

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

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
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

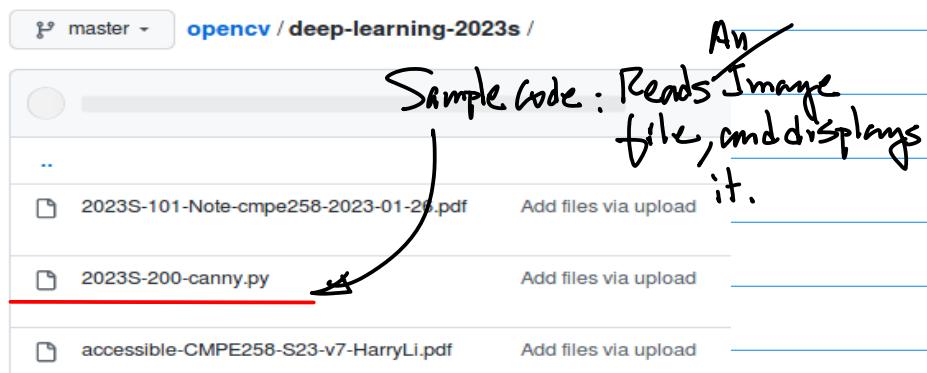
Jan 31 (Thu).

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

1. Class Ref

2. Honesty Pledge Due
this Thursday. Opt.

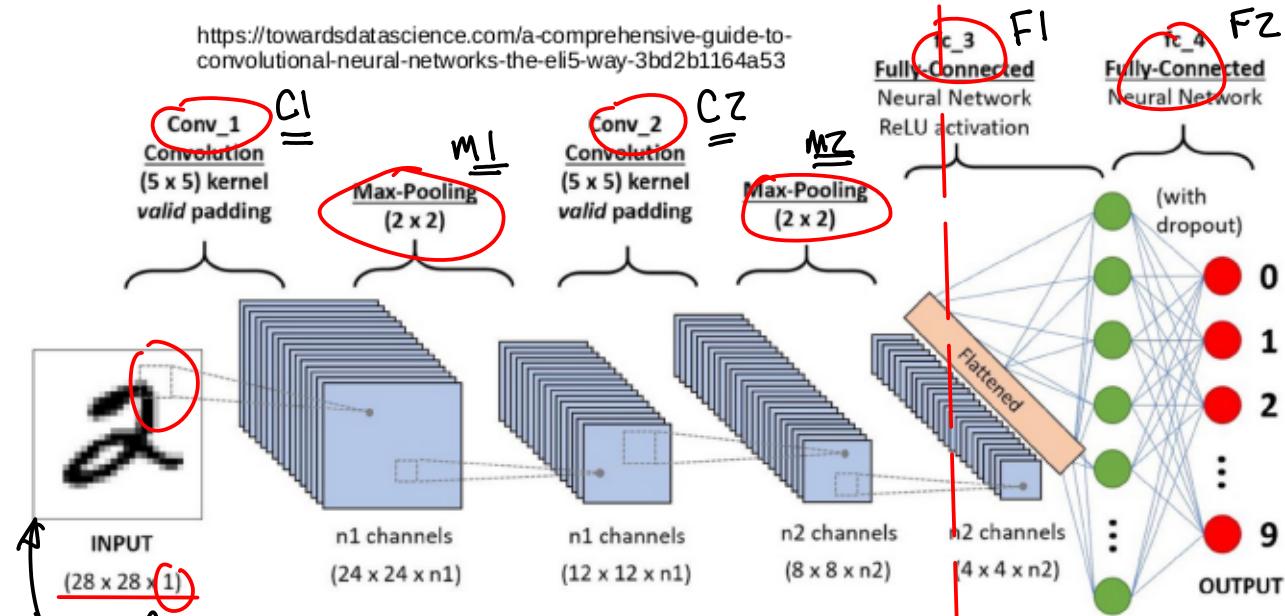
3. OpenCV & Anaconda
Installation Due A week
from Today. 1 pt.



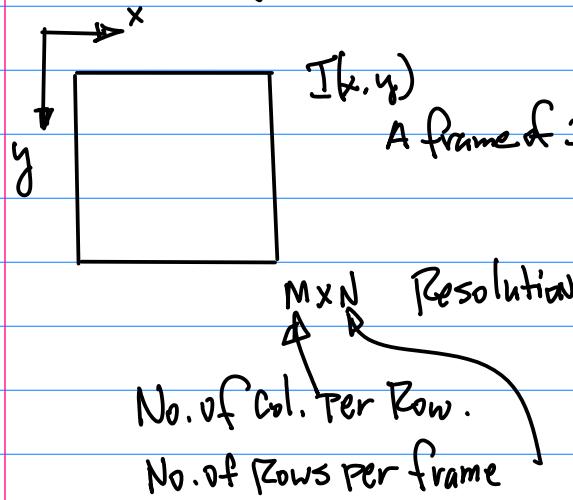
Example:

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.



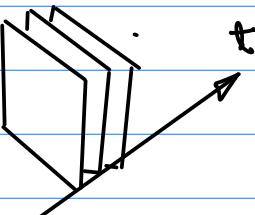
1920×1280

$M \quad N$

Pixel: picture element

for a Video Clip, DR Video Stream.

$I(x,y,t)$



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

8 bit Gray scale image
 $I(x,y) \in [0, 255]$
 Brightest, White
 Darkest, Black

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

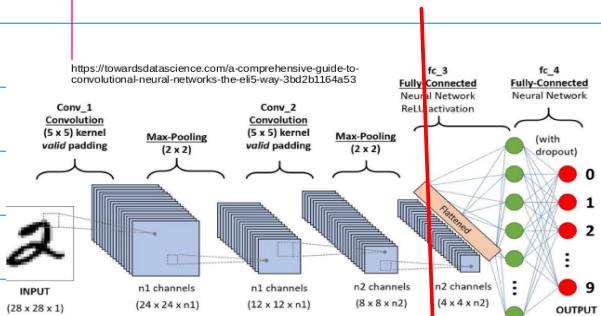
4. $\text{Conv-1} \rightarrow C \rightarrow C_1$

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 F₁

And then the 2nd Block : F₂.

Conclusion: The Architecture is defined as C₁M₁C₂M₂F₁F₂

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](#)

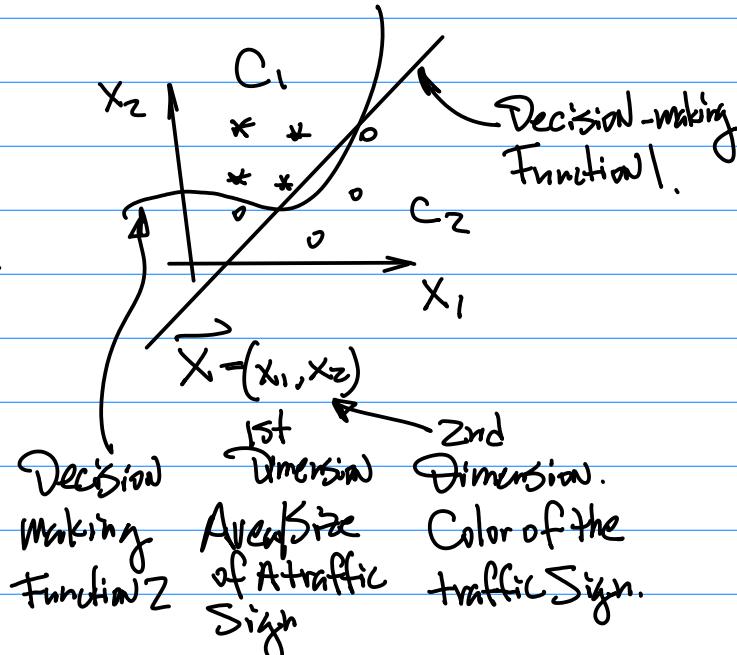
Example: Continuation of the Architecture

Question: How to Extract more features in general?

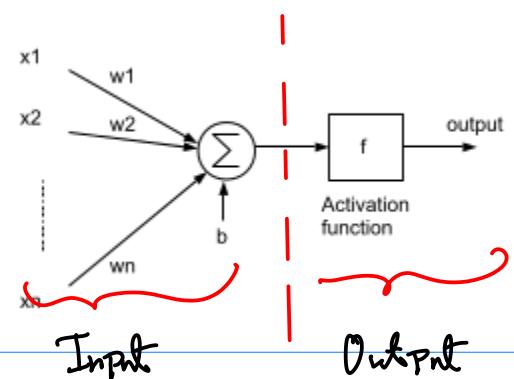
C₁M₁ Architecture V.S. C₁M₁C₂M₂?

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

C₁M₁—C₂M₂—C₃M₃.



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

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

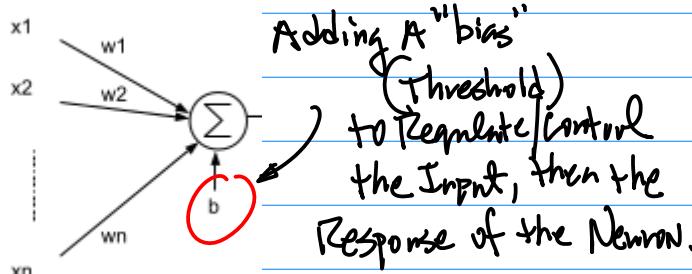
Or simply in a short hand vector form notation:

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

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^o 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

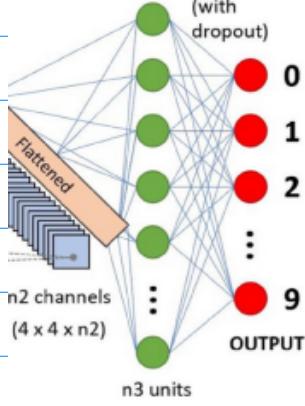
Example: Note 1. Output y_j

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

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

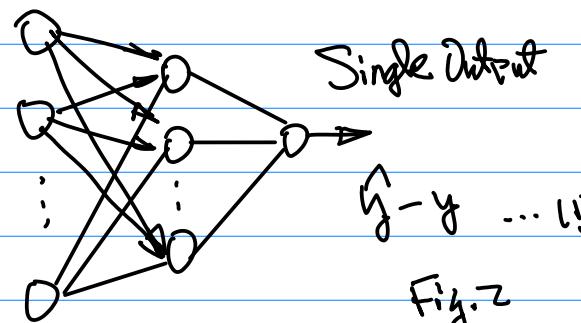
$$\hat{y}_j^i |_{j=1}^N = \hat{y}_i$$

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 | 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),

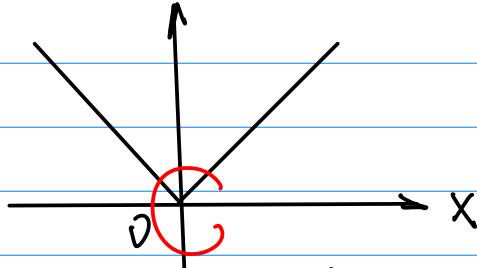
$$\begin{aligned} \sum_{i=1}^N (\hat{y}_i^j - y_i^j) &= (\hat{y}_1^j - y_1^j) + \\ &(\hat{y}_2^j - y_2^j) + (\hat{y}_3^j - y_3^j) + \dots + (\hat{y}_N^j - y_N^j) \end{aligned}$$

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.

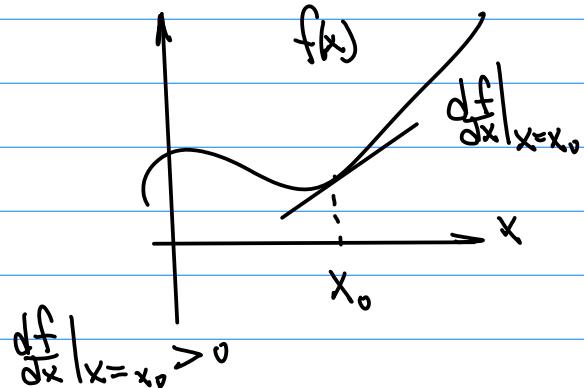
Coefficient $\frac{1}{2}$ is constant to allow a better/Simpler Manipulations in Gradient Descent Analysis.

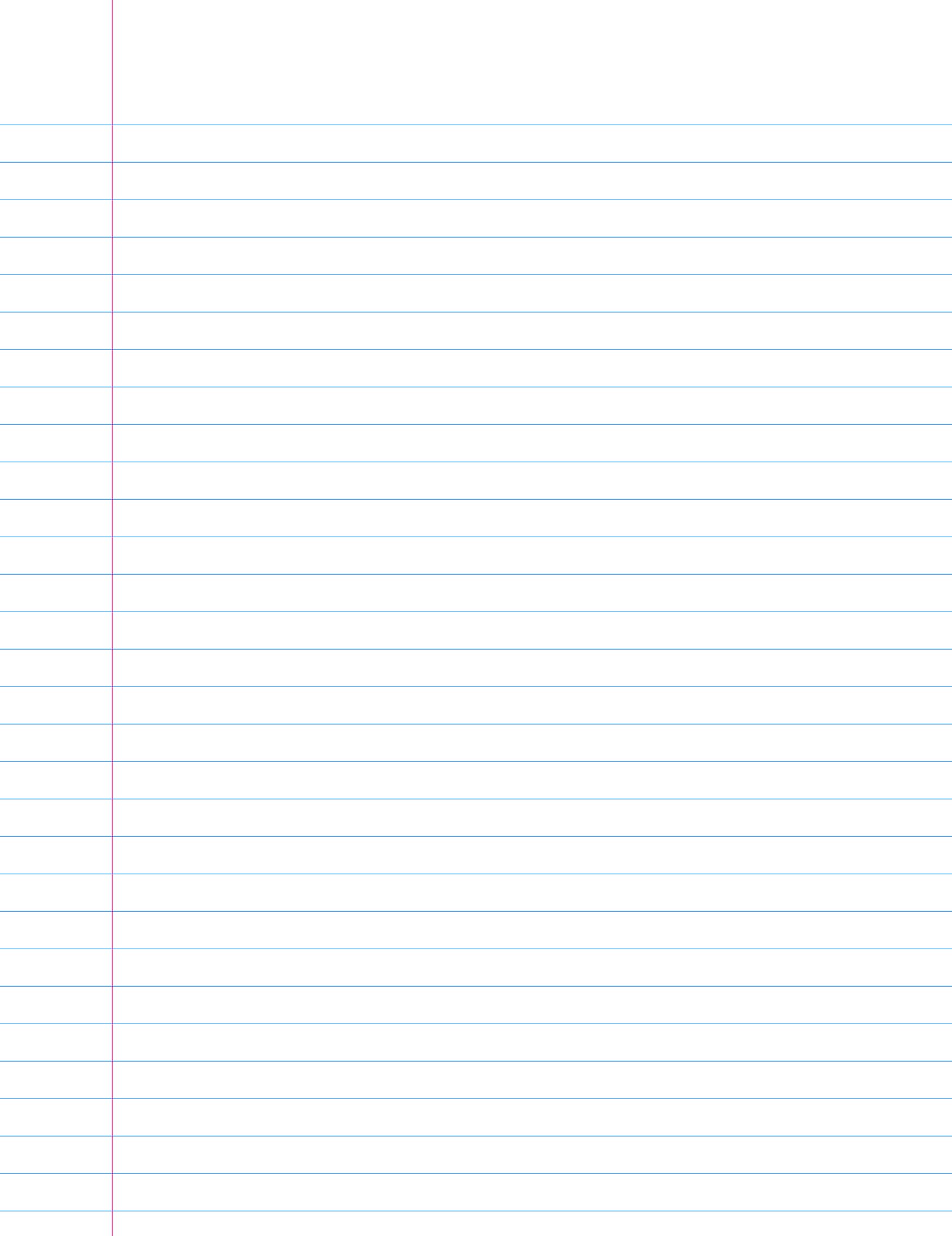
Consider the Improvement of A Neural Network Performance. See Eqn(24) from the white paper.

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

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

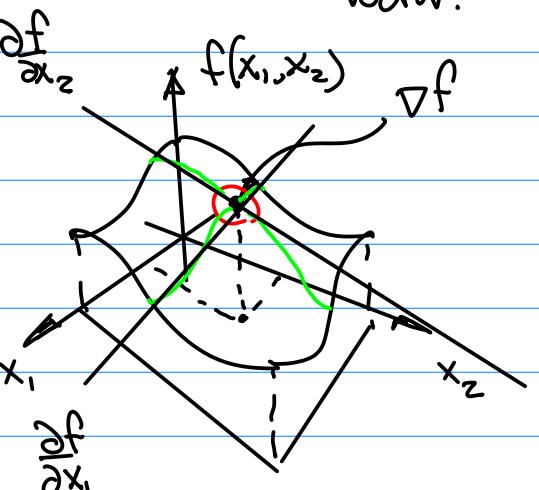
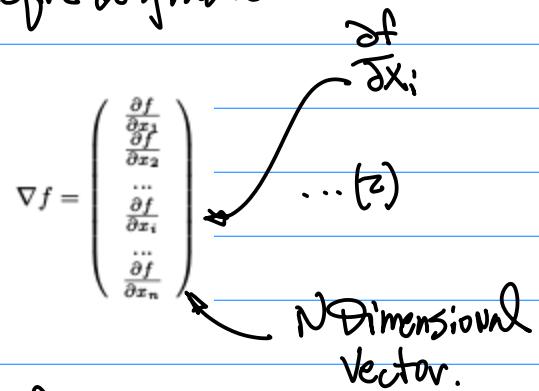


Fig.1

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.

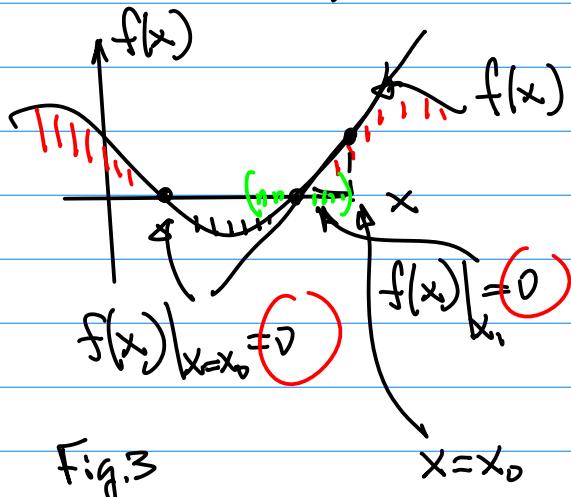


Fig.3

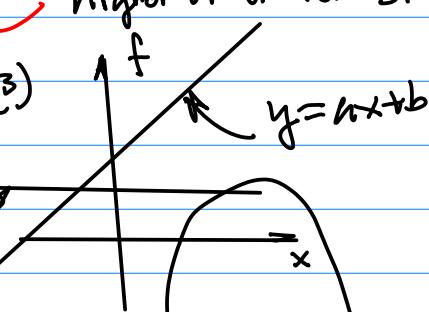


Fig.2

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

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

... (1)

just the linear term,

Approximate

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

$$\left(\frac{1}{3} \right) \frac{d^3 f}{dx^3}(x-x_0)^3 + \dots - \left(\frac{1}{(n-1)} \right) \frac{d^{n-1} f}{dx^{n-1}}(x-x_0)^{n-1} = f(x_0, y_0) + \frac{\partial f}{\partial x_1}(x_0, y_0) + \frac{\partial f}{\partial x_2}(y_0)$$

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

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

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

Step k.
Step k+1

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

from the definition of a gradient, we have

$$\nabla f(x_0, y_0) = \begin{pmatrix} \frac{\partial f}{\partial x}(x_0, y_0) \\ \frac{\partial f}{\partial y}(x_0, y_0) \end{pmatrix}$$

$$= \begin{pmatrix} f_x \\ f_{nx} \end{pmatrix}$$

Feb 1 b (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,

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

From the white paper let's ... $(x-a, y-b)$

From the white paper, let's
make

$$\Delta x_1 = x_1 - a = -f_{x_1}, \Delta x_2 = x_2 - b = -f_{x_2}$$

Therefore, substitute $\Delta x, \Delta y$ ($\Delta x_1, \Delta x_2$)
into Eqn(3-b) f_{x1}

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

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

Since

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

we have

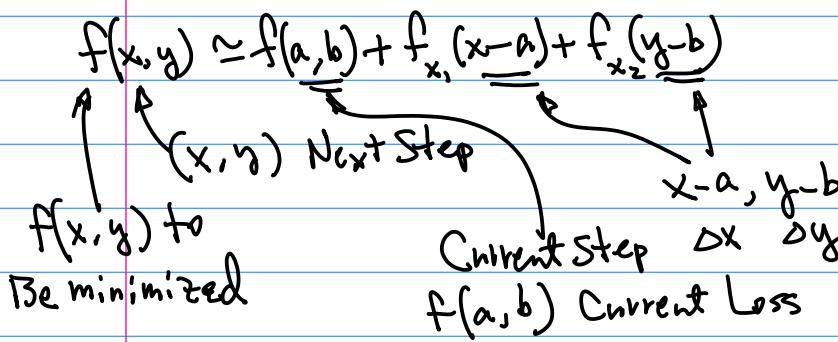
$$-(f_x^2 + f_y^2) \leq 0 \quad \dots (4)$$

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:

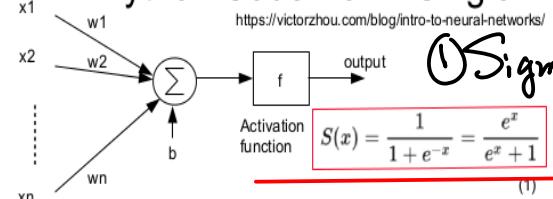
Example 1. Given

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

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



Note: 1. define activation function

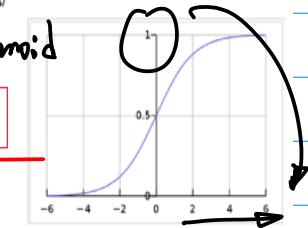
```
def sigmoid(x):
    # Our activation function: f(x) = 1 / (1 + e^-x)
    return 1 / (1 + np.exp(-x))
```

2. define a single neuron

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

    def feedforward(self, inputs):
        total = np.dot(self.weights, inputs) + self.bias
        return sigmoid(total)
```

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

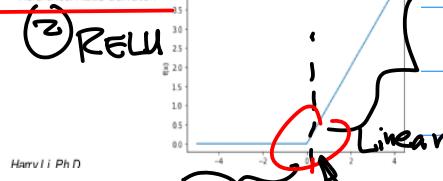


https://en.wikipedia.org/wiki/Sigmoid_function

Reference: for further discussion on Sigmoid

<https://deeplearning.net/machine-learning-glossary-and-terms/sigmoid-function>

Reference: RELU activation



Harvard PhD

Evaluate the Response of A Activation function,

Suppose we have $f_i(w_i, x_i; b) = 0.5$

Find Response of the Sigmoid, and the RELU

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

We have $x = f_i(w_i, x_i; b) = 0.5 \quad \dots (6)$

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.

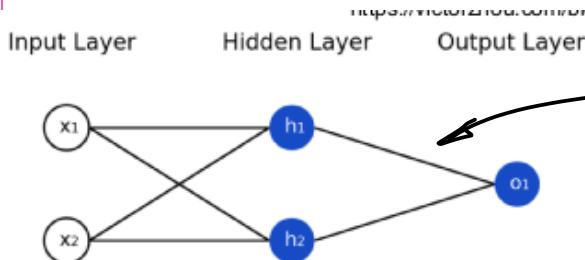
Note: 1° Derivative, gradient Computation
are Carried out by Finite Difference Approach, But in

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^o

[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^o Update your OpenCV Code to Display Video from a Live Web CAM and from a file.

Feb 21st (Tue).

Note: 1^o Team Formation update;
2^o 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^o PP.3. Architecture of CNN

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

2^o Python Code

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

Design A System with the following Requirements:

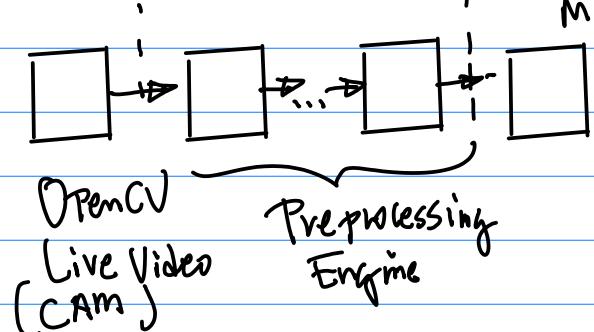
1^o Use Laptop Web Cam to Capture Live Video ;
Use A marker to write 7 Digits "ID" on a Blank (White) Paper ;

2^o 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^o 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.

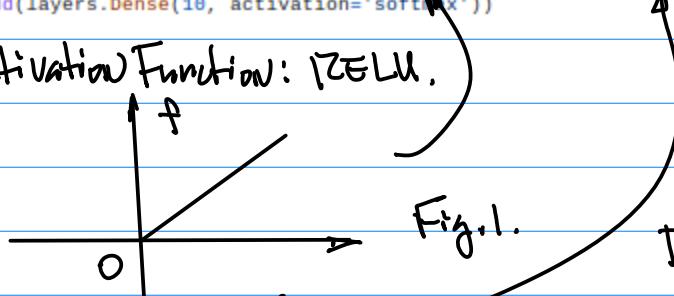


Fig.1.

z^0 input-shape = $(28 \times 28,)$

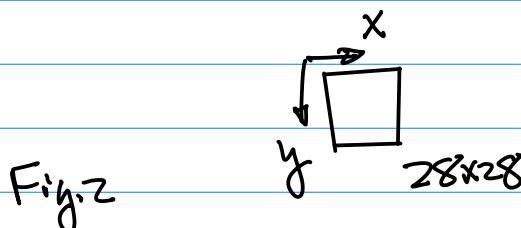
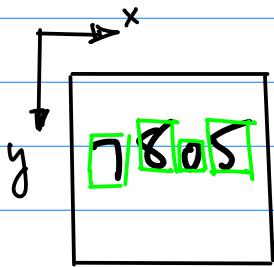


Fig.2



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

1080P Resolution; OR

720P Resolution Resolution.

720P

$M \times N$
No. of Col. Per Row.
 M
No. of Rows per frame
 N

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).

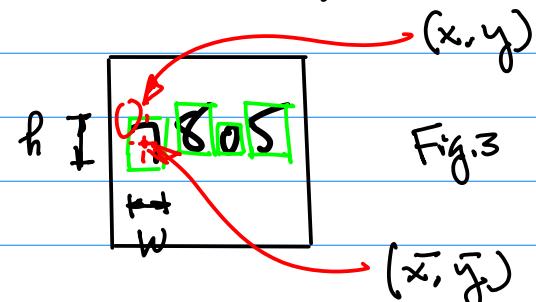
R.O.I. Denoted As

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

w : Width

h : Height

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

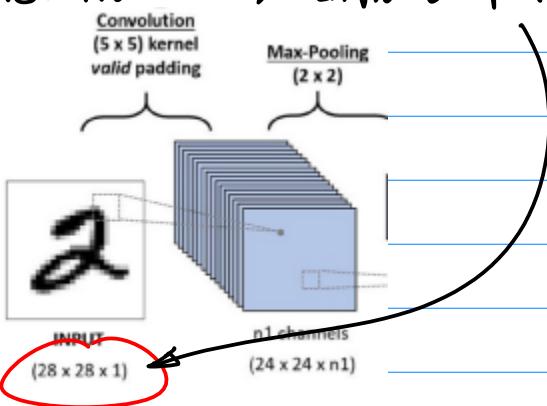


Question 3: To Be Able to Extract the digit Inside R.O.I. Even if there are Some Artifacts Around it. (Such as Background Color Variation, different Objects Superimposed on top of the Digit, etc.) .

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 28×28 , we

have to make same the R.O.I.

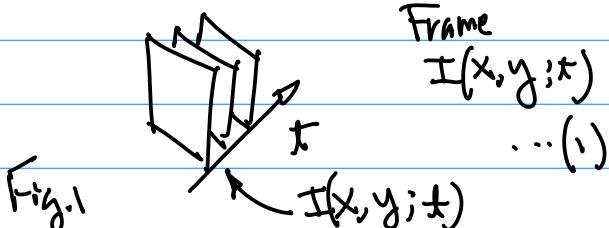
Aspect Ratio is preserved.

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

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

$$r(x, y) \in [0, 255] \quad (Z^8 = 256,$$

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))

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

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

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

Note: 1^o Techniques / Algorithms
quick Review

Pattern Recognition

Multi-Dimensional
Feature Vector

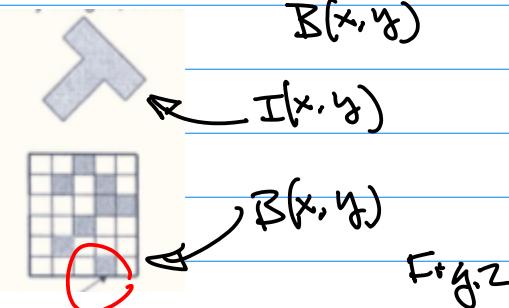
(x_1, x_2) for 2D.

(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^o Illustration of $I(x, y)$ v.s.



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 Fe

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.conv

6. Checking Cc

k = cv2.isConto

7.a. Straight B

1 x,y,w,h = cv2.b

2 cv2.rectangle(i

7.b. Rotated Re

1 rect = cv2.m

2 box = cv2.box

3 box = nn.interv

3^o 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

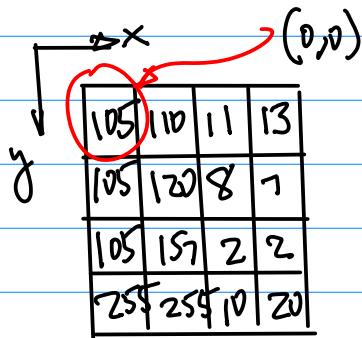
Example: Given a gray scale image
 $I(x, y)$ below, find threshold
 $T = 30$; find (1) Binary Image
 $B(x, y)$; (2) Find its Size (Area)

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

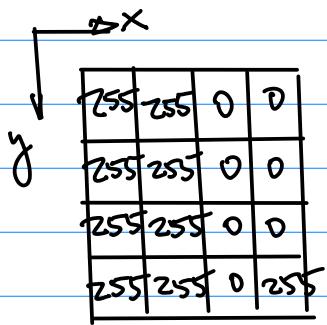
Fig. 3

\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).

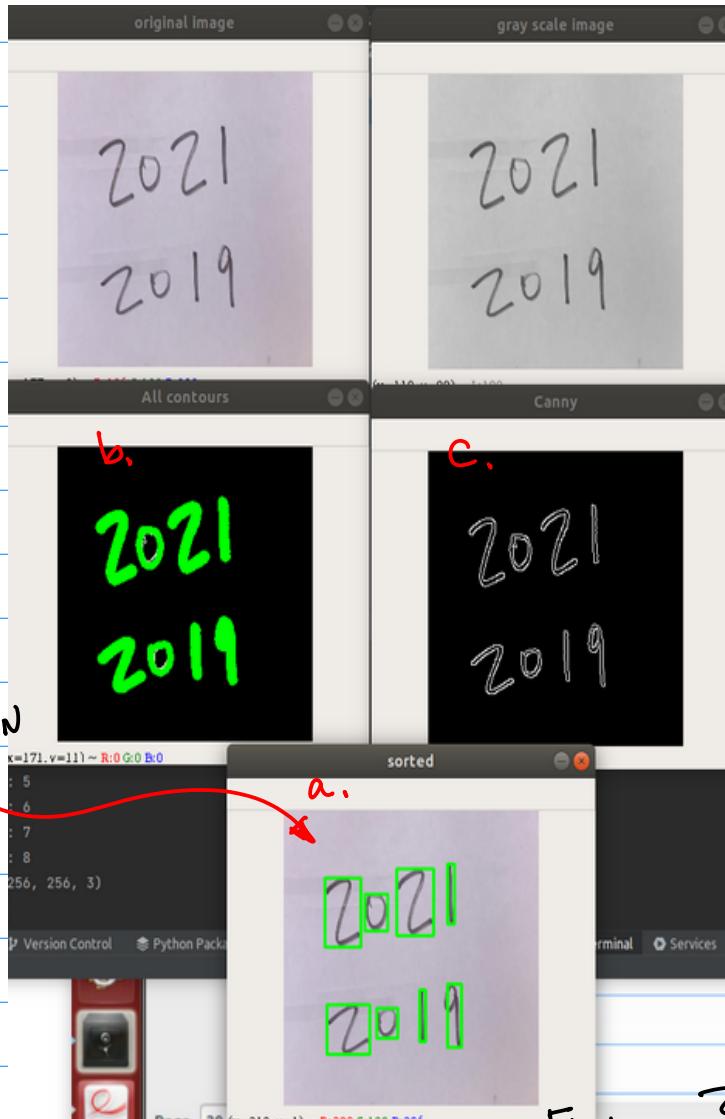
Note: 1° Update Team Project Next Tuesday;

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 28×28 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 Image to Gray
from CAM Image Scale

CANNY
Edge Detection

Note: 1° "imread()" key word

```

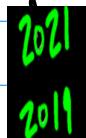
20     img = cv2.imread('test.jpg')
21     image = input('Enter image file name:')
22
23     img = cv2.imread(image, cv2.IMREAD_COLOR)
24     img = cv2.resize(img, (256, 256))
25     cv2.imshow('original image', img)
26
27     imggray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
28     cv2.imshow('gray scale 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^o Line 25, CV2.imshow()

4^o Line 27, CV2.cvtColor()

Naming Convention

lower case

5^o 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

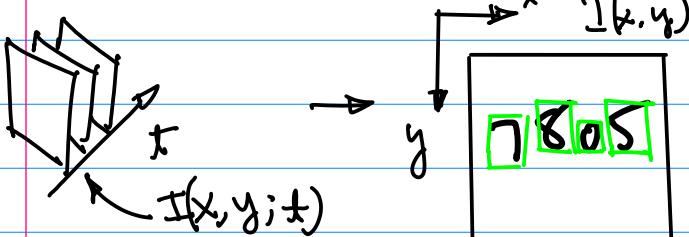


Fig.3

Edge Detection.

$I_{edge}(x, y)$

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.

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

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

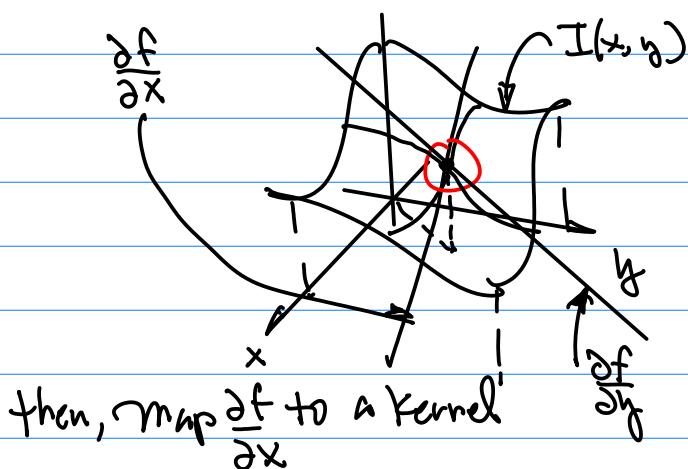


Fig.4.

Frequency Domain \rightarrow 2D F.T.

(Fourier Transform)

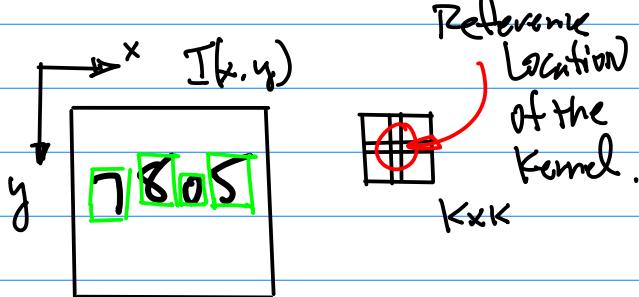
March 2nd (Th).

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).

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

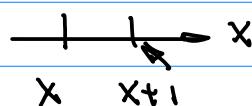
$$\approx \frac{f(x+\Delta x, y) - f(x, y)}{\Delta x} \quad |_{\Delta x=1}$$

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

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

Finite Difference Technique.

Eqn(3a). "Forward Difference"



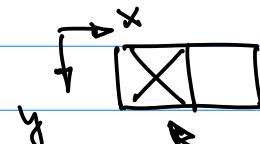
2023S-103-project-team-v2-2023-3-2.pdf

(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

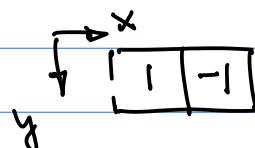
Example: Continuation

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"

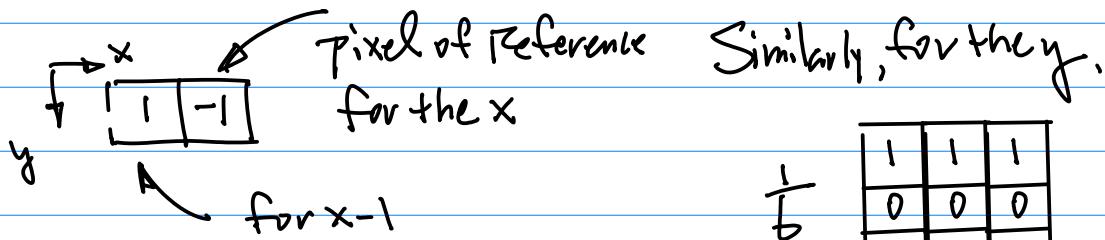
$$\frac{\partial f}{\partial x} \approx \frac{f(x, y) - f(x-1, y)}{1} = f(x, y) - f(x-1, y)$$

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

\downarrow
Edge Detection \rightarrow Binarization
 \downarrow Canny
Contour Analysis

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

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

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

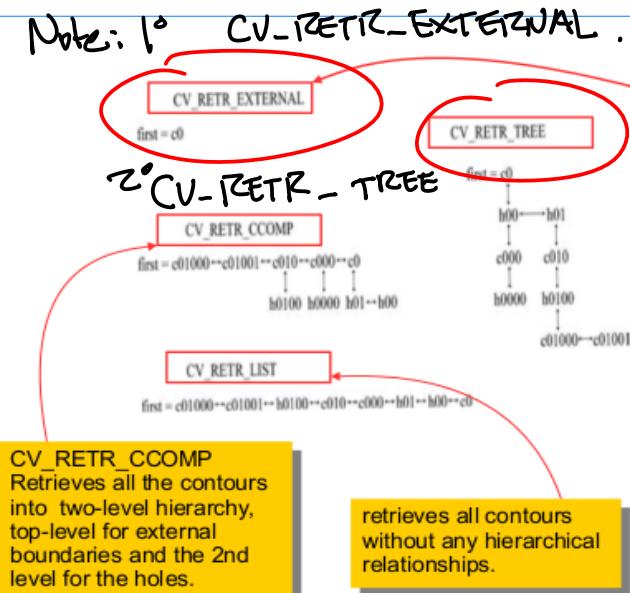
Similarly for y, we have the kernel for the central difference

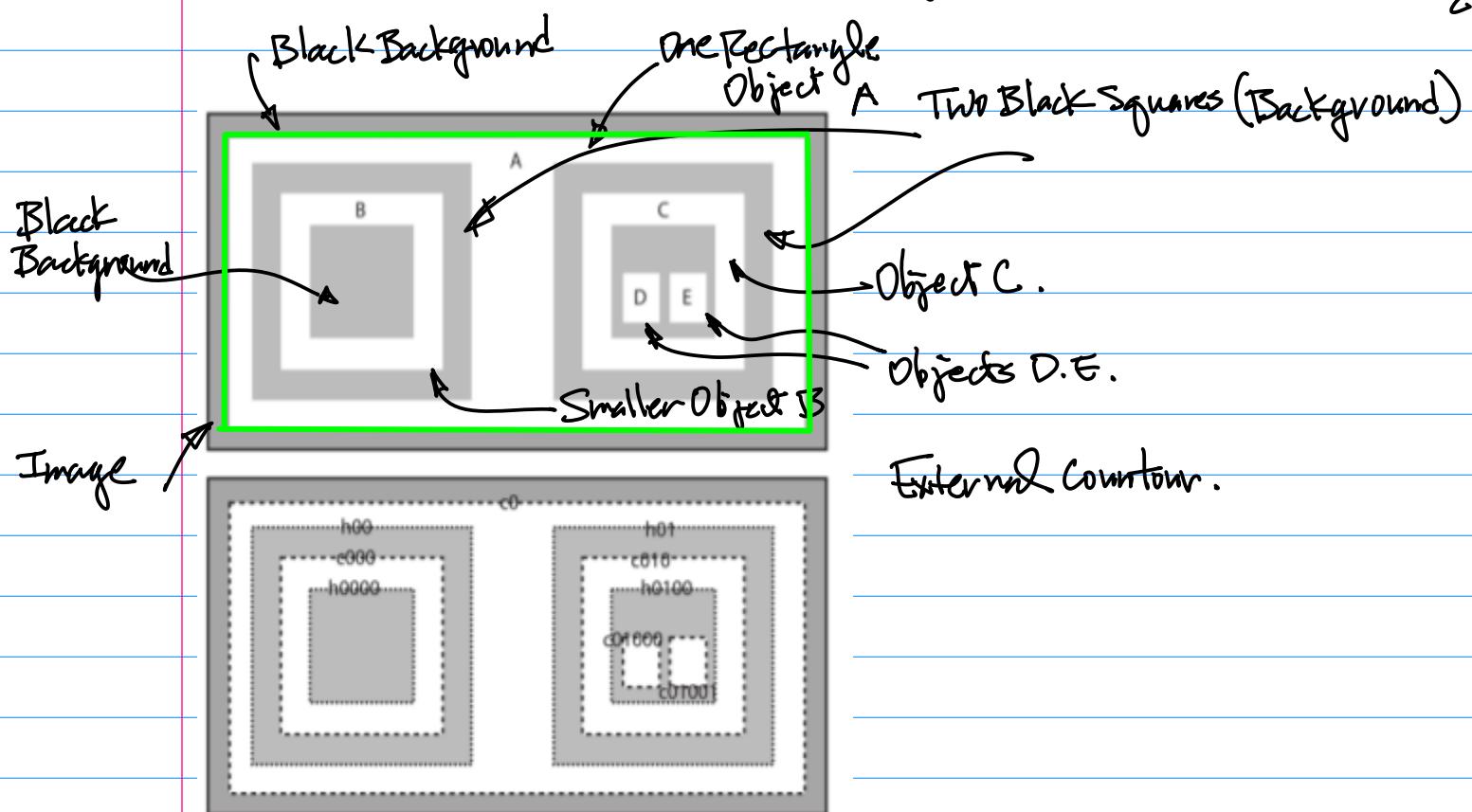
$$\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \quad \dots (b)$$

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

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

for the x.





Tree Contours.

- ✓ One Object, "C" Contour, ✓ Index for the first ~.
- ✓ Two Holes, "H", ✓ Sequential Order, ✓, ✓, ...
Index from its Parent
1st H. 2nd H.

Note: 1° please memorize this API function.

OpenCV Sample Code

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)

```

cv2.findContours()

↳ a Source image

b. External Contour.

c. Keyword.

2°. cv2.drawContours()

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

b. External Contour found in the previous Step.

B, G, R.

255

Bright Green

Note: 1 week from today.

Quiz/Homework 2 pts.

Note: Loop 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 .

cv2.boundingRect()