

CMPE258

Fall 2023

✓

August 22 (Tues)

Organizational meeting.

1. Class material on github
github/huawili

alv100	Add files via upload
deep-learning-2020S	Add files via upload
deep-learning-2022s	Add files via upload
deep-learning-2023s	Add files via upload
facial-detect	Add files via upload
lec1Capture/CMakeFiles	Delete CMakeDirectoryInformation
lecOpenCV_GL	openGL and openCV sample commi
riscv	Create readme.txt
20-2021S-0-7-1convnets-NumericalD...	Add files via upload

Course and Contact Information

Instructor(s): Harry Li

Office Location: Engineering Building, Room 267A

Telephone: (650) 400-1116 for text messaging only

Email: hua.li@sjsu.edu

Office Hours: M.W. 3:00-4:00 pm

In-Person.

Class Days/Time: Tuesdays and Thursdays 4:30-5:45 pm.

Classroom: Engineering Building Room 337

Prerequisites: CMPE 255 or CMPE 257 or instructor consent. Computer Engineering majors only.

Course Description

2. Prerequisites Requirements

Bring your Proof to the next Class.

3. Emphasis on "Deep Neural Networks",
Semantic Segmentation

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.

Note: Definition (n.b.): (†) Human Intelligence
is Symbolic Representation of
Learn + Experience.

4. Projects. 2

Plks | team project
(Semester Long) } 30% 25%
30 pts

5. In-Person Class; CANVAS is

utilized to Post Homework/Project Requirements, and to Collect the Submission.
of the Homework,
as well as for the Exams.

No

This course is an online course. The students must have Internet connectivity and Zoo his/her machine. The students must participate in the class activities and submit all assignments to SJSU CANVAS. The syllabus, faculty contact information on the syllabus, projects, and exam papers are all available on CANVAS. See [University Policy F13-2](#)

6.

Grading Information

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

7. Textbooks & References

Textbook

- Deep Learning with Python, 1st or 2nd Edition, by François Chollet, ISBN-10: 9781617294433, <https://github.com/fchollet/keras/blob/master/2018F-6-DeepLearningCh02.pdf>
- Robot Vision by B.K. P. Horn, the MIT press, ISBN 0-262-08159-8, (Hill).
- Reference textbook Learning OpenCV, Computer Vision with the OpenCV Library, O'Reilly Publisher, ISBN 978-0-596-51613-0, 2011.

8. Software Tools & Dev. Environment

Python.
Anaconda.
TensorFlow.

Rcharm.

Note: Rm268 Available per ON
Approved Basis.

August 24 (Thursday)

Note:

- CANVAS To Be up by the end of the day, Friday;
- Tools & Software To be installed (will provide "Readme" as Ref)

OpenCV

Python.

T.F. Version 2.0 or higher

Today's Topic:
Intro. to Deep Convolutional Neural Networks.

Ref: github.

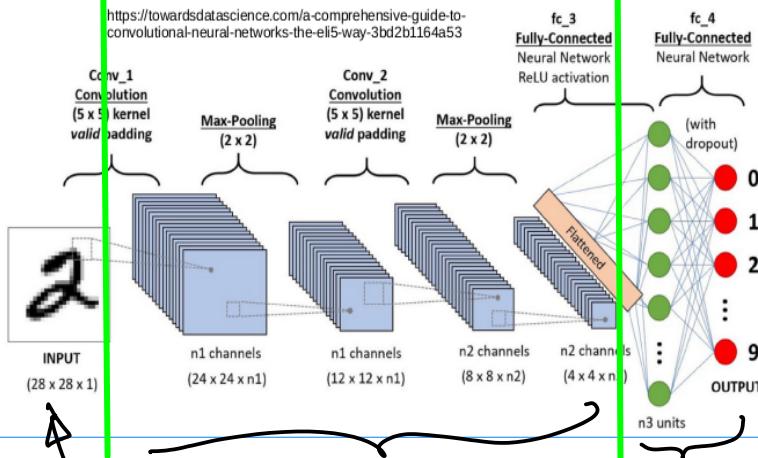
2022F-103-NN-Intro-Python-v5-2022-8-25

Note: Lab Space for the class
Rm268.

Example: Architecture Overview.

Note 1.

Illustration of A CNN for Digits Recognition



Preprocessing
Computer
Vision.

Convolution
Layers

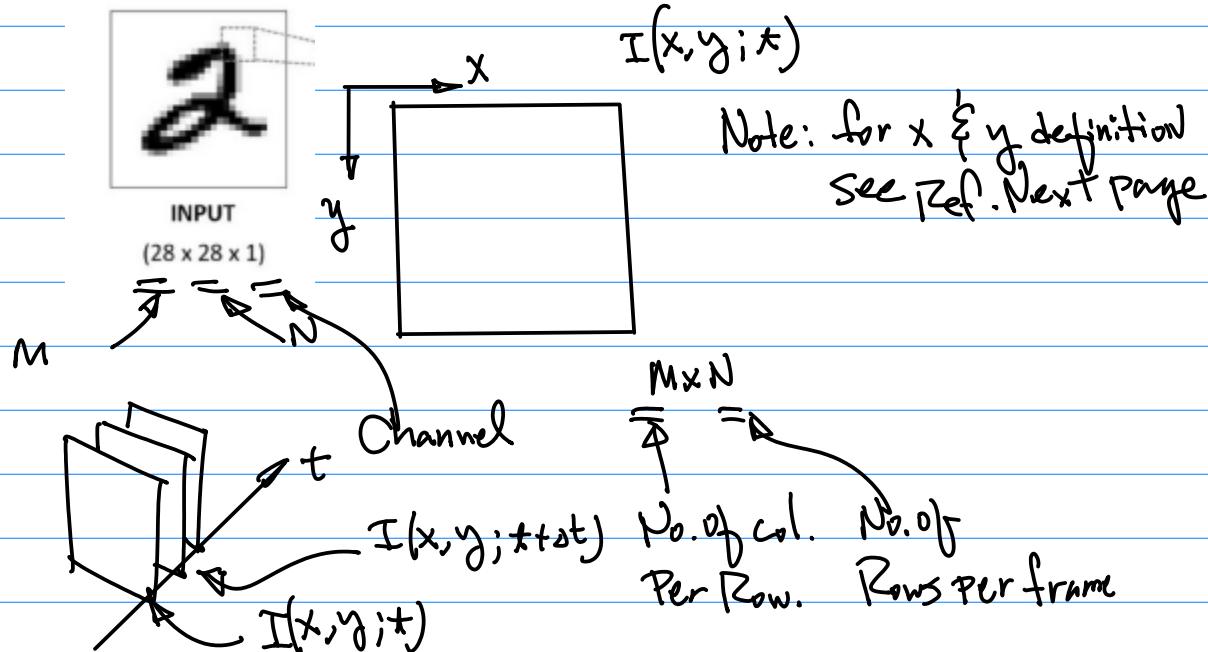
Feature Extraction
To produce Feature Vectors.

Feed Forward
Neural Networks.

Decision
Making

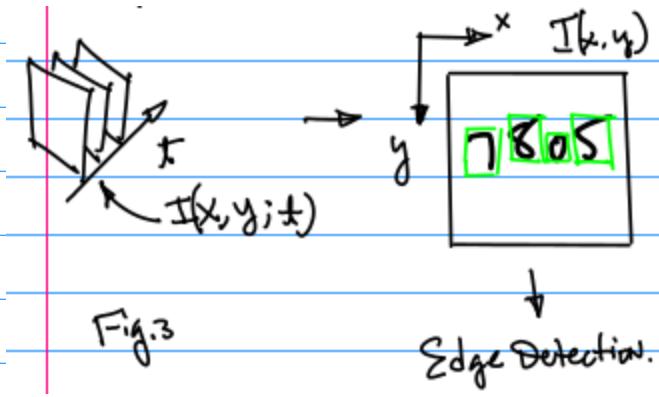
$$\vec{u} = (u_1, u_2, \dots, u_N)$$

2. Image Definition



Ref: on the github.

[2023S-101-Notes-cmpe258-2023-03-16.pdf](#)



Comparison to Web Cam.

Web Resolution: 1920x1080

$$\begin{aligned} & \downarrow \\ & \sim 2^{11} = 2 \cdot 2^9 \\ & = 2K \\ & \sim 2^{11} \\ \\ & 2^{22} = 2 \cdot 2^{20} \\ & = 4 \text{ Meg} \end{aligned}$$

$\approx 4 \text{ million.}$

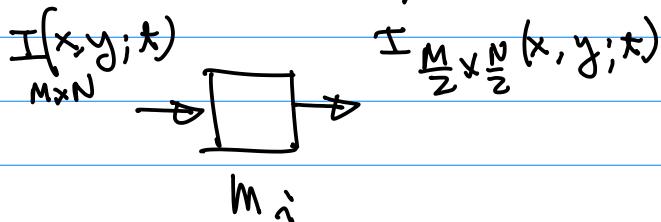
3. For the Convolutional Layer, we denote it as C_i , where $i=1, 2, \dots, N$

for Max Pooling Layer, we denote it as M_j , where $j=1, 2, \dots, K$

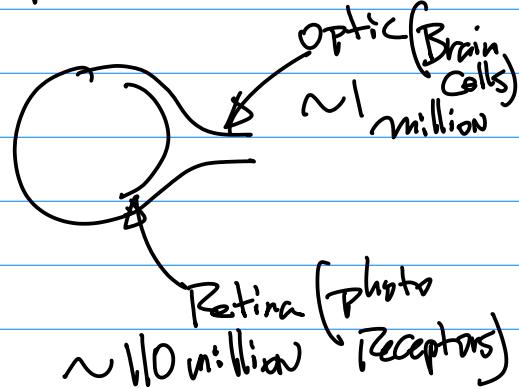
So, for the example, we have

$$C_1 M_1 \rightarrow C_2 M_2$$

\nearrow
Maxpooling for Resolution Reduction/Feature Reduction.



Note: Biologic Inspiration,
Human Visual Perception System,
Retina, $\approx 1/0$ million
photo Receptors



4. At the 3rd Segment (Blocks) of the Architecture, we denote Feed Forward Neural Networks as F_M

$$\uparrow$$

 F_M
No. of Neurons

for example, F_{10} (10 Nodes) for the output layer.

August 29 (Tue)

Note: 1^o CANVAS is up.2^o Homework Assignment

} a. Honesty Pledge to Be
Signed/Signed Copy
has to be uploaded
to CANVAS.
b.

Software Tools
Installation. (0 pt)

(1) OpenCV. By Friday

Next Tuesday. Bring Your
Laptop w/ OpenCV installed.

↓
Please use Smartphone
to take a photo, And upload
the photo to your laptop to
display.

Note: Sample code (Python)

was posted on the github.

(2) Anaconda Installed on
your Laptop.Note: Readme for Anaconda
installation was posted on
the github.

(3) Create ChatGPT Account.

Python Interface to ChatGPT
API (3.5 Version) is to
be utilized in your Team Project.3^o Form Team for the
Semester Long Project.Example: Consider 2D Convolution
Technique.Ref: [github/valihi/OpenCV](https://github.com/valihi/OpenCV)

2022

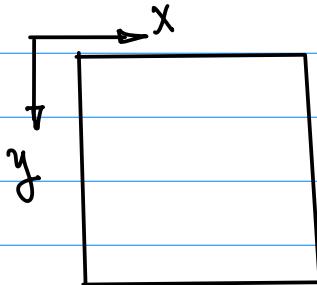
Note 1:

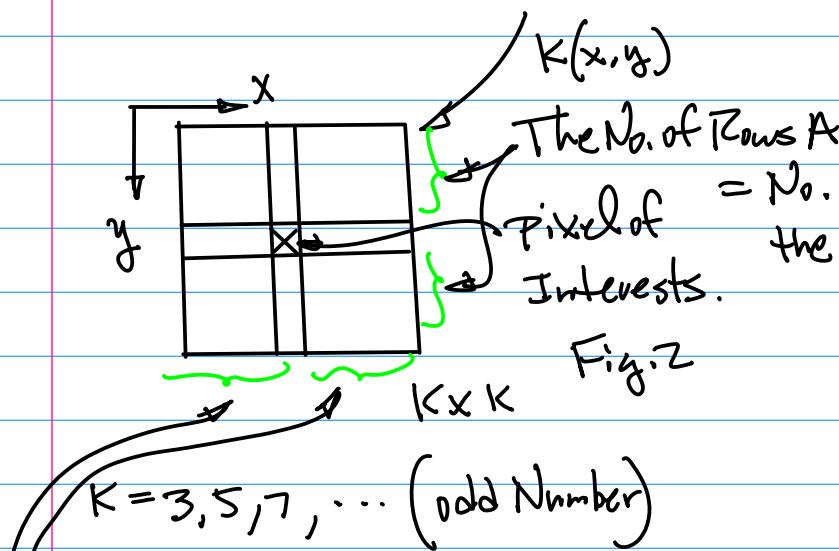
$$c(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} a(k_1, k_2) b(n_1 - k_1, n_2 - k_2)$$

Image Kernel

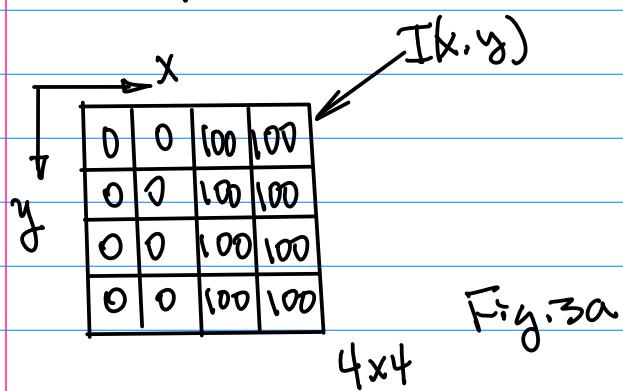
$n_1 = x, n_2 = y$

... (1)

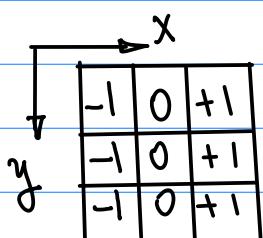
Summation
Index: k_1, k_2 are
for x and y .Note 2: $a(x, y)$ or $a(k_1, k_2)$ as
an Image. $I(x, y)$ $b(x, y)$: a Kernel for 2D
Convolution, $b(k_1, k_2)$ Note 3: Kernel $b(x, y)$ can be rewritten
 $K(x, y)$ Size of A kernel is denoted as
 $K \times K$



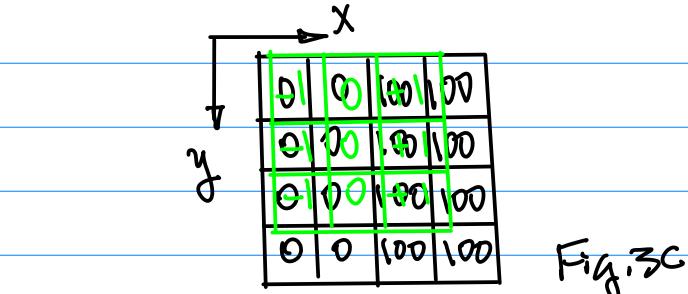
The No. of Col. Right to the
P.O.I. = The No. of Col. Left
to the P.O.I.



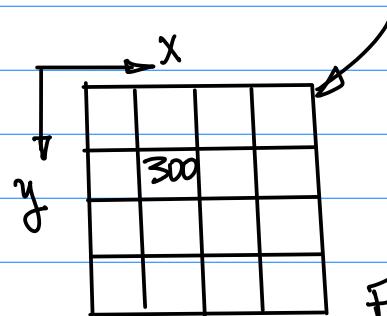
$$8 \text{ bit Grayscale: } 2^8 - 1 = 255$$



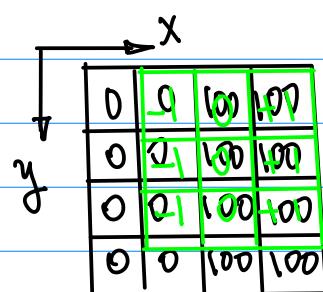
Step 1. Take the Kernel, and place
it at the initial position.



$$\begin{aligned} & -1 \times 0 + 0 \times 0 + 1 \times 100 + \\ & -1 \times 0 + 0 \times 0 + 1 \times 100 + \\ & -1 \times 0 + 0 \times 0 + 1 \times 100 \\ & = 300 \end{aligned}$$



Step 2. Shift $K(x,y)$ to the Right By
1 pixel (on the Same Row).



$$\begin{aligned} & -1 \times 0 + 0 \times 100 + 1 \times 100 + \\ & -1 \times 0 + 0 \times 100 + 1 \times 100 + \\ & -1 \times 0 + 0 \times 100 + 1 \times 100 = 300 \end{aligned}$$

$$I(x,y) * K(x,y)$$

August 31 (Th).

Note: 1° Honesty Pledge on CANVAS,
Signed pdf due this Friday.
(ON CANVAS). Homework ON CANVAS

2° Will post Anaconda Installation
& OpenCV Installation, display
an image.

(1) Screen Capture of the activated
CONDA Environment, with personal
identifier;

(2) Screen Capture of the OpenCV
Display with P.I.D.

Sample Code for the display to
Be Re-posted on github, 2023F

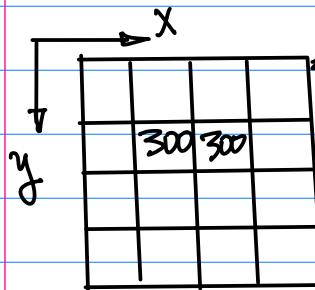


Fig. 3 F

Step 3.

$$I(x,y) * K(x,y)$$

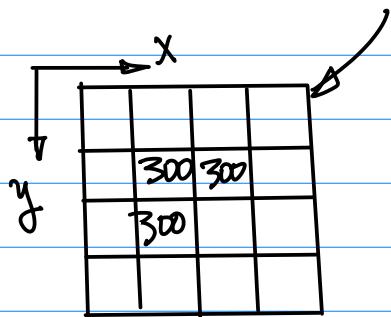


Fig. 3 G

Step 3.

$$I(x,y) * K(x,y)$$

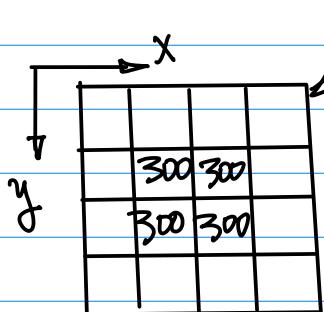


Fig. 3 H

Note: 2D Convolution (Continuous Case)

given Image $f(x,y)$
a Kernel $K(x,y)$

$$\iint f(u,v) K(x-u, y-v) du dv \dots [2]$$

Example: Theoretical Aspects
for 2D Convolution.

Ref: 1° PPT on the github.



1-2020S-#2019S-23-
2DConvolution-2019-2-4.
pdf

2° 2023S 1D-note-...

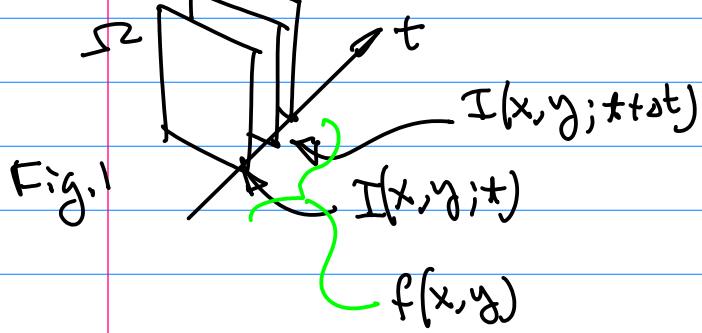
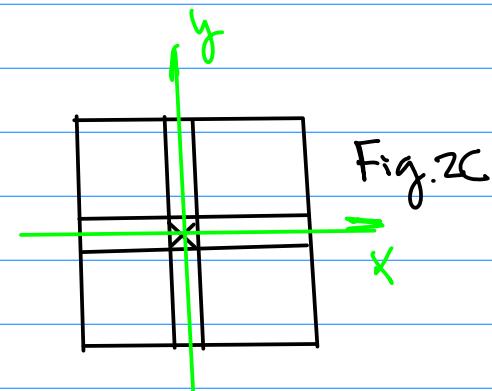
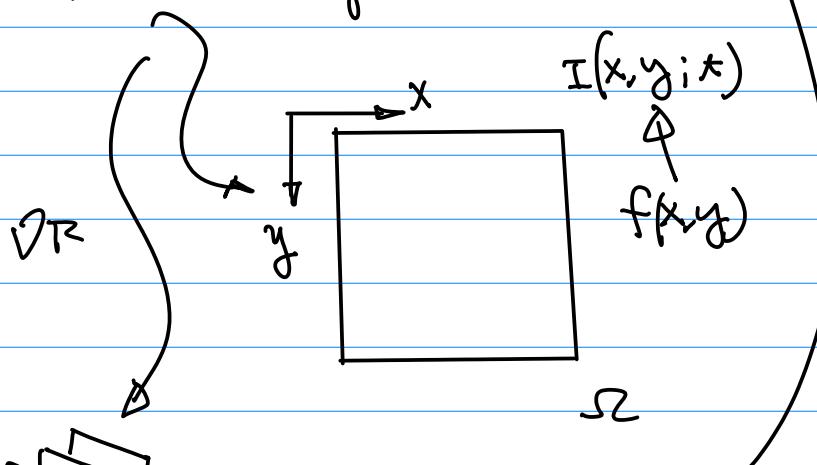
2023-03-21.pdf

$$\sum_{\Sigma} f(u,v) h(x-u, y-v) du dv \quad \text{... (1)}$$

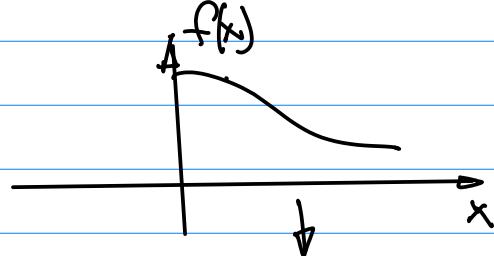
~~\sum~~ = Image Kernel

Note 1. Σ : Image Planes

→ (2) Flip the kernel w.r.t. X-axis.
(since $k(-x, y)$ changed to $k(-x, -y)$)



Note: for 1D Case



Note 2: Kernel

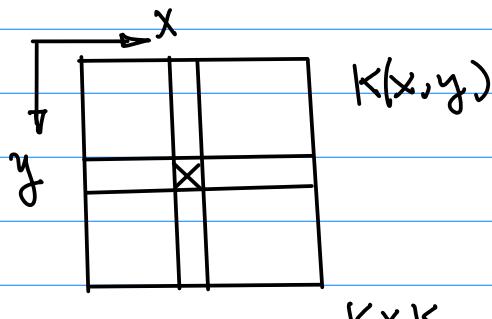
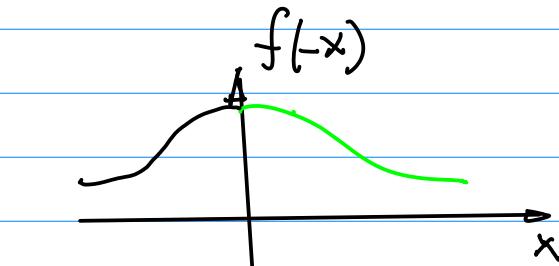


Fig. 2a

↓ (1)

$K(-x, y)$

Flip the Kernel
w.r.t y-axis



Next, make $k(-x, -y)$ to
 $k(x-x, y-y)$

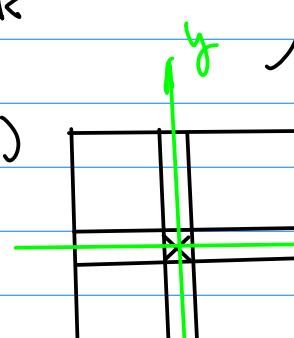
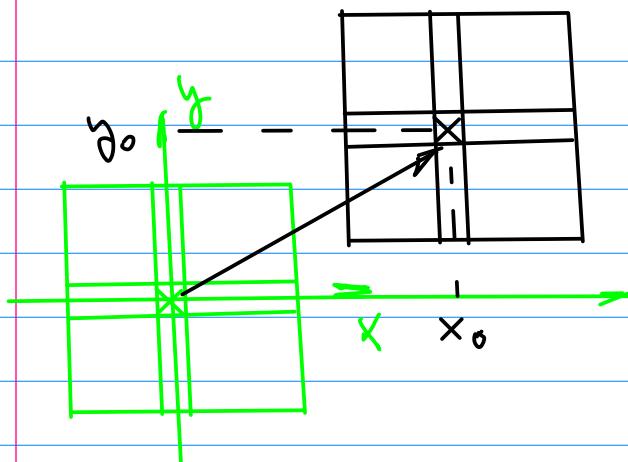


Fig. 2b



Note: $\frac{k-1}{2}$ for the top Rows
 $\frac{k-1}{2}$ for the Bottom Rows
 Lost
 $2 \times \frac{k-1}{2} = k-1$

Similarly for the Col.s.

Fig.2d

Now, for Discrete Image (OR
Digital
Feature plane)

Replace $\int \int$ by $\sum \sum$

$$c(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} a(k_1, k_2) b(n_1 - k_1, n_2 - k_2)$$

Image Kernel ... (z)

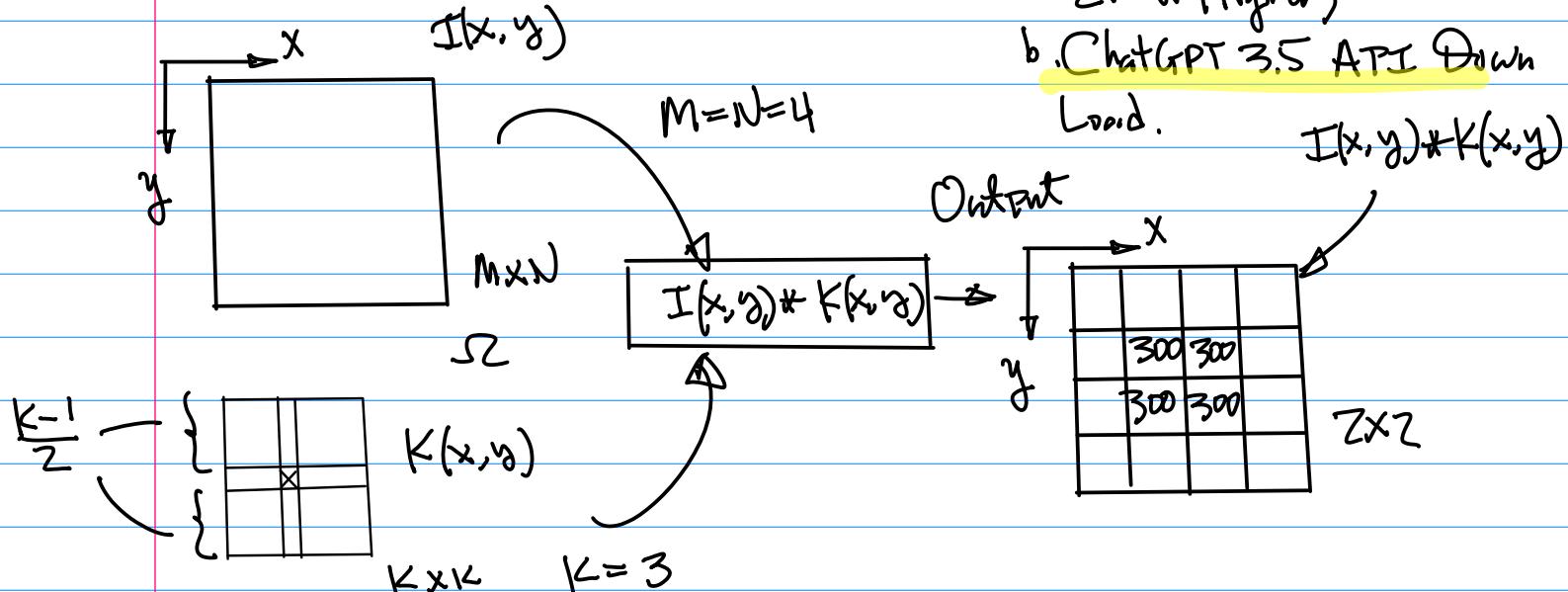
Dimension Change Due to the Convolution.

Therefore,
 Input Dimension $M \times N$ \rightarrow Output After Convolution
 $m - (k-1) \times N - (k-1)$
 ... (3)

Sept. 5 (Tue)

Note: 1^o CANVAS OpenCV, Anaconda Installation;
 2^o Homework Coming On Thursday.

- a. Installation of T.F. Version 2.0 or Higher;
- b. ChatGPT 3.5 API Down Load.



Example: Continuation on
Convolutional Layers and
Architecture for Handwritten
Digits Recognition.
Based on Eq(3), PP.9.

$M \times N$ Image $\rightarrow M - (K-1) \times N - (K-1)$ System.

$K \times K$
Convolution

From MNIST Architecture,
the Input Image $I(x, y)$ with
 $M \times N$ ($M = N = 28$) + the
Output feature layers Resolution
 24×24 . $\rightarrow K = (28-24) + 1$
 $\underbrace{}_{\text{Resolution difference.}}$

Now, the total No. of feature
layers = n

Since one Convolution (e.g.
using one $K \times K$ Kernel
to perform Convolution)
produces 1 feature layer,

Therefore
 n Convolutional feature
Layers is the
result of n Convolutions,
e.g. n different
Kernels.

Note: Multiple feature layers
are due to the inspiration
of Human Visual Perception
System, e.g. "Early" Vision

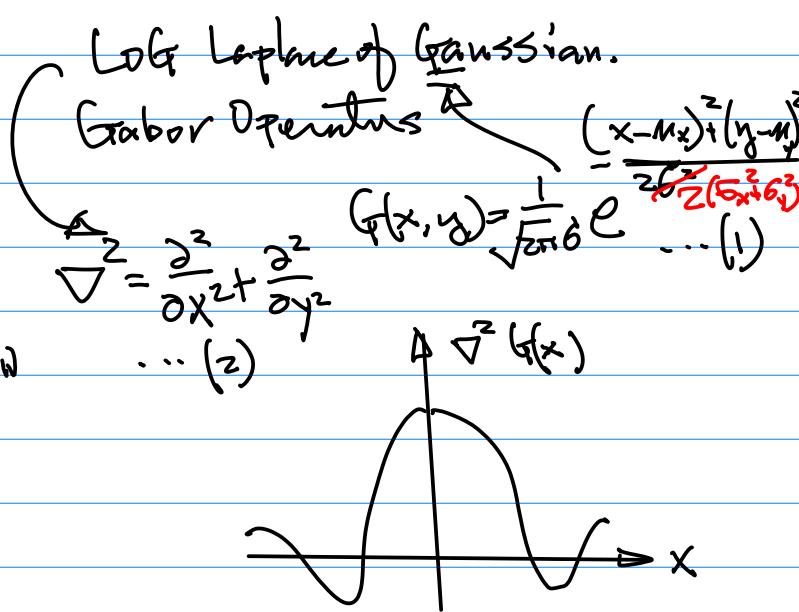


Fig.1.

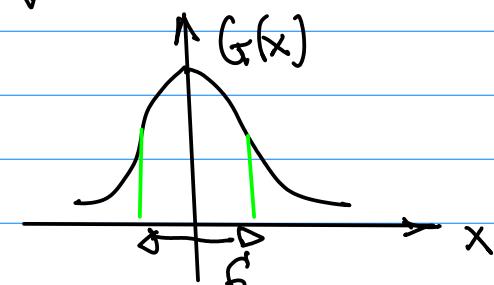


Fig.2

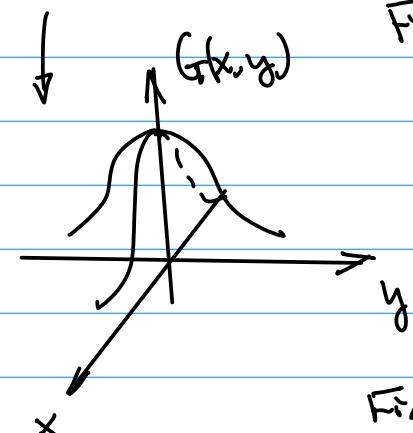
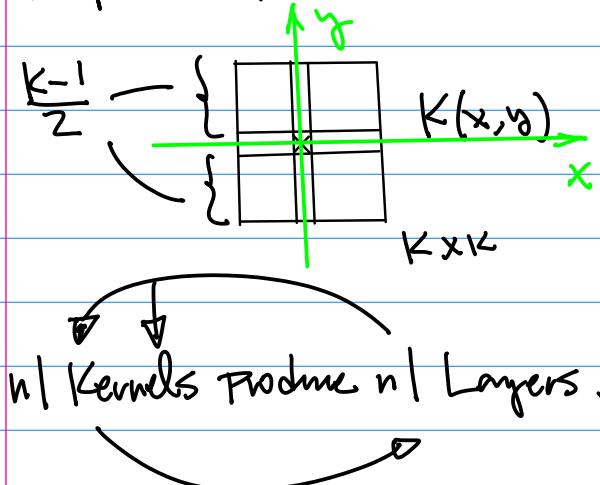


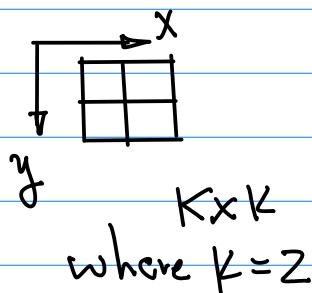
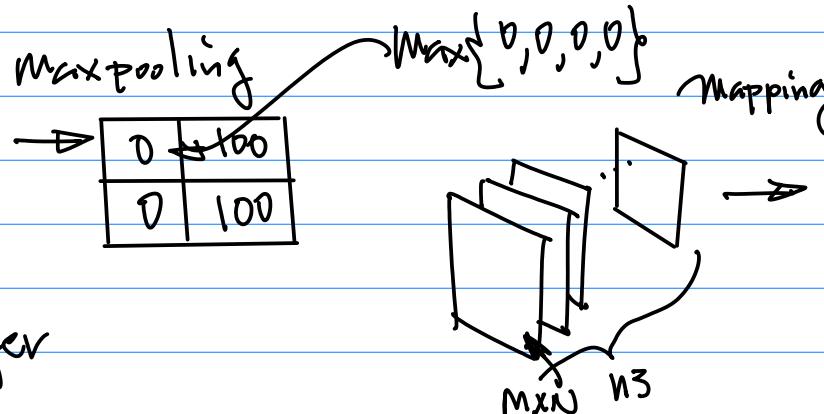
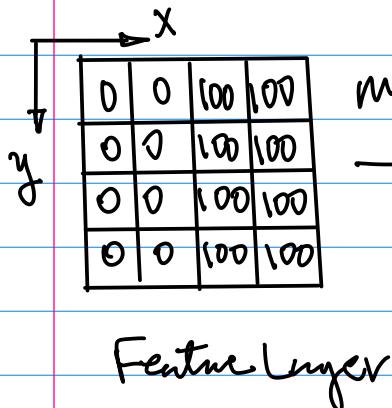
Fig.3

Take $G(x, y)$ in Fig. 3,
Map it to $K \times K$.



Now, consider a technique
for feature Reduction,

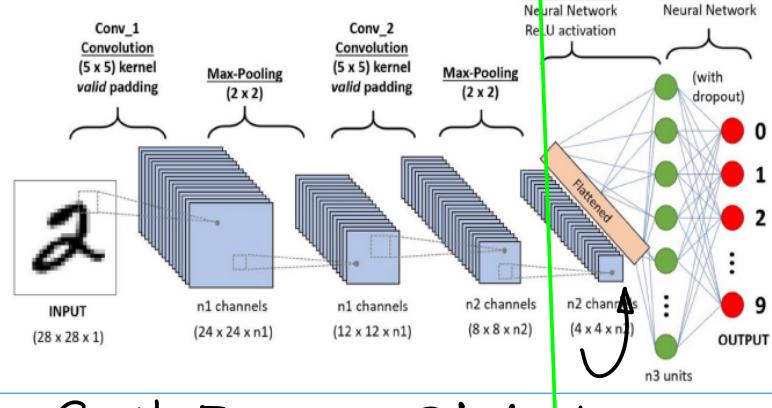
Given an image $I(x, y)$



Now, consider the Architecture of MNIST

Illustration of A CNN for Digits Recognition

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



Consider Re-arrange 2D feature layers
into 1D
Each
Neurons.



Example: Sample code
OpenCV for Displaying
an image

cv2.imread(imageName, cv2.IMREAD_COLOR)
cv2.imshow(window_name, src)

Sept. 7 (Th).

Note: 1^o About OpenCV
Reference/Sample
Code.

a) On github. for Video Capture

[opencv / deep-learning-2022s / 2022S-109c-v2-localization-contours.py](#)

hualili Add files via upload

to get A Single

Frame.

b) Sample for Single frame, e.g.
.jpg, .png etc Image Input.

2022S-104d ~ Python Code

c) Script for Conda Environment

On github :

2022S-104b ~

" - 104c ~

d) Readme for Creating Conda
Environment.

Note 2. Homework Due A week

from today (Sept. 17, Sun)

Requirements :

a) Installation of T.F. Version 2.0
or higher

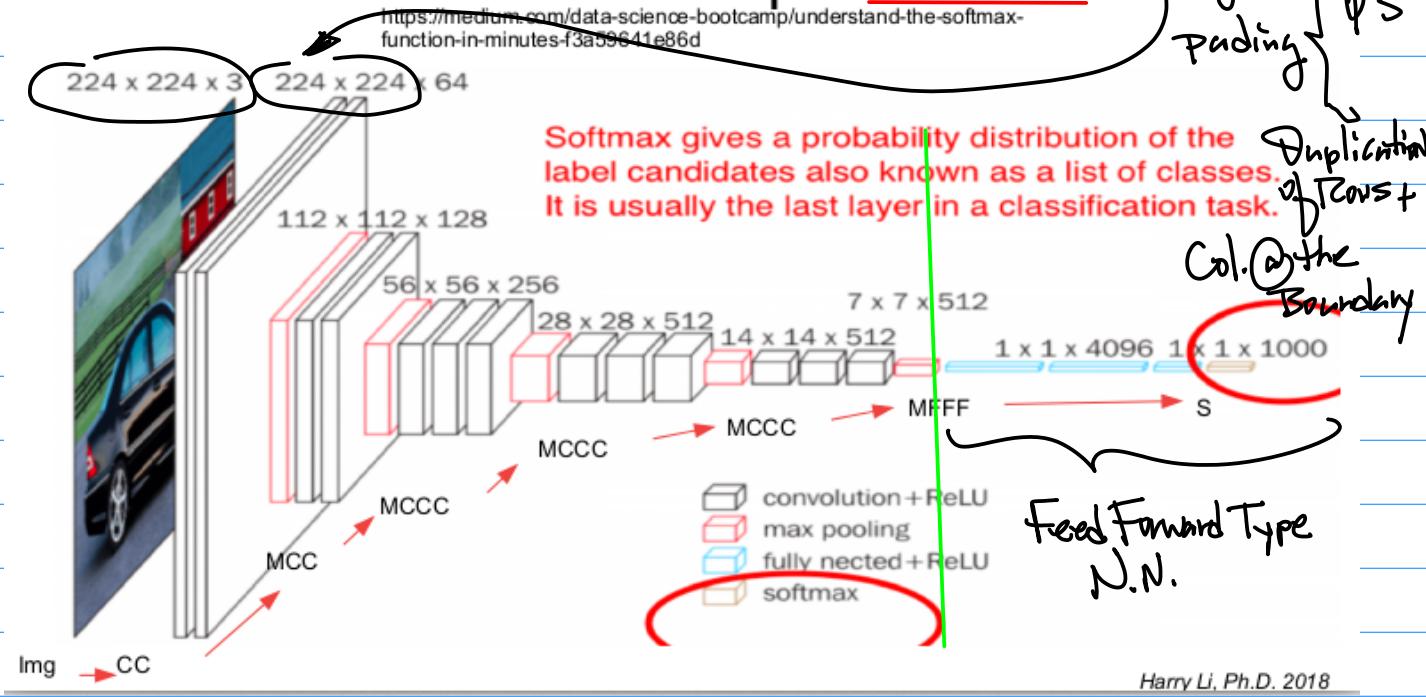
b) Screen Capture that Shows
T.F. installed Successfully.
(with Personal Identifier)

Submission on CANVAS.

Example: Convolutional Layer,
Architecture Analysis.

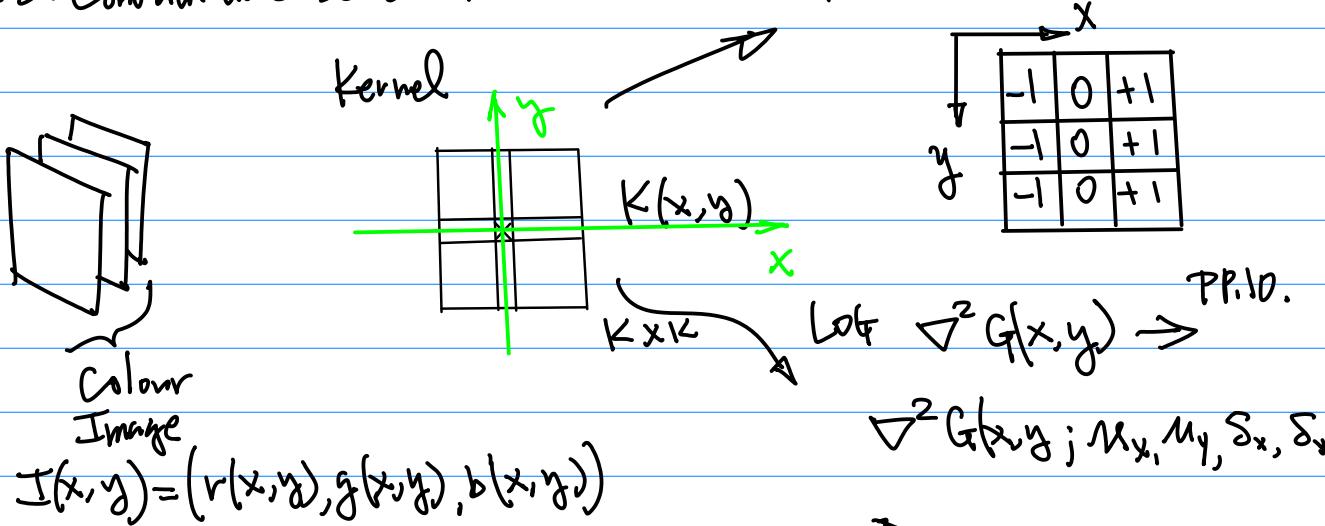
Note 1. Architecture; Note 2:
Padding v.s. No
Padding.
"ψ's
Duplication
of Prev +
Col. @ the
Boundary

Architecture Example VGG16



Note 3: Convolution Discussion.

One Example



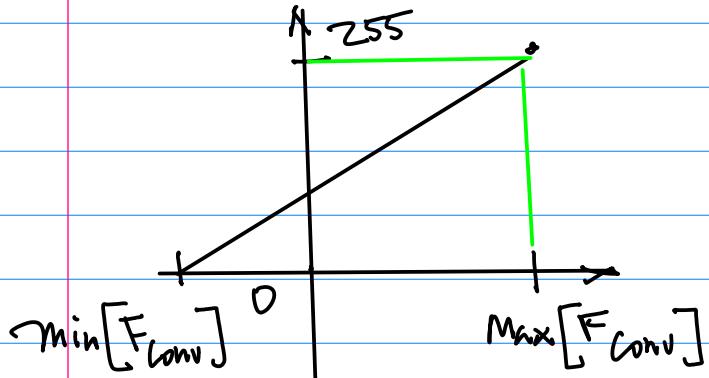
To Perform Convolution. $\rightarrow K(x, y)$

First, kernel is one channel in this case;

Then, convolutions. $r(x, y) * K(x, y)$, $g(x, y) * K(x, y)$, $b(x, y) * K(x, y)$

$$\rightarrow \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) G(x, y; M_x, M_y, S_x, S_y)$$

$$F_{\text{Conv}}(x, y) = \frac{1}{3} [r(x, y) * k(x, y) + g(x, y) * k(x, y) + b(x, y) * k(x, y)]$$



Consider A Design of S.I.D
Recognition System Using T.F &
MNIST CNN as a processing
Engine.

Requirements:

1° Live CAM Input,

JPG or JPEGP

$(1280 \times 720 \text{ (MxN)})$

Col.

Row.

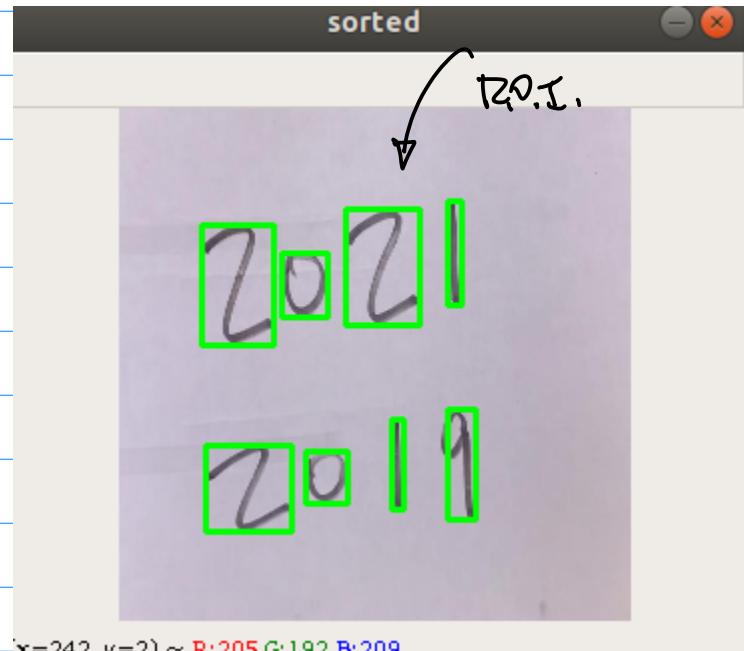
1920×1080
 $(M \times N)$

Col.

Row.

2° Printer Paper (Blank/White)
Black Mark to Write
4 Digits

3° Localize R.O.I. On Each
Digit



Region of Interest.

Sept. 12 (Tue).

Note 1. For Homework 1 Extended to
the Saturday, Requires the
Submission of Python Code.

Note 2. Team Project.

Ref:

→ 2023F-103-project-team-2023-9-12.pdf

ON CANVAS as well.

Due on Nov. 27 (Monday) 11:59 PM

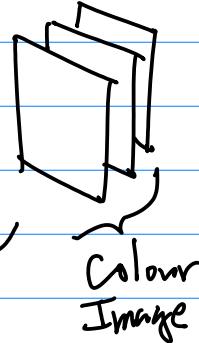
Example: Suppose a color image
 $I(x, y)$ is given below, and
a Convolution Kernel $K(x, y)$

is given as follows

$$\begin{array}{c} \downarrow y \\ \frac{1}{5} \end{array} \quad \begin{array}{c} \rightarrow x \\ \hline -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{array}$$

3×3 .

A Color Image $I(x, y)$



Find Convolution Result.

$$\begin{array}{c} \downarrow y \\ \frac{1}{5} \end{array} \quad \begin{array}{c} \rightarrow x \\ \hline 0 & 0 & 0 & 255 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}$$

$I_r(x, y)$

$$\begin{array}{c} \downarrow y \\ \frac{1}{5} \end{array} \quad \begin{array}{c} \rightarrow x \\ \hline 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \\ 0 & 0 & 5 & 100 \\ 0 & 0 & 2 & 0 \end{array}$$

$I_g(x, y)$

$$\begin{array}{c} \downarrow y \\ \frac{1}{5} \end{array} \quad \begin{array}{c} \rightarrow x \\ \hline 0 & 0 & 255 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 255 & 0 \end{array}$$

$I_b(x, y)$

$$\begin{array}{c} \downarrow y \\ \frac{1}{5} \end{array} \quad \begin{array}{c} \rightarrow x \\ \hline -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{array}$$

$$\frac{1}{5} \quad \begin{array}{c} \rightarrow x \\ \hline -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{array}$$

$$\frac{1}{5} \quad \begin{array}{c} \rightarrow x \\ \hline -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{array}$$

$$\begin{array}{c} 0 & -255 \\ 0 & 1 \end{array}$$

$$\begin{array}{c} -95 & 5 \\ 2 & -98 \end{array}$$

$$\begin{array}{c} -255 & 255 \\ 255 & -255 \end{array}$$

Note: In the future, we have need to reduce the Number of feature layers

By introducing Newer Kernel(s) for Convolution.

Homework: Due 1 week from Today.
(Sept. 19th).

1^o Installation of T.F., Screen
Capture the installation Result.
(W/Personal ID).

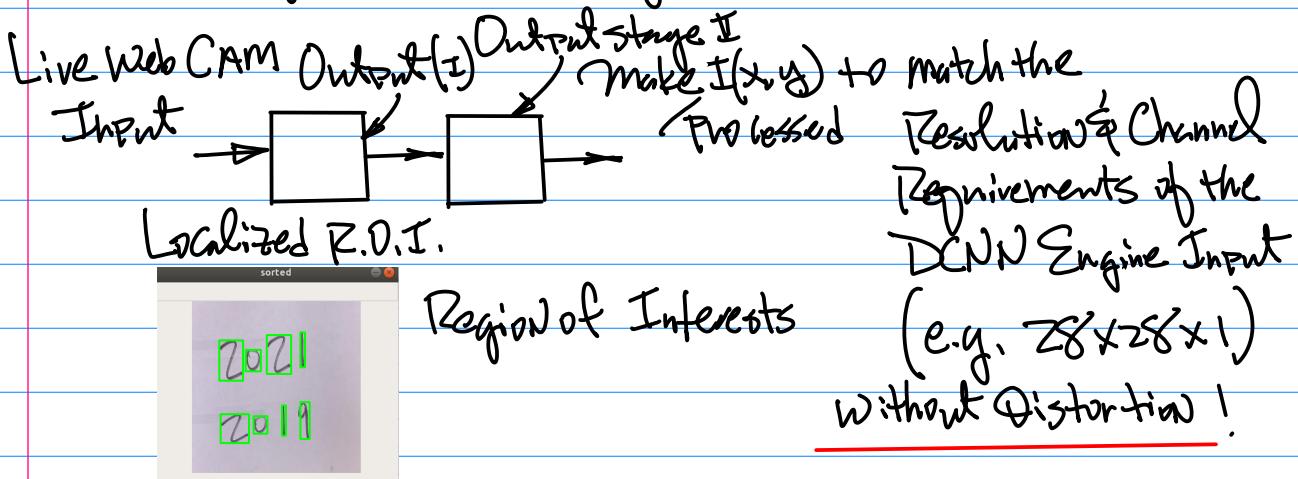
2^o OpenCV Code to Handle

- a. Live CAM Input Video,
Web
and Display it;
- b. file input, MP4G4 format
Video and display it.

Note: Please use A4 printer
paper, Write with a Black
marker, 4 Digits of your
SID.

Submission: the code &
Video Clips.

Preprocessing for Hand Written
SID Recognition System Design.



Ref:

Homework: The 1 week from today.
Use your OpenCV for the Sept 21st.
following Preprocessing functions.

1° Convert the color Image $I(x,y)$
to a gray Scale Image. $I_g(x,y)$

2° Binarize the gray scale image
to Obtain a Binary Image $I_b(x,y)$.

3° Perform Canny Edge Detection
on $I_g(x,y)$, and display the
edge map.

4° Perform Gaussian Blur on
the Gray Scale Image.

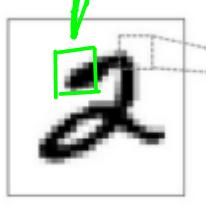
Submission:

5° Screen Capture of 1° to 4°.
with Personal Identifier.

Example: Continuation of project
Discussion. Preprocessing.

The First Objective: To get all
the Bounding Boxes

Color Video \rightarrow Gray Scale
Image.



INPUT
(28 x 28 x 1)

C. Filtering Operation, conducted
by Convolution w/ predefined
Kernel Coefficient.

→ B. To Be Continued.

Edge Detection Canny, Log,
Gabor

Note: Input Should be a
grayScale(8 bits), Output
Should be 8bit (1 Channel)

A.

→ Binarization. To Obtain A

Representation of A Digit.

Note: Binarization May lead to the
Loss of Useful information.

Example: Image Binarization.

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

Binarization is defined as

$$B(x, y) = \begin{cases} 255 & \text{if } I(x, y) \geq T \\ 0 & \text{D/W} \end{cases} \quad \dots (1)$$

$$A = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} B(x, y) \quad \dots (3)$$

$$\sum_{y=0}^{N-1} \sum_{x=0}^{M-1} (x - \bar{x})^m (y - \bar{y})^n B(x, y) \quad \dots (4)$$

Note: T is set for the entire image for now.

Example from PP. 18, Lecture

Notes on GitHub, 2023S.

Note: Treat the top left corner

as (1,1) for the Binarized

Image Descriptors

Calculation (to Void

Skewed Result).

Most Important Descriptors
are those Built Based on
Moments.

Where

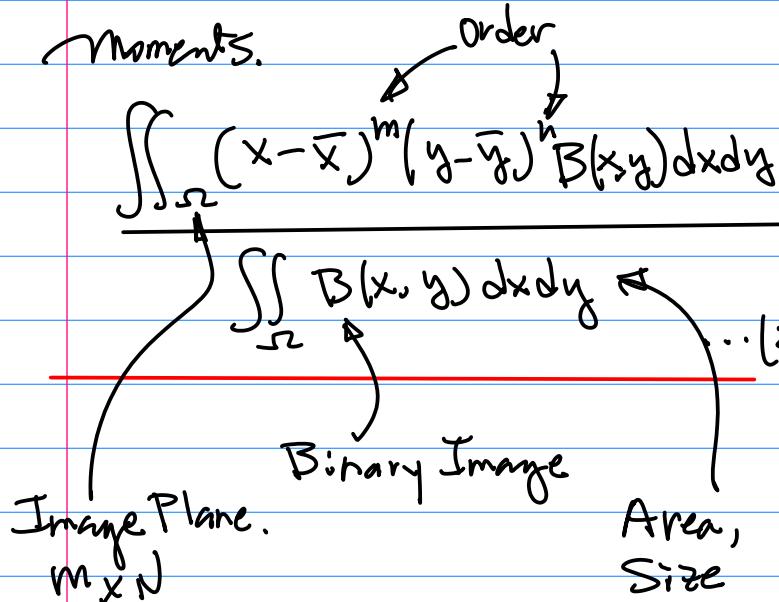
$$\bar{x} = \frac{\sum_{y=0}^{N-1} \sum_{x=0}^{M-1} x B(x, y)}{A} \quad \dots (4-a)$$

$$\bar{y} = \frac{\sum_{y=0}^{N-1} \sum_{x=0}^{M-1} y B(x, y)}{A} \quad \dots (4-b)$$

Sept 19 (Tue).

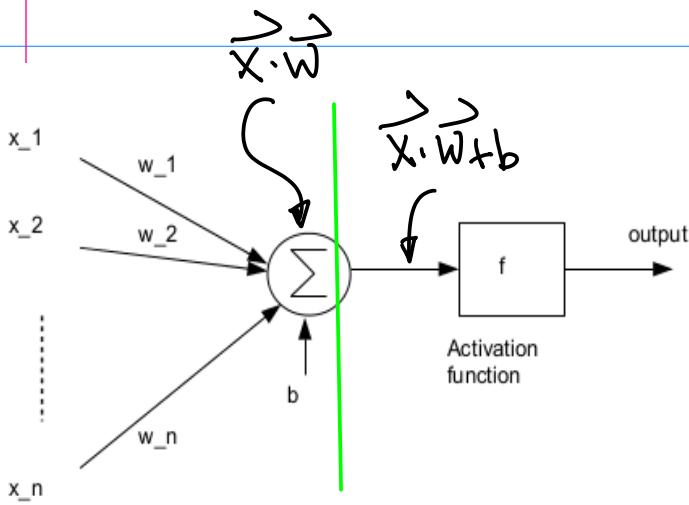
Example: Consider the Architecture
of MNIST CNN.

Roadmap:



MNIST \rightarrow Yolo. \rightarrow Yolact
CNN with
Preprocessing. To generate
Bounding Boxes
Semantic Segmentation
Creation
R.D.I.

Ref: On the github, 2022-103C ~



$$\sum_{i=1}^n x_i w_i \text{ or } \sum_{i=1}^n w_i x_i$$

$$= \vec{x} \cdot \vec{w} \quad \dots (3)$$

Together with b (Bias), we have

$$\sum_{i=1}^n w_i x_i + b = \vec{x} \cdot \vec{w} + b \quad \dots (4)$$

Note 1. Feature vector, or Input, Excitation

$$\vec{x} = (x_1, x_2, \dots, x_n) \quad \dots (1)$$

$$\vec{w} = (w_1, w_2, \dots, w_n) \quad \dots (2)$$

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

where $i = 1, 2, \dots, n$;

b : Bias, e.g., offset

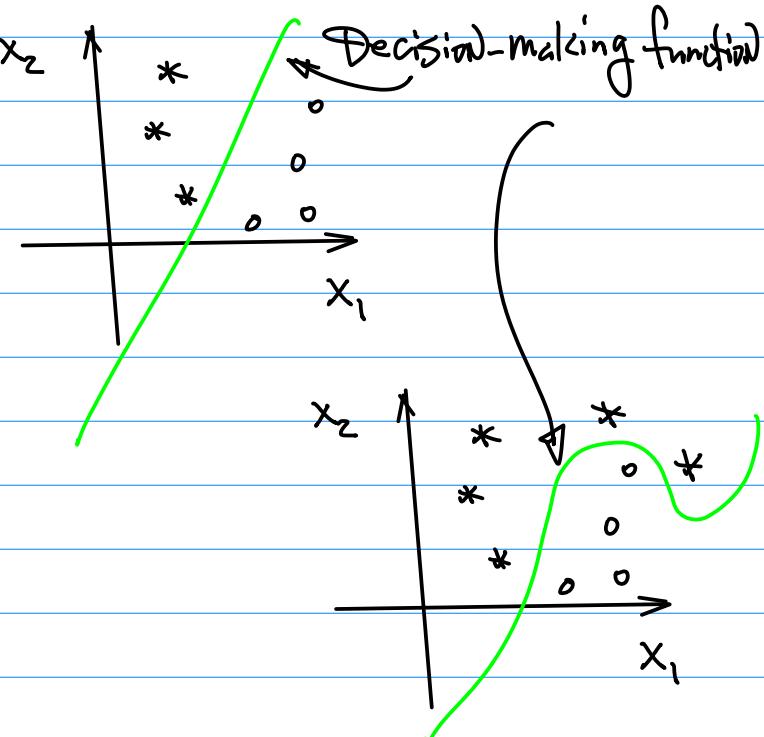
Note 2. $x_i w_i \rightarrow$ the Neuron
input 1 to

$x_2 w_2$: input 2 to
the Neuron.

:

$x_i w_i$: .. i ..

$$x_1 w_1 + x_2 w_2 + \dots + x_i w_i + \dots + x_n w_n$$

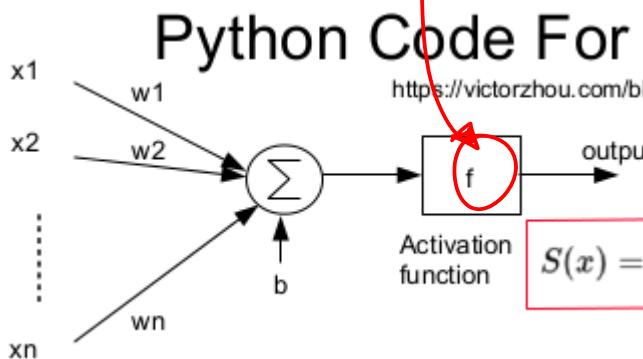


Note 3. Activation Function.

$$f\left(\sum_{i=1}^n w_i x_i + b\right)$$

... (5)

Note: RELU



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

$$y = f\left(\sum_{i=1}^n w_i x_i + b\right)$$

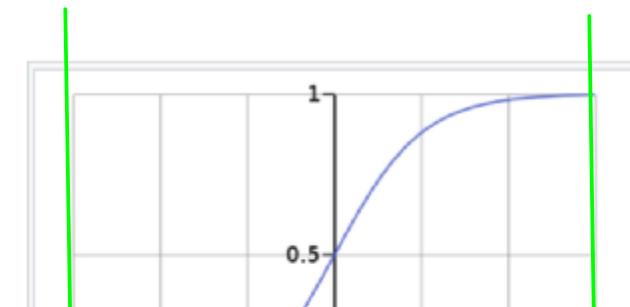
Output of the Neuron.

Note 4. Sigmoid function
<https://victorzhou.com/blog/intro-to-neural-networks/>

$\sum w_i x_i + b$

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

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

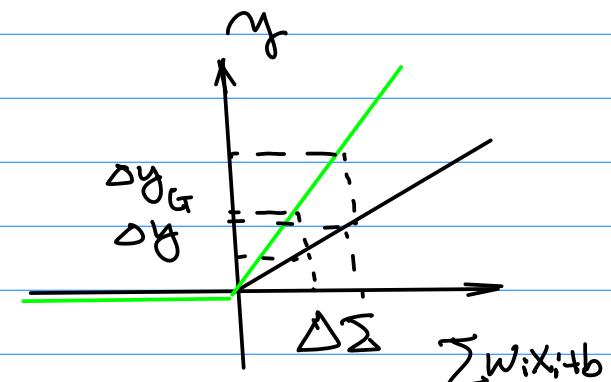
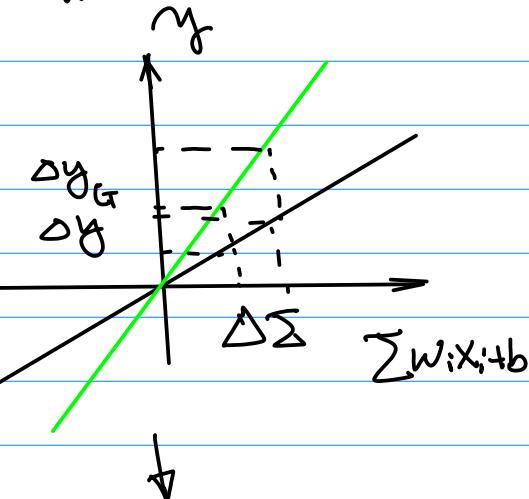


https://en.wikipedia.org/wiki/Sigmoid_function

I

II

III



$$\sum_{i=1}^n w_i x_i + b$$

CMPE258

F2023

21

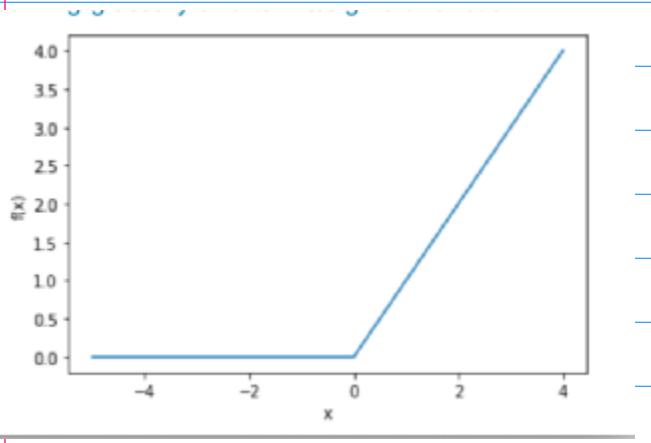
Sept. 21 (Th),

Continuation on the Formulation
of Feedforward NN.

Ref: github.



Fig. 1.



$$f(x) = \begin{cases} x & x \geq 0 \\ 0 & \text{o/w} \end{cases} \dots (1)$$

Note: Comparison to Sigmoid
in terms of its Computation

$$S(x) = \frac{1}{1 + e^{-x}} \dots (2)$$

from Taylor Expansion.

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

Sample plain python code on github.
Example.

Note: 1. define activation function

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

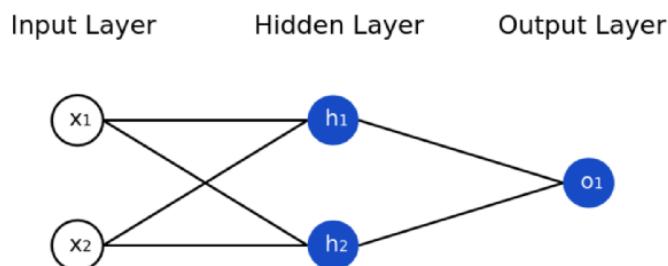
2. define a single neuron

```
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)
```

$$\text{Note: } y = f\left(\sum_{i=1}^n w_i x_i + b\right)$$

Can be applied to the Neurons
in the following illustration



A sigmoid
mathemat
characteris

Note 1. If output of a Neuron. { y_{pred} Compare the two, to
 y_{true} , evaluate the

Define Loss Function

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

Performance of

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

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

$$\downarrow$$

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

$$\downarrow$$

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

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

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

$$\begin{aligned} & \text{Note 2. } y_{pred} \neq y_{true} ? \\ & \quad \begin{cases} y_{pred} - y_{true} = 0 & \text{the same} \\ y_{pred} - y_{true} > 0 & y_{pred} > y_{true} \\ y_{pred} - y_{true} < 0 & \text{o/w} \end{cases} \end{aligned}$$

Note 3. Computation for a large dataset

$$\tilde{y}_1 - y_1, \tilde{y}_2 - y_2, \tilde{y}_3 - y_3, \dots$$

$$\tilde{y}_n - y_n.$$

Where \tilde{y}_i : Experiment Output.
 Actual Output.

y_i ground Truth.

$$\sum_{i=1}^n \tilde{y}_i - y_i$$

↓ To Avoid Cancellation
 of errors. Let's
 square them.

$$\sum_{i=1}^n (\tilde{y}_i - y_i)^2 \dots (4)$$

$$\overline{\text{MSE}} = \frac{1}{n} \sum_{i=1}^n (\tilde{y}_i - y_i)^2 \dots (4-b)$$

Average of the Accumulative
 Computation for the entire Data
 Set.

M.S.E (mean square error).

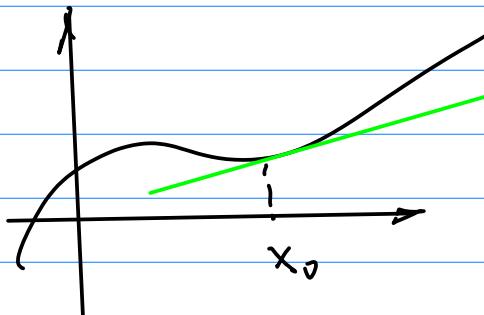
Example: Gradient Descent

Given $f(x_1, x_2, \dots, x_n)$, such as
 the objective function in
 Eqn.(4-b).

To Evaluate its performance
 for training purpose, we use
 derivative.

Consider One dimensional
Case first.

$$\frac{df(x)}{dx} = \lim_{\Delta x \rightarrow 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \dots (5)$$



if Eqn(5) > 0

$$f(x+\Delta x) > f(x)$$

if Eqn(5) < 0, then

$$f(x+\Delta x) < f(x).$$

The goal is to make the objective function, e.g., loss function reduced to its minimum through the training.

For $f(x_1, x_2, \dots, x_n)$, we have partial derivatives.

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

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

⋮

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

Now, to put each partial derivative together to get a whole view of the loss (e.g. Objective function), we define a gradient.

$$\left[\begin{array}{c} \frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_1} \\ \frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_n} \end{array} \right]$$

... (b)

Sept. 26 (Tue).

Note 1. New Homework Posted
on the CANVAS.

k+1

$$w_1 = w_1^k + (-\eta) \frac{\partial f}{\partial w_1} \quad (1-c)$$

$$w_2 = w_2^k + (-\eta) \frac{\partial f}{\partial w_2}$$

⋮

$$w_i = w_i^k + (-\eta) \frac{\partial f}{\partial w_i}$$

⋮

$$w_n = w_n^k + (-\eta) \frac{\partial f}{\partial w_n}$$

Example: On the gradient descent
Technique.

Ref:

2022S-105c-#20-2021S-4gradie

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

Note 1. Weights, $\vec{w} = (w_1, w_2, \dots, w_n)$
where \vec{w} matches to
the feature vector $\vec{x} =$
 (x_1, x_2, \dots, x_n)

Superscript "k+1", index for
the training at Step k+1

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial w_n} \end{pmatrix} \quad \text{... (1)}$$

" " transpose
→

$$\nabla f^t = \left(\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \dots, \frac{\partial f}{\partial w_n} \right) \quad \text{... (1b)}$$

Now, verify the equation.

From Eqn (b).

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

Consider 1D Case

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

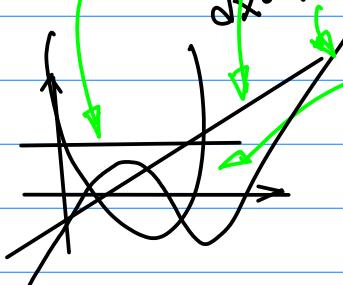


Fig. 1

The multidimensional Taylor
Expansion can be realized based
on the Similar Principle.

Simplify it, so just use the constant and the linear term \rightarrow Linear Term.

$$y = ax + b \rightarrow y' = a x + \underbrace{b + c}_{b'}$$

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

Start from here

Note: Denote $\frac{\partial f}{\partial x_1}$ as $f_{x_1}(a, b)$

$\frac{\partial f}{\partial x_2}$ as $f_{x_2}(a, b)$

$$f(x_1, x_2) - f(a, b) \approx f_{x_1}(a, b) * (x_1 - a) + f_{x_2}(a, b) * (x_2 - b) \quad (8)$$

Or,

$$f(x_1, x_2) - f(a, b) = (x_1 - a, x_2 - b) \begin{pmatrix} f_{x_1}(a, b) \\ f_{x_2}(a, b) \end{pmatrix} \quad (9)$$

Hence,

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

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

The error becomes smaller

Each iteration as long as

$\Delta x_1, \Delta x_2$ defined by Eqn (5)

$$f(x_1, x_2) - f(a, b) = -(f_{x_1}^2 + f_{x_2}^2) < 0 \quad (13)$$

Now, Review the Python Sample Code

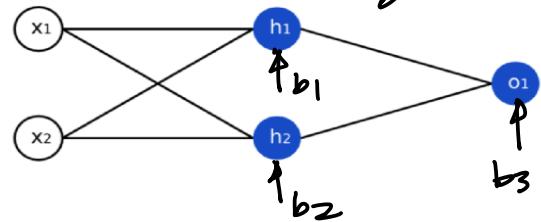
Ref



2022S-103c-#nn_sample_2022.py

Sample code for this NN.

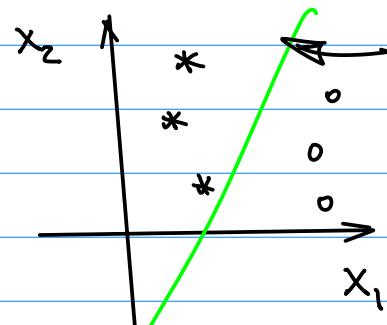
Input Layer Hidden Layer Output Layer



```

159 # -----
160 data = np.array([
161     [1, 2.5],      # person A
162     [1, 3],        # person A
163     [2.1, 3.4],   # Person A
164     [2.1, 1],     # person B
165     [3.3, 1],     # person B
166     [3, 2.3],     # person B
167     # HL 2020-9-7 Part E
168     # for the testing of adaptive learning
169     # (1) create a new program based on this o
170 ])

```



Sept. 28 (Th)

Note 1. LLM (Large Language Model) Based on ChatGPT.
Focus API Package,

Note: the 1st of the 3 PPT.

Deploy API with Python.

Ref: On the class
github.



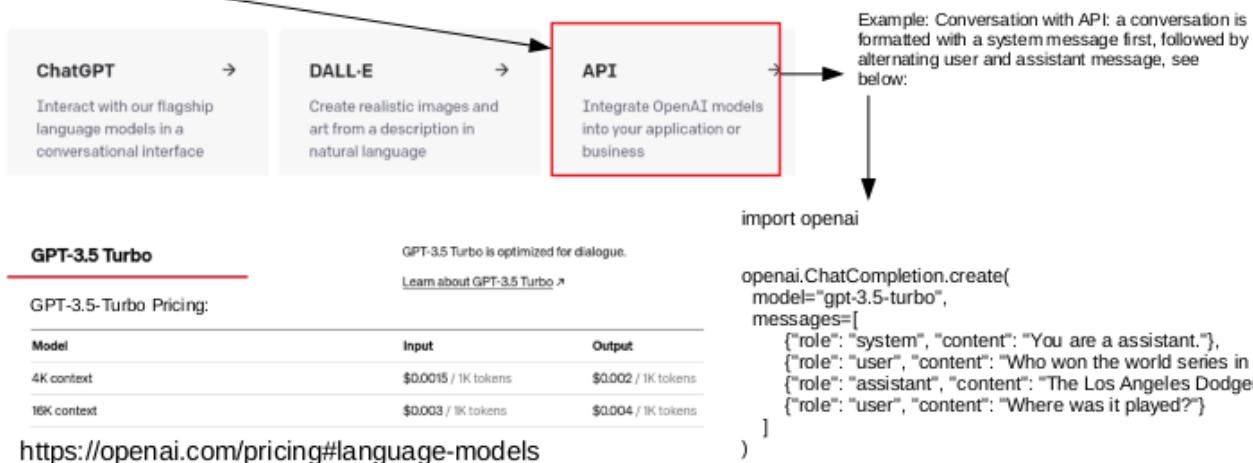
ChatGPT API from OpenAI

<https://community.openai.com/t/is-chat-gpt-provided-for-free/86249>

Step 1. Log in ChatGPT, then click on API



Step 2. Design your prompt to program your model



```
harry@harrys-gpu-laptop: ~/PycharmProjects/chatGPT
File Edit View Search Terminal Help
(base) harry@harrys-gpu-laptop:~/PycharmProjects/chatGPT$ tree -L 2
.
└── API_Key.json
    └── ChatHistory.json
        └── fine-tuning
            ├── fine-tuning.jsonl
            └── finetuning.py
            └── sample
                └── fine-tuning-10QA-2023-9-14
                    └── GPT_Fine_Tuning.zip
                    └── GPTWithHistorySaved.py

3 directories, 6 files
(base) harry@harrys-gpu-laptop:~/PycharmProjects/chatGPT$
```

Note 1. Be Sure to Save it on a Separate file from the ChatGPT(OpenAI) Page.

harry@harrys-gpu-laptop: ~/PycharmProjects/chatGPT

```

File Edit View Search Terminal Help
fine-tuning
  fine-tuning.jsonl
  finetuning.py
  sample
fine-tuning-10QA-2023-9-14
  GPT_Fine_Tuning.zip
GPTWithHistorySaved.py

3 directories, 6 files
(base) harry@harrys-gpu-laptop:~/PycharmProjects/chatGPT$ tree -L 3
.
└── API_Key.json
    ├── ChatHistory.json
    └── fine-tuning
        ├── fine-tuning.jsonl
        ├── finetuning.py
        └── sample
            └── train_data.jsonl
    └── fine-tuning-10QA-2023-9-14
        └── GPT_Fine_Tuning.zip
    GPTWithHistorySaved.py

3 directories, 7 files
(base) harry@harrys-gpu-laptop:~/PycharmProjects/chatGPT$ 
```

Notez: ".jsonl" file that user created by developer to provide training data for fine-tuning purpose

Note: Create ".jsonl" file to provide Training Data for fine-tuning.

harry@harrys-gpu-laptop: ~/PycharmProjects/chatGPT/fine-tuning

```

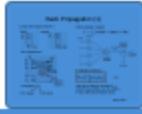
File Edit View Search Terminal Help
{
  "messages": [
    {
      "role": "system",
      "content": "CTI One Test: W100 is a smart walker to listen to your instruction and drive up to you when you call it"
    },
    {
      "role": "user",
      "content": "Tell me how can I use W100 walker."
    },
    {
      "role": "assistant",
      "content": "Make sure you have powered-up W100 and initialized it already. Then you can use your smartphone to call for its attention."
    }
  ]
} 
```

Homework, Due 1 week,
Oct. 5th.

- 1° Install openAI to
use ChatGPT API 3.5;
- 2° Create and Run "Hello,
theWorld" Python code;
- 3° Screen Captures. for 1°
and 2° with Personal ID.
- 4° Submission: On CANVAS.
as zip file. Use the
following Naming convention:
firstName_LastName_4SID_-
ChatGPT1.zip.

Consider Back propagation, or,
"Back Prop" Technique.

Ref: on the github



20225-106a-#20195-38-
lec13-BackProp-activation.
pdf

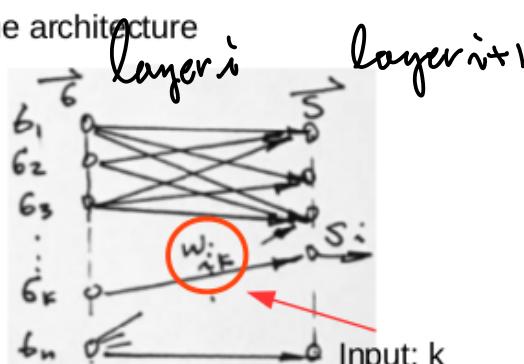
Note 1. Generalize the Architecture from a Single Neuron. Back Propagation (1)

1. input and output neurons

$$\vec{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix}_{n \times 1}$$

$$\vec{s} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{bmatrix}_{m \times 1}$$

The architecture



2. weights w_{ik}
i for output k for input

Note 2: for the Weights Notation

We adopted 2 subscripts

$$\vec{w} = (w_1, w_2, \dots, w_n)$$

$$(\dots, w_{ik}, \dots)$$

4. Transfer function h_i

let

$$h_i = \sum_k w_{ik} g_k \quad \dots (2)$$

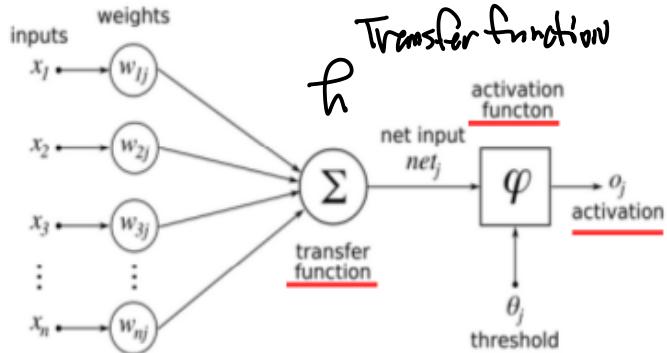
Index i for the output neuron

Neuron output function s_i

$$s_i = f(h_i) = f[h_i(w_{ik})] \quad \dots (2-1)$$

Other popular notation

$$Y = \sum (\text{weight} * \text{input}) + \text{bias}$$



3. Activation function

$$s_i = f\left(\sum_k w_{ik} g_k\right) \quad \dots (1)$$

Note the activation function $f(\cdot)$ and the bias (offset can be written in the unified summation term)

Note 3.

Harry Li, Ph.D.

$$s_i = f\left(\sum_k w_{ik} g_k\right)$$

"i" for
the output
layer.

Input,
feature vector,
excitation.

Note: The New Notation By Substitute Egn(2) into Egn(1).

5. Error at each neuron output (the difference between the true output Zeta (desired true output) and the current output S_i^M) at the experiment Mu

$$S_i^M - S_i^D$$

... (2-2)

Harry Li, Ph.D.

6. total error for all output neuron I and all experiments Mu

$$D = \frac{1}{2} \sum_m \sum_i (S_i^M - S_i^D)^2 \quad \dots (3)$$

MSE? (mean Square Error)

N : No. of Neurons

M : No. of Experiments / Epochs.

$\frac{1}{NM}$ Scaling factor

Note: Rewrite the output S_i^M by its functional form.

7. minimize the error wrt to w_{ik}

$$\begin{aligned} \frac{\partial D}{\partial w_{ik}} &= \frac{\partial}{\partial w_{ik}} \cdot \frac{1}{2} \sum_m \sum_i [S_i^M - f(h_i^M)]^2 \\ &= \sum_m [S_i^M - f(h_i^M)] f'(h_i^M) \frac{\partial h_i}{\partial w_{ik}} \end{aligned}$$

8. Training the NN by updating the weights

$$w_{ik}(t+1) = w_{ik}(t) + \delta w_{ik}(t)$$

... (4)

where

$$\delta w_{ik}(t) = -\epsilon \frac{\partial D}{\partial w_{ik}}$$

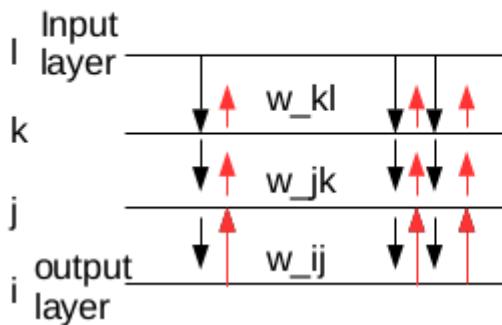
... (5)

Activation function (Transfer function)

Derivative Calculation is Based on "Chain Rule"

Note: Egn(5) on TP. 24.

10. feed forward NN (4 layers)



Blank: for Experiments.

Red: Training Update

Black for the input; red for the back prop training direction

Math. Description.

Back Propagation (4)

11. chain rule, updating the weights (training)

For layer i

$$\frac{\partial D}{\partial w_{ij}} = \sum_m [S_i^m - f(h_i^m)] f'(h_i^m) \frac{\partial h_i^m}{\partial w_{ij}}$$

Note: the training described here all related to the derivative to the activation function, $f'(\cdot)$. So selection of the activation function is important and will be discussed in details next.

For layer j

$$\frac{\partial D}{\partial w_{jk}} = \sum_n \sum_i [S_i^n - f(h_i^n)] f'(h_i^n) \frac{\partial h_i^n}{\partial S_j} \cdot \frac{\partial S_j}{\partial w_{jk}}$$

For layer k

$$\frac{\partial D}{\partial w_{kl}} = \sum_n \sum_i \sum_j [S_i^n - f(h_i^n)] f'(h_i^n) \frac{\partial h_i^n}{\partial S_j} \cdot \frac{\partial S_j}{\partial S_k} \cdot \frac{\partial S_k}{\partial w_{kl}}$$

Project 1 Due 2 weeks
from Next Monday.

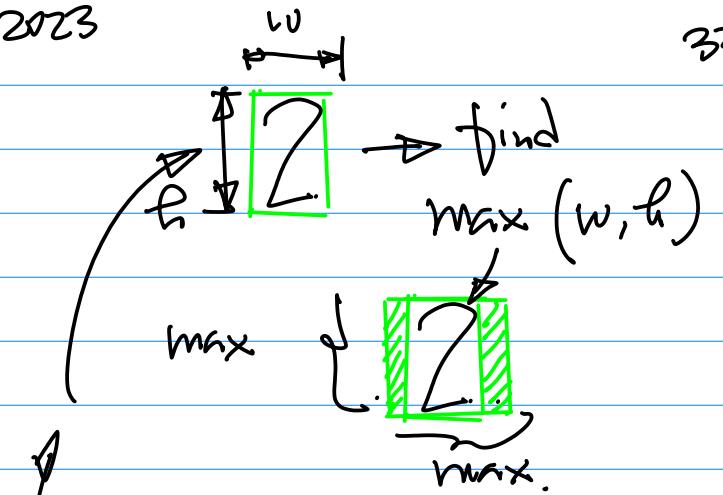
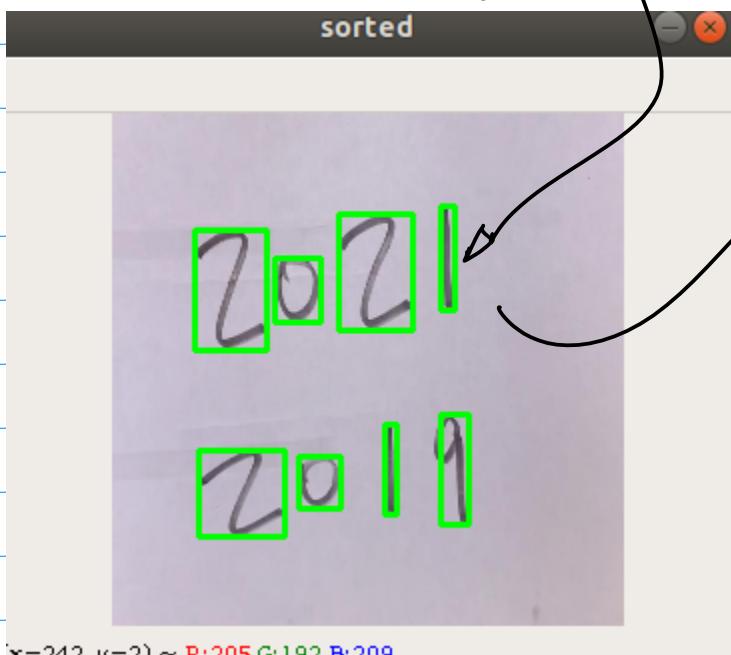
Oct. 3rd (Tue).

Note1 : Move the homework to the Sunday.

Note2: Project 1 Assignment for Handwritten SID Recognition System.

1. Objective To use either Live CAM Input or Video file input to Recognize Handwritten SID. Last 4 Digits Please keep Handwritten Digits as the inputs, No printed digits.

2. Preprocessing of the input image, to get Bounding Boxes



use OpenCV Bitwise Operations

3. Input Image Resolution (1pt).

Bounding Box $B_i(x, y; w, h)$

Width Height
Make sure in the preprocessing you will produce square image \rightarrow preserve the Aspect Ratio of the digits.

$$\lambda = \frac{h}{w} \dots w$$

Example: Preprocessing.

Ref: 2023S-101- ~ -D3-21.

Binarized Image,
Contour Analysis.
Sample Code

~~Ref: 2023S-101-Part2-~
-D4-18.~~

Step1. Capture Image
from a Video, Convert
it to a Gray Scale
image $I(x,y)$.

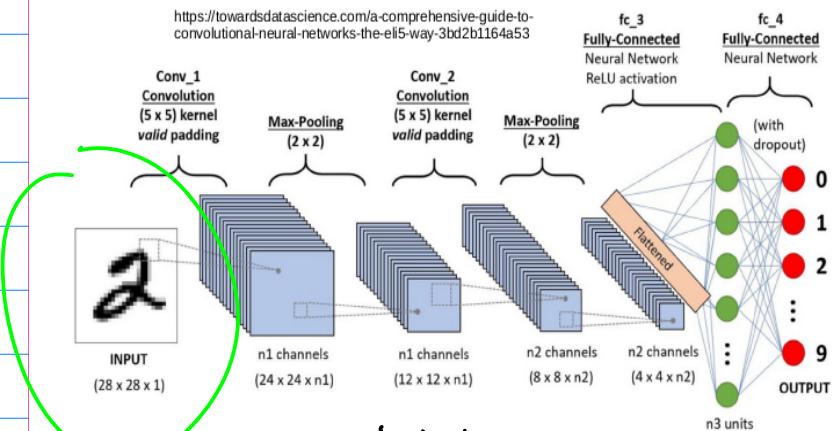
↓
Step2. Canny Edge Detection
to produce Edge
map. $I_{\text{Canny}}(x,y)$

↓
Step3. Contour Analysis.
Find 22 primitive
Contour features.

Draw the contours on the
input preprocessed image.

- a) W,h of a given object
- b) Bounding Box of an object
- c) Shape Descriptor
- d) Moments
- e) Others.

Illustration of A CNN for Digits Recognition



is to be
→ 28x28x1 Changed to Higher
Resolution. Such as 56x56x1
by selecting Proper Convolution
Layers / Maxpooling Layers.
please preserve the Rest of the
MNIST Architecture.

4. Comparison of your implementation
with the conventional MNIST.

5. Provide Readme file,
(good quality document),

Due Oct. 21st, Sat (11:59 PM).

Colab, and Online Tools are
Encouraged, However the submission
of the projects, Homework Require
Python code.

Step 4. Filtering Operation.

Based ON the Contour

$$\text{Mmp. } I_{\text{count}}(x, y)$$

Filter Out the Artifacts.

Look into the features

$$\text{of } I_{\text{count}}(x, y)$$

Step 5. Prepare the image ROI.

And Capture the Image(s)
from the ROI, then form
Square image, to feed
the image (digit) at
time to the CNN Engine.

Example: Continuation of Examples
on Binarized image

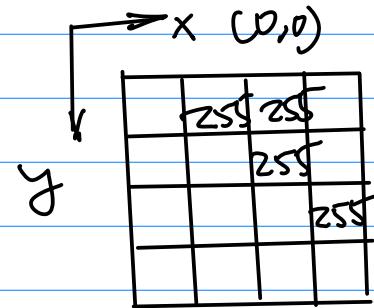
PP. 17 & 18

Motivation for Binarization: To
Extract Objects.

$$\bar{x} = \frac{\iint_S x \cdot B(x, y) dx dy}{\iint_S B(x, y) dx dy} \dots (1)$$

$$\downarrow \quad \iint_S B(x, y) dx dy$$

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



Note: Redefine the top left
corner coordinate as (1,1)
for Hand Calculation. !

$$\begin{aligned} & \cancel{\sum_{y=0}^{N-1} \sum_{x=0}^{M-1} B(x, y)} = \\ & \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} [B(0, y) + B(1, y) + B(2, y) + B(3, y)] \\ & = B(1, 1) + B(2, 1) + B(3, 1) + B(4, 1) + \\ & B(1, 2) + B(2, 2) + B(3, 2) + B(4, 2) + \\ & B(1, 3) + B(2, 3) + B(3, 3) + B(4, 3) + \\ & B(1, 4) + B(2, 4) + B(3, 4) + B(4, 4) \\ & = 255 * 4 \end{aligned}$$

Similarly,

$$\begin{aligned} \text{Now, } & \sum_{y=1}^4 \sum_{x=1}^4 x \cdot B(x, y) \\ & = \sum_{y=1}^4 [0 + B(2, 1) + B(3, 1) + \\ & 3 * B(3, 2) + B(4, 2)] \\ & = 255 * (2 * 1 + 3 * 2 + 4 * 1) \\ & = 255 * 12 \end{aligned}$$

Oct. 5 (Th)

Note 1: Check the CANVAS for the project 1, due Oct. 21st Sat;

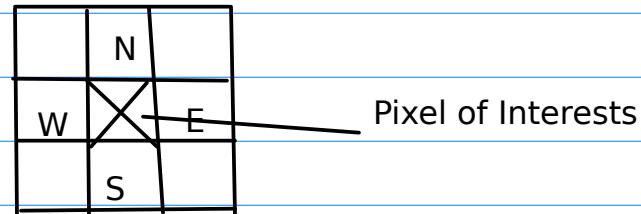
Consider edge detection, e.g., detection of sudden changes of pixel values

Example: X-bar calculation

Continuation from the previous example

hence,

$$X_{\text{bar}} = 12/4 = 3;$$



Similarly, for Y_bar , see the equation from the PPT of the class ref on github.

Define 4 connected neighbors N, S, E, W then the sudden change of the intensity value is defined as the difference between the Pol and any of one of its neighbors.

Denominator from the equation for Y_bar is the same as for X_bar (Area)

Similarly, we can also define 8 connected neighbors, by adding NE, NW, SE, and SW

| ... (4-a)

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

To pick up the sudden change, we use derivative, see pp. 22 of the lecture note

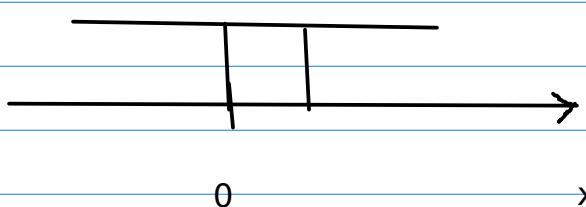
from pp. 18 of this note.

From eqn (2a) for x and (2b) for y, we have forward difference to compute derivatives

Now, let us derive convolutional kernel for this computation

Then the top part of this equation is evaluated:

$$\begin{aligned} & 255 * (1 * B(2,1) + 1 * B(3,1)) \\ & 2 * B(3,2) + 3 * B(4,3)) \\ & = 255 * 7 \end{aligned}$$



Therefore

$$Y_{\text{bar}} = 7/4$$

Note: keep the float for the coding implementation if needed.

Form a small grid with index 0 to map the equation (2a)

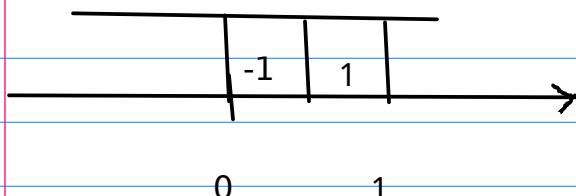
let delta x = 1 for digital image and we have

f(x+1) - f(x) for the right hand side of the equation,

we can rewrite the above form into $1 * f(x+1, y) + (-1) * f(x, y)$ Consider edge detection technique.
ref. lecture note on the github

Map the coefficients to the template,
e.g., kernel

so we have

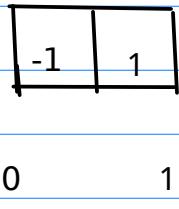


-1		1
-1		1
-1		1

1 / 2

Similarly, we can do the same for
y direction.

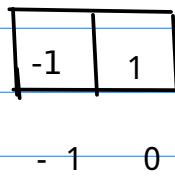
Note for LoG(x,y)



Note this is a kernel but it is not
skewed, not balanced.
So with the modification of derivative
from the forward difference to
backward difference,

$$f(x,y) - f(x-1,y)$$

which give the kernel



Add this kernel (BW) to the kernel (FW)
and divide the result by 2, we have

