursday.

Example: Continuation of Preprocessing.

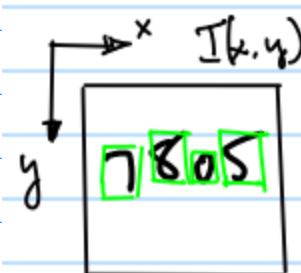


Fig. 1.

Read the P.D.I. from  
the Image I(x,y)

→ Filter Out  
the Areas  
that is too  
Small, 5 pixels  
So is for the  
"too Big" P.D.I.  
(for Example  
 $\geq 4000 \times 4000$ )

Localized P.D.I.



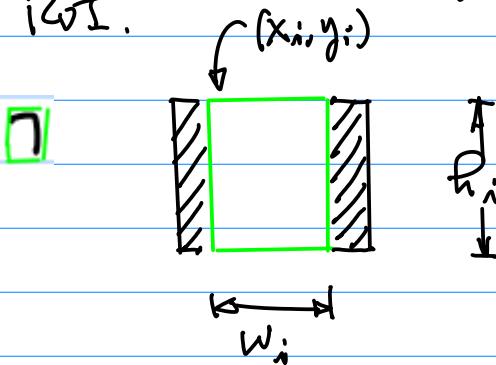
Question: How to  
fit each P.D.I  
Digit to DCNN

Engine.? MNIST 28x28x1

Squared Image  
Resolution

To fulfill the Requirements,

1, make the ROI as a squared ROI.



Square Image Using OpenCV Resize function.

Example: Computation of  $\bar{x}, \bar{y}$ .

Definition (see Ref from the Class github). Weighted with  $x$

$$\bar{x} = \frac{\iint_{\Omega} x \cdot f(x, y) dx dy}{\iint_{\Omega} f(x, y) dx dy} \quad \dots(1)$$

Area.

$$\bar{y} = \frac{\iint_{\Omega} y \cdot f(x, y) dx dy}{\iint_{\Omega} f(x, y) dx dy} \quad \dots(2)$$

for the Digital Image,

$$\bar{x} = \frac{\sum_{y=1}^N \sum_{x=1}^M x \cdot B(x, y)}{\sum_{y=1}^N \sum_{x=1}^M B(x, y)} \quad \dots(1-b)$$

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

$$\bar{y} = \frac{\sum_{y=1}^N \sum_{x=1}^M y \cdot B(x, y)}{\sum_{y=1}^N \sum_{x=1}^M B(x, y)} \quad \dots(2-b)$$

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

for Hand Calculation

$x$	1	0	1	1
$y$	105	110	11	13
	105	120	8	1
	105	151	7	7

Google

bitwise operations opencv

Images

Videos

News

Books

More

About 184,000 results (0.43 seconds)

<https://pyimagesearch.com> > Blog

OpenCV Bitwise AND, OR, XOR, and NOT - PyImageSearch

Jan 19, 2021 — Bitwise operations function in a binary manner and are represented grayscale images. A given pixel is turned "off" if it has a value of zero, ...

<https://docs.opencv.org> > tutorial\_py\_image\_arithmetics

Arithmetic Operations on Images - OpenCV

Learn several arithmetic operations on images, like addition, subtraction, bitwise oper and etc. Learn these functions: cv.add(), cv.addWeighted(), etc



2. Resize the Resulted Image to A

(1,1) for Hand Calculation!

	x			
y	255	255	0	0
	255	255	0	0
	255	255	0	0
	255	255	0	255

From PP.19.

Fig(4b)

 $B(x, y)$ 

From Egn.(1-b),

$$\bar{x} = \frac{\sum_{y=1}^N \sum_{x=1}^M x \cdot B(x, y)}{\sum_{y=1}^N \sum_{x=1}^M B(x, y)} \dots (1-b)$$

$$\text{Where } \sum_{y=1}^N \sum_{x=1}^M B(x, y) = 9$$

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

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

$$= 1 \cdot B(1, 1) + 2 \cdot B(2, 1) + 3 \cdot B(3, 1) + 4 \cdot B(4, 1) \quad \checkmark$$

$$1 \cdot B(1, 2) + 2 \cdot B(2, 2) \quad \checkmark$$

$$+ 3 \cdot B(3, 2) + 4 \cdot B(4, 2) \quad +$$

$$1 \cdot B(1, 3) + 2 \cdot B(2, 3) \quad \checkmark$$

$$+ 3 \cdot B(3, 3) + 4 \cdot B(4, 3) \quad +$$

$$1 \cdot B(1, 4) + 2 \cdot B(2, 4)$$

$$+ 3 \cdot B(3, 4) + 4 \cdot B(4, 4) \quad \checkmark$$

$$= 3 + 3 + 3 + 7 = 16$$

$$\therefore \bar{x} = 16/9$$

/ 2022F-105b-#6mnist-numerals-ch02.py

Architecture, PP.5 (Notes).

$$+ 3 \cdot B(3, 1) + 4 \cdot B(4, 1) \quad +$$

Note: 1<sup>o</sup> Keras.datasets , Note 2<sup>o</sup>. Images for Training with its Labels  
 (Annotation) Images for Testing with its Labels

```

12 from keras.datasets import mnist
13
14 #-----load data-----
15 # MNIST dataset pre-loaded in Keras, in the form of 4 Numpy arrays:
16 (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
17
18 train_images.shape
19 test_images.shape
20
  
```

Then,

```

18 train_images.shape
19 test_images.shape
20
21 train_labels
22 test_labels
23
24 len(train_labels)
25 len(test_labels)
  
```

Note 3<sup>o</sup>. Architecture ; a) models.Sequential() ; b) network.add(); Define/Build CNN

```

27 #-----build neural network-----
28 from keras import models
29 from keras import layers
30
31 network = models.Sequential()
32 network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
33 network.add(layers.Dense(10, activation='softmax'))
34
35 network.summary()
  
```

Note 4<sup>o</sup>. print the Architecture.

Project on Handwritten Digits  
 is due on March 21st (Tue).

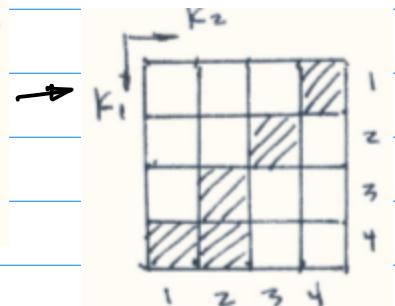
March 14 (Tue) Note: Convolution  
 homework & key will be posted  
 within 24 hrs.

Example: Binary Processing. Ref

2022S-108c-example-binary.pdf

Binarization of the Image Below

$T=135$			
$y + h(x,y)$			
70	70	80	59
75	85	140	121
90	210	101	90
211	209	115	85



(\*) Find  $\bar{x}$ .

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

So, in discrete form,

$$\bar{x} = \frac{\sum_{K_1=1}^4 \sum_{K_2=1}^4 K_1 B(K_1, K_2)}{\sum_{K_1=1}^4 \sum_{K_2=1}^4 B(K_1, K_2)}$$

Note: Always Keep the decimal Number  
 $\bar{x} = \frac{14}{5} = 2.8 \approx 3$

Generalize Computation for  $B(x,y)$   
 manipulations.

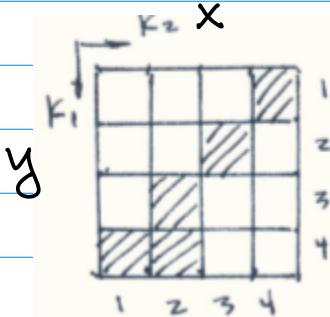
Ref

2022S-108b-BinaryImageFormula.pdf

To Replace the integration

$$m_{pq} = \frac{\sum_{y=1}^4 \sum_{x=1}^4 (x - \bar{x})^p (y - \bar{y})^q B(x,y)}{\sum_{y=1}^4 \sum_{x=1}^4 B(x,y)} \quad \dots (4)$$

Example of Hand Calculation for  $I(x,y)$   
 from the image Below



Suppos

And

$$m_{pq} = \frac{\iint_{\Omega} (x - \bar{x})^p (y - \bar{y})^q B(x,y) dx dy}{A} \quad \dots (1)$$

$$m_{12} = \frac{\sum_{y=1}^4 \sum_{x=1}^4 (x - \bar{x})^1 (y - \bar{y})^2 B(x,y)}{\sum_{y=1}^4 \sum_{x=1}^4 B(x,y)} \quad \dots (2)$$

if  $p=q=0$

$$m_{00} = \frac{\iint_{\Omega} B(x,y) dx dy}{\iint_{\Omega} B(x,y) dx dy} \quad \dots (2)$$

$$m_{10} = \frac{\iint_{\Omega} (x - \bar{x}) B(x,y) dx dy}{\iint_{\Omega} B(x,y) dx dy} \quad \dots (3)$$

We can then Compute Any meaningful moments, Such as  $p,q \leq 2$ . (in the most cases).

$$(2 - \bar{x})^1 (y - \bar{y})^2 B(2, y) + (3 - \bar{x})^1 (y - \bar{y})^2 B(3, y) + (4 - \bar{x})^1 (y - \bar{y})^2 B(4, y) \cdot ]$$

where

$$\sum_{y=1}^4 \sum_{x=1}^4 B(x,y) = 5 \times 255$$

$$\begin{aligned} & \sum_{y=1}^4 \sum_{x=1}^4 (x - \bar{x})^p (y - \bar{y})^q B(x,y) \\ &= \sum_{y=1}^4 \left[ (1 - \bar{x})^1 (y - \bar{y})^2 B(1, y) + \right. \end{aligned}$$

$$\begin{aligned}
 &= (1-\bar{x})^1 (1-\bar{y})^2 \mathcal{B}(1,1) + \\
 &\quad (2-\bar{x})^1 (1-\bar{y})^2 \mathcal{B}(2,1) + (3-\bar{x})^1 (1-\bar{y})^2 \mathcal{B}(3,1) + (4-\bar{x})^1 (1-\bar{y})^2 \mathcal{B}(4,1) + \\
 &\quad (1-\bar{x})^1 (2-\bar{y})^2 \mathcal{B}(1,2) + \\
 &\quad (2-\bar{x})^1 (2-\bar{y})^2 \mathcal{B}(2,2) + (3-\bar{x})^1 (2-\bar{y})^2 \mathcal{B}(3,2) + (4-\bar{x})^1 (2-\bar{y})^2 \mathcal{B}(4,2) + \\
 &\quad (1-\bar{x})^1 (3-\bar{y})^2 \mathcal{B}(1,3) + \\
 &\quad (2-\bar{x})^1 (3-\bar{y})^2 \mathcal{B}(2,3) + (3-\bar{x})^1 (3-\bar{y})^2 \mathcal{B}(3,3) + (4-\bar{x})^1 (3-\bar{y})^2 \mathcal{B}(4,3) + \\
 &\quad \dots \quad \dots \\
 &\quad + \\
 &\quad (1-\bar{x})^1 (4-\bar{y})^2 \mathcal{B}(1,4) + \\
 &\quad (1-\bar{x})^1 (4-\bar{y})^2 \mathcal{B}(1,4) + (2-\bar{x})^1 (4-\bar{y})^2 \mathcal{B}(2,4) + (4-\bar{x})^1 (4-\bar{y})^2 \mathcal{B}(4,4),
 \end{aligned}$$

In the Early Days of Computer Vision  
 the Combinations of the Mpgs, and  
 together with Eqn (5) & (b) Are  
 Being Investigated to Develop  
 "Shape Descriptors"

=  
 March 16 (Th).

Note: 1<sup>st</sup> Midterm Exam ON  
 the 23rd (Thursday)

Starts : 4:30 pm.  
 Ends : 5:30 pm.  
 Then : 5:30 - 5:45 pm  
 for the Submission.  
 Exclusively for the prep of the  
 Submission, Not for Continuing  
 the Exam Work.

Brief Review for the Midterm:

1<sup>o</sup> 3 Questions of Questions.

Hand Calculations,

Coding (Homeworks, Project)

2<sup>o</sup> Score: Covers the material upto today's lecture.

3. Please No Notebook Usage,  
Close-Book, Close-Note.

Exam.

One page formula sheet is allowed, But No OpenCV or T.F. API functions on the Sheet.

Please Submit your formula sheet together with your Answer

Paper.

4<sup>o</sup> Screen Capture with your personal identifier on it.

O/w the work is acceptable.

Note: please show the intermediate steps of your work, Not just the final result.

Review, Early Vision System / Preprocessing Techniques (Vision) → Inspirations for the Development of CNN.

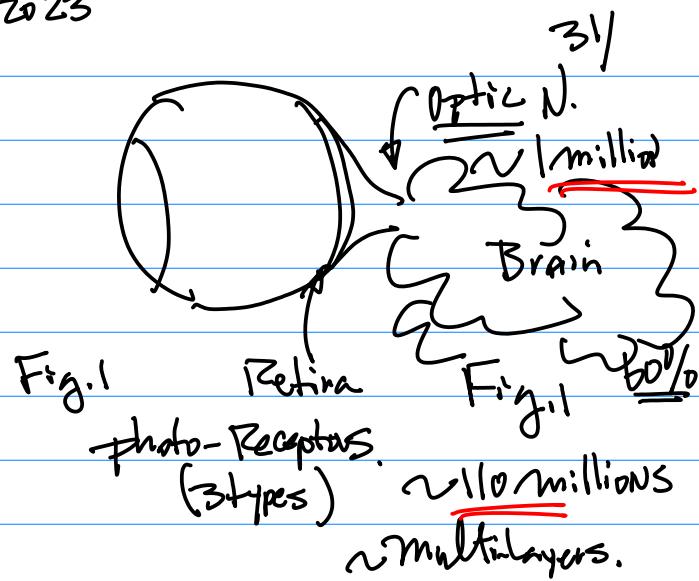


Fig.2. Feature Extraction(B) By Orientation Fine-tune Filters

