

August 23 (Tue)

First Day of the Class

1. Organizational meeting

"Green Sheet"

Repo: github.com/kuaili10/opencv/deep-learning-2023/Email: hua.li@sjtu.edu(656) 490-1116 Cellphone for
Text message Only.

Office Hours: M.W. On Zoom

(See Syllabus for the
Zoom link).

2 Software Tools:

Anaconda — Install it by the end
of this week;

TensorFlow, TF 2.0

OpenCV.

3. Prerequisites: CMPS259 & CMPS251

Homework: To upload a copy of
your un-official transcript to
show the required courses satisfied.

On CANVAS.

4. Textbook: Deep Learning with Python.

Keras (API) for TF

Robot Vision Book By Horn (Horn, Hancock)

Book, (and Reference for OpenCV Algorithms. From the Architecture diagram:

Grad Theoretical Foundations)

5. Projects { Mandatory Assigned Project

Team Project
(Mandatory)(4-Person) Team. Presentation By the
End of the Semester.

August 25 (Wed)

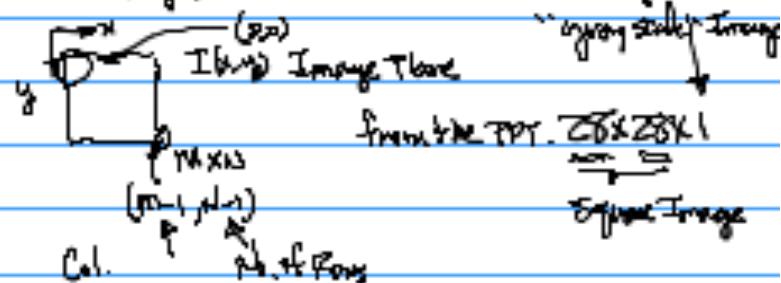
Note: 1. The lecture notes will be posted on
to the github.2. Zoom Recording will be posted on the
the github.Homework: By A week from today. 1. Anaconda
Installation; 2. OpenCV Installation. Submission
on CANVAS. $JPG/PNG/JPEG \rightarrow PDF \rightarrow ZIP$
 ZIP .

Example (github: 2022Fu-13)

Minist Architecture for Handwritten Digit

Recognition.

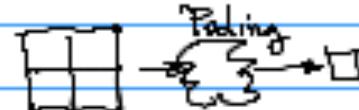
Single channel

(gray scale image \rightarrow 1 channel \rightarrow bit \rightarrow [0, 255])

First Layer of the Minst Architecture

1) Channel/Plane of the 1st convolutional layer

C1

Next, pooling \rightarrow Reduction \rightarrow Resolution \rightarrow 256 bins,

M1

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

COMPENG

Aug 25, 22

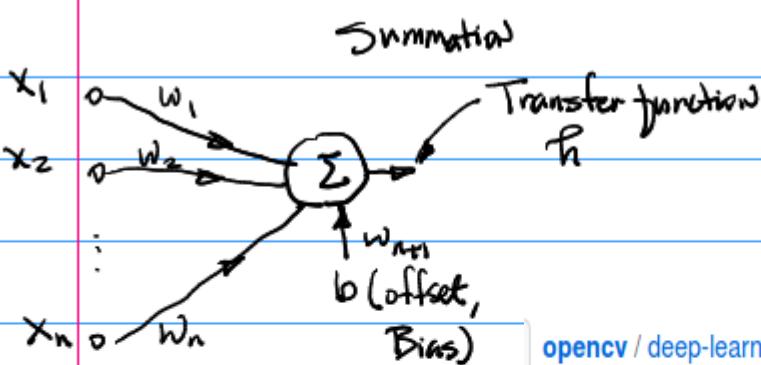
August 30, Tue

2/

Ref. 1

<https://github.com/hualili/opencv/blob/master/deep-learning-2022s/2022F-103b-NN-Intro-Python-v5-2022-8-25.pdf>

[opencv / deep-learning-2022s / 2022S-103a-notation-neuro-loss-function-2022-2-8.pdf](https://github.com/hualili/opencv/blob/2022S-103a-notation-neuro-loss-function-2022-2-8.pdf)



Input / Excitation in Vector Form: $\mathbf{x} = (x_1, x_2, \dots, x_n) \dots \mathbf{v}$

Weights, links each excitation to the Neuron Ref 2. Code

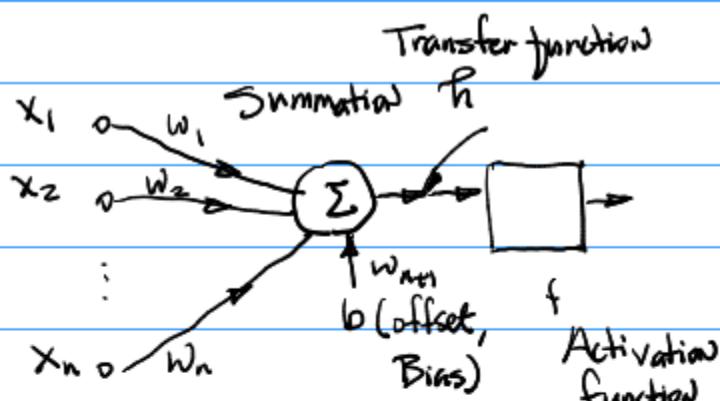
$$\mathbf{W} = (w_1, w_2, \dots, w_n) \dots (z)$$

<https://github.com/hualili/opencv/blob/master/deep-learning-2022s/2022S-110b-%232019S-31-6mnist-numerals-ch02.py>

$$x_1 w_1 + x_2 w_2 + \dots + x_n w_n + b w_{n+1} = f \quad \text{Example:}$$

$$\sum_{i=1}^n x_i w_i + b w_{n+1} = f(x_i w_i) \text{ or Simply } f(\mathbf{x} \mathbf{w}) \dots (3)$$

$$f(\mathbf{x} \mathbf{w}; b), f(\mathbf{x}, \mathbf{w})$$



$$f(f(\mathbf{x} \mathbf{w})) = f\left(\sum_{i=1}^n x_i w_i + b w_{n+1}\right) \dots (4)$$

$$f(\mathbf{x} \mathbf{w}) = \sum_{i=1}^n x_i w_i + b w_{n+1}$$

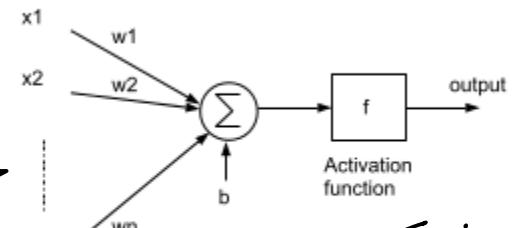


Fig. 1

(x_1, x_2, \dots, x_n) Feature Vector with Dimension N.

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

Transfer function $f(\cdot)$.

$$w_{n+1} b = b'$$

Examples of Different Activation functions

include RELU. A piecewise linear.

Note: Be Able to Build A Single Neuron per a technical specification, Such as

ReLU, Activation $f(\cdot)$, Draw a Block

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

Activation function. Its output is the Response of the Neuron.

Aug. 30.

Consider the output of the Neuron

 y from Eqn(17).

Output of a Single Neuron.

For Multiple Neuron Output, see Fig.2

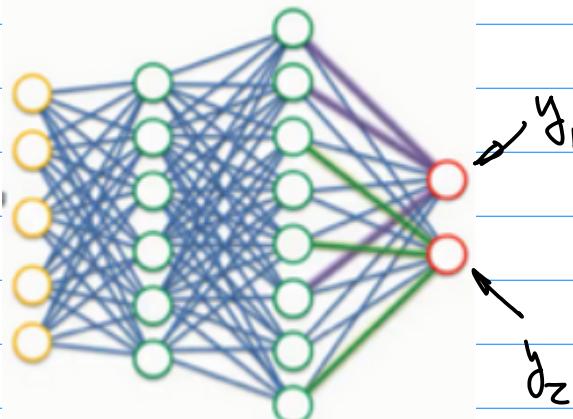


Fig.2

 y_{di} ... (1)Subscript: $i=1, 2$

No. of Output at the Output Layer.

 $y_i, i=1, 2, \dots, M$.

In practical Application,

 y_{ij}^* ... (2)

Superscript

 $j=1, 2, \dots, P$ No. of Experiments

Performed, Training Performed.

Look at the Concept & Definition of Loss function.

Mathematically To Compare a Neural Network Output (Single Neuron Output)

function f . function g Comparison of the Similarity or difference between f and g . $f - g$ f/g

Difference Between Two Functions.

Take this Approach to define Loss function,

 $y - \hat{y}$... (3)

Ground Truth.

Output (Prediction) from the Neuron

Outputs

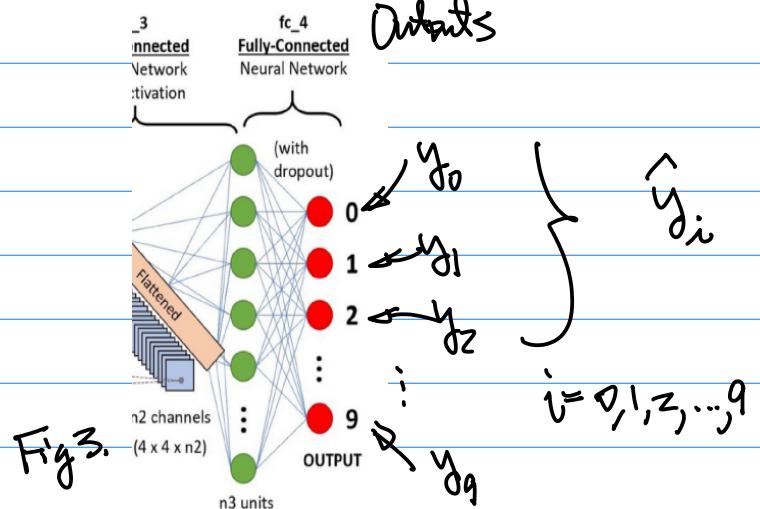


Fig.3.

August 30.

$$\hat{y}_i - \tilde{y}_i \quad \dots (4-a)$$

To measure All the content for Each Training Experiment

$$\sum_{i=0}^q (\hat{y}_i - \tilde{y}_i) \quad \dots (4-b)$$

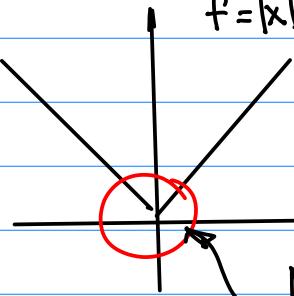
Expand this to Experiment/Training up to "P" Times

$$\sum_{j=1}^P \left[\sum_{i=0}^q (\hat{y}_i - \tilde{y}_i) \right] \quad \dots (4-c)$$

Note: Eqn (4-c) may lead to positive & Negative Terms Cancellation.

Fix: Absolute Value? \rightarrow Squared Instead,

$$f = |x|$$



Not "well-behaved"!

$$L_{total} = \frac{1}{2} \sum_{j=1}^P (\tilde{y}_i^j - y_i^j)^2. \quad \text{Ground Truth} \quad (23)$$

For a Single Neuron @ the Output Layer

Training Based "Steepest Gradient Descent" Example: Given A

function $f(x) = x^2$, Find its Derivative

$$\frac{df}{dx} = \frac{d}{dx} x^2$$

To get rid of Coefficient from the derivative,

$$\text{Let's define } f(x) \triangleq \frac{1}{2} x^2$$

$$\frac{d}{dx} f(x) = \frac{1}{2} \cdot 2 \cdot x = x$$

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

Sept 1st (Th).

Ref:

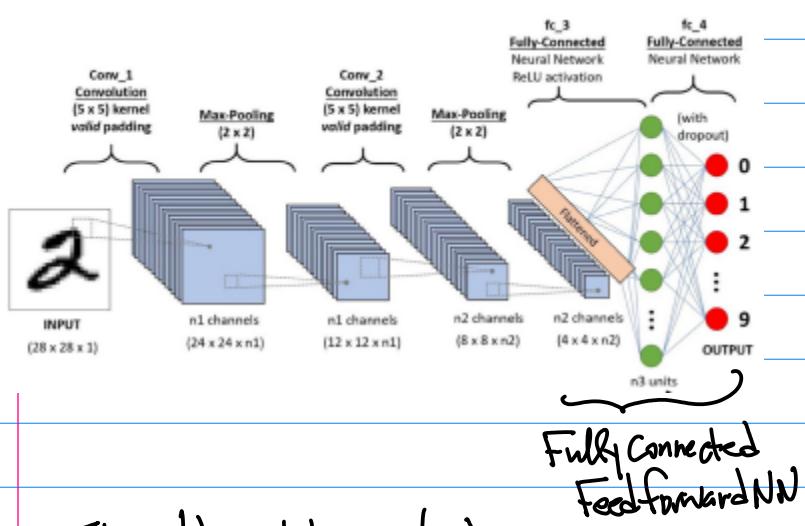
[2022S-103a-notation-neuro-loss-function-2022-2-8.pdf](#)

Example: Background D) "Learning" of A NN.

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

$$L = \sum_{j=1}^P \left[\sum_{i=0}^q (\tilde{y}_i^j - y_i^j)^2 \right] \quad \dots (4-d)$$

J, or Φ



To Train AN, we take eqn (23), e.g.
Loss function (error function),

$$L(\cdot) \rightarrow L(W_i)$$

Independent Variables

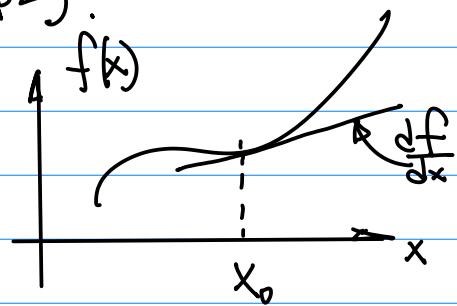
Minimize the Loss $L(\cdot)$, which is the process of Training, which leads to Learning for the NN.

Since the Loss function in Eqn (23) is formulated with a ground truth, this defines a Supervised Learning.

Math. Formula: Prediction of Function's Behavior, we like to know given the current function value (Loss function) how this function is going to change at next moment, increase? stay the same?

Or Decrease?

Derivative of A given function is a good indicator to give us the description of the function behavior
Next Step Ahead (Very small tiny step).



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

f Intuition

$f(x+\Delta x) - f(x) \rightarrow > 0$ (derivative) if the derivative is greater than 0
 $\Delta x \approx 1$ unit

then $f(x+\Delta x) > f(x)$, the next Step function $f(x)$ is increased;

if the derivative $\frac{df}{dx} < 0$, then

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

if the derivative $\frac{df}{dx} = 0$, then
 $f(x+\Delta x) = f(x)$

Consider Two Dimensional Case as an Example for n-dimensional Case.

Sept. 1st.

6

$$f(x_1, x_2) \left\{ \begin{array}{l} \frac{\partial f(x_1, x_2)}{\partial x_1} \\ \frac{\partial f(x_1, x_2)}{\partial x_2} \end{array} \right. \rightarrow \text{Single Neuron}$$

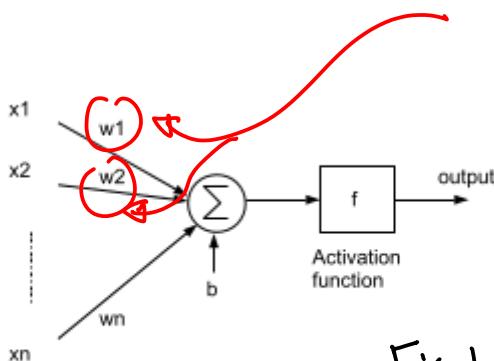


Fig.1

$\frac{\partial f}{\partial x_i} \rightarrow \frac{\partial f}{\partial w_i}$, In the Context of
the Training.

Conclusion: Use Partial Derivative

With respect to the weight w_i as an indicator to measure if the Loss function is got reduced or not.

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

Example: Consider A Technique which allows training to be more effective, e.g., to minimize training and prediction error (Loss) function

Steps for Development of this technique :

In Case of Training
They are "Weights"

w_1, w_2, \dots

Multi-dimensional Derivative

gradient

the Steepest gradient

the Steepest Descent gradient

The core technique to train NN. SGD

Ref: from the github

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

Homework (Opt), Due A week from Today.
Sept. 8th, Thursday. On CANVAS.

1^o Installation of TensorFlow, Version 2.1 or higher.

2^o Screen Capture to show the installation is successful

Note: All Different Development Tools/Environments including google colab, jupyter Note Book etc. Are OK, However for the Deployment purpose, projects homework Submission must be in Python Stand-Alone form.

Sept. 6 (Th)

Homework: Due 1 week from Today Sept 13.

1. OpenCV Installation, Python.
2. Use Smart phone to Capture 5~10 Seconds Video Clips.
.avi, .mp4 (mpeg4).
3. Sample Code, github.
See CANVAS for the Detailed Links & Requirements.

[https://github.com/hualili/opencv
/blob/master/deep-learning-2022s/
2022S-104d-%232-pdisplay-2019-1-30.py](https://github.com/hualili/opencv/blob/master/deep-learning-2022s/2022S-104d-%232-pdisplay-2019-1-30.py)

Higher Dimensional Function

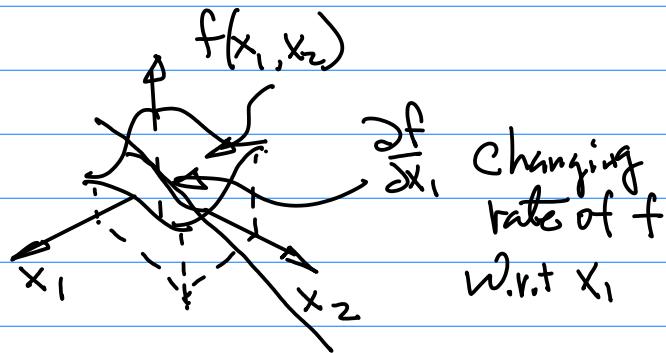
$$f(x_1, x_2, \dots, x_n) \dots (z)$$

Weights, w_1, w_2, \dots, w_N

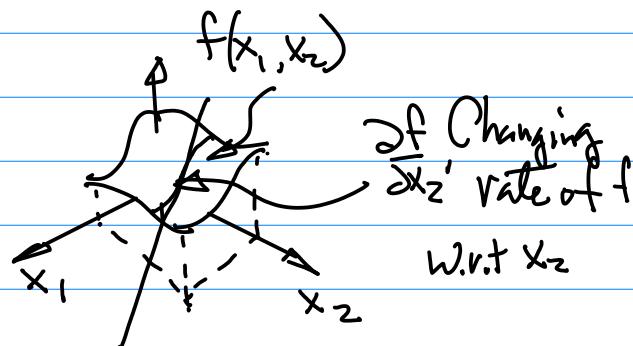
Partial Derivatives:

$$\frac{\partial f}{\partial x_1} \text{ w.r.t } x_1, \frac{\partial f}{\partial x_2} \text{ for } x_2, \dots$$

$$\frac{\partial f}{\partial x_n} \text{ w.r.t } x_n.$$



Changing rate of f
w.r.t x_1



Changing rate of f
w.r.t x_2

Example: Gradient Definition

Ref:

[https://github.com/hualili/opencv
/blob/master/deep-learning-2022s/2022S-105c-%2320-2021S-
4gradient-descent-v2-final-2021-2-8.pdf](https://github.com/hualili/opencv/blob/master/deep-learning-2022s/2022S-105c-%2320-2021S-4gradient-descent-v2-final-2021-2-8.pdf)

Loss function

Derivative, e.g. Given $f(x)$, then

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

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

Sept. 6, 22

Consider the minimization of function f (Loss Function) w.r.t. All possible weights.

Therefore, put all the partial derivatives together to form a vector, e.g., gradient.

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

for $n=2$,

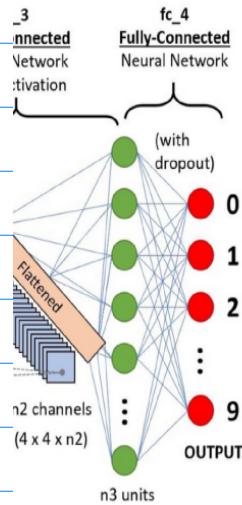
$$\nabla f(x_1, x_2) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} \dots (zb)$$

for $n=3$

$$\nabla f(x_1, x_2, x_3) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \end{pmatrix} \dots (zc)$$

c. On the Right hand side of Eqn(5):
 (x_1^k, x_2^k) Dimension $n=2$, (x_1, x_2)
 Time Index "K", Superscript

Output of the NN with its weights at Step K (Time) is



$$x_1^k, x_2^k$$

e. On the left (x_1^{k+1}, x_2^{k+2}) , at the step $k+1$, to Reduce the Loss function, so update the new step by following

$$-\nabla f = -\eta \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} \dots (z)$$

IV. GRADIENT STEEPEST DESCENT FOR MINIMIZATION

Conclusion:

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

a. Loss function
 f

(5) b. $n=2$

e.g.

$$f(x_1, x_2, \dots, x_n)$$

$$\Rightarrow f(x_1, x_2)$$

c. Gradient

$$\nabla f(x_1, x_2), \text{ or}$$

$$\nabla f$$

Background:

Given a function $f(x)$, How do you Approximate this function By using Basic Building Blocks (B^3)?

$$f(x) = \text{Constant} + \text{A Linear Term} + \text{A Quadratic Term} + \text{A Cubic Term} + \dots \dots (4)$$

CmpE258

Sept 6, 22

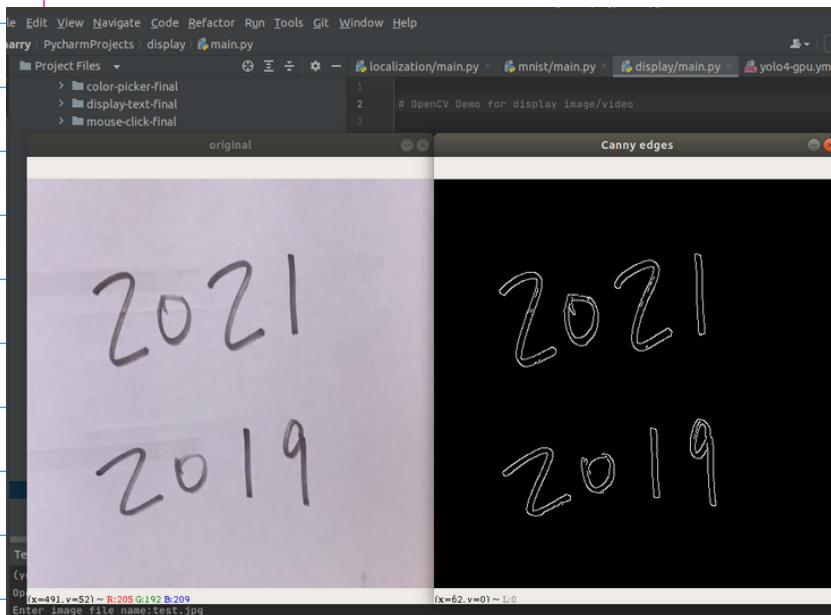
9.

Taylor Expansion:

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

$$f(x_1, x_2) = f(x_1, x_2) \Big|_{\substack{\text{Constant} \\ x_1=x_{10} \\ x_2=x_{20}}} + \frac{\partial f}{\partial x_1}(x_1 - x_{10}) + \frac{\partial f}{\partial x_2}(x_2 - x_{20})$$

Note: The screen capture for your homework reference.



Sept. 8 (Thu)

Note: 1^o Check the CANVAS for Both Homeworks.

Example: From 1D Case in Eqn(4), we can expand the Taylor Expansion to higher dimension n. to Capture multiple excitations, multiple weights $w_i, i=1, 2, \dots, n$.

Consider n=2

$$+ \frac{\partial^2 f}{\partial x_1^2}(x_1 - x_{10})^2 + \frac{\partial^2 f}{\partial x_2^2}(x_2 - x_{20})^2 + \dots + R_n(x_1, x_2)$$

... (5)

Higher Order Terms

The goal is to verify the formula for updating the weights of A given NN. (see Handout 1, Eqn(5)).

Step 1: Taylor Expansion \rightarrow Step 2

Simplify the Taylor Expansion By just using upto the Linear terms \rightarrow Step 3: Re-arrange the Taylor Expansion in the form of Training formula (In Eqn(5), in Handout 1) \rightarrow Step 4: Analyze the rearranged formula, to Reach the Conclusion, e.g., Using gradient descent, we can Reduce the Loss

function through each step of the training.

From the Handout, we have as Step 1 & 2:

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

$$\underbrace{f(x_1, x_2) - f(a, b)}_{\text{Comparison of A "loss" function}} \approx f_{x_1}(x_1 - a) + f_{x_2}(x_2 - b)$$

Comparison of A "loss" function, write

$$\Delta x_1 = x_1 - a$$

$$\Delta x_2 = x_2 - b$$

And $\nabla f = \begin{pmatrix} f_{x_1} \\ f_{x_2} \end{pmatrix}$

Hence, we have

$$f(x_1, x_2) - f(a, b) = (\Delta x_1, \Delta x_2) \cdot \nabla f$$

$$\text{Let } \Delta x_1 = -f_{x_1}, \Delta x_2 = -f_{x_2}$$

Therefore,

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

$$= -\left(f_{x_1}^2 + f_{x_2}^2\right) \leq 0$$

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

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

Loss function updated at the next step by Eqn(5) in Handout 1.

Note: The requirement for this discussion on Notations, And Formulation, especially, the Eqn(5) in Handout 1 is required

To Be Able to use these tools to analyze the problem, And to perform Verification (Eng. design).

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

To Be Continued.

OpenCV Homework. Sample code

[opencv/2022S-104d-#2-pdisplay-2](#)

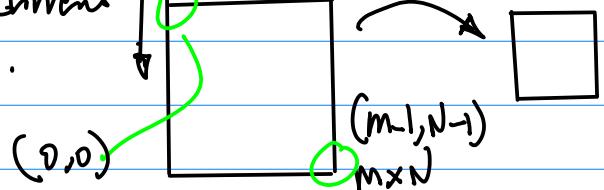
```

9     #include<sys.argv[1]>
10    window_name = 'Display Image'
11
12    imageName = sys.argv[1] #get file name from command line
13
14    src = cv2.imread(imageName, cv2.IMREAD_COLOR)
15
16    if src is None:
17        print ('Error opening image!')
18        print ('Usage: pdisplay.py image_name\n')
19
20    ind = 0
21
22    while True:
23        cv2.imshow(window_name, src)
24
25        c = cv2.waitKey(500)
26        if c == 27: #ESC

```

Loss function at the current Step.

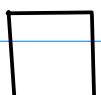
(0,0)



Note: OpenCV Resize

Function.

Resize



```

12 import numpy as np
13 #import argparse
14 import cv2
15
16 img = input('Enter image file name:')
17
18 image = cv2.imread(img, cv2.IMREAD_COLOR)
19
20 if image is None:
21     print('Error opening image!')
22     print('Usage: pdisplay.py image_name\n')
23
24 image = cv2.resize(image, (512, 512))
25
26 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) a.
27 edges = cv2.Canny(gray, 100, 200) b.

```

- a. Conversion to Gray-Scale Image;
- b. Canny Edge Detection.

c. Preprocessing of the dataset

Name	x_1	Weight (minus 135)	Height (minus 66)	Gender
Alice	-2	-1	1	
Bob	25	6	0	
Charlie	17	4	0	
Diana	-15	-6	1	

Sept.13 (Tue)

Homework: (opt) Due to A week from today. Capture the screen to the T.F. Installation is successful.

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

a. Feature Vectors

Example: Collecting data for training

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

Sig	Mij	Mpq
V1	133	65
V2	160	72
V3	152	70
V4	120	60

Sign
Stop
Right
Right
Stop

$$V = (v_1, v_2)$$

$$\Xi = (x_1, x_2)$$

$$\Xi_i = (x_{i1}, x_{i2})$$

b. Ground

Truth y_i corresponds to Ξ_i

Simple feedforward NN.

Step1. Match the Dimension of the

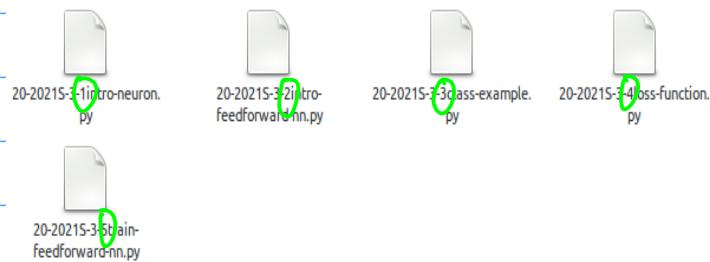
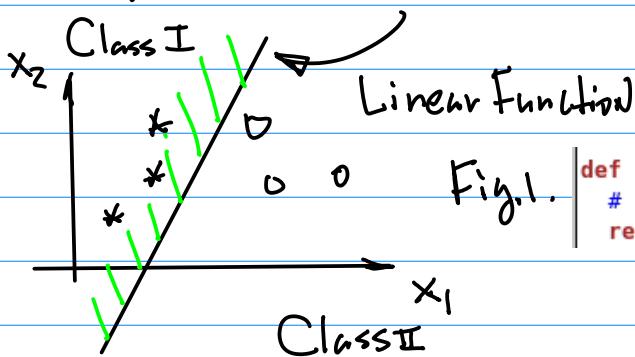
Feature Vector Ξ_i to the Input Neurons

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

(24)

Also, Determining the Output Layer: Now, Let's use Python Code to Inspect the Nature of this Prediction, Hence, Choose One Single Neuron Output

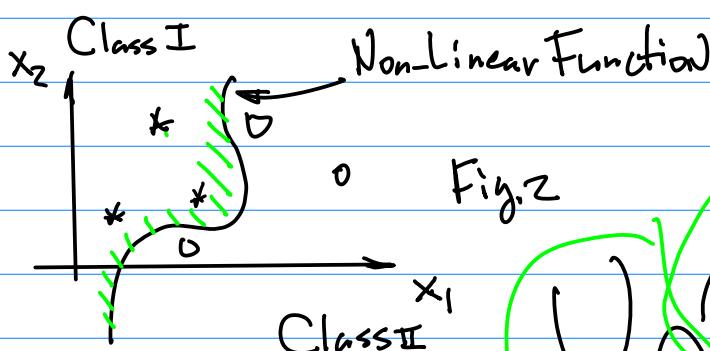
Hidden Layer(s) → Affects the Behavior of Decision-making function.



Define An Activation function.

$$y = f(h(w_i x_i; b)) = f(\mathbf{I} \mathbf{w})$$

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

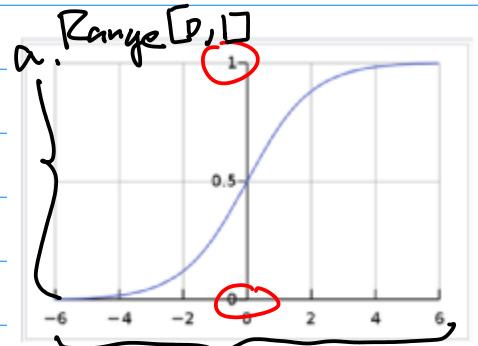
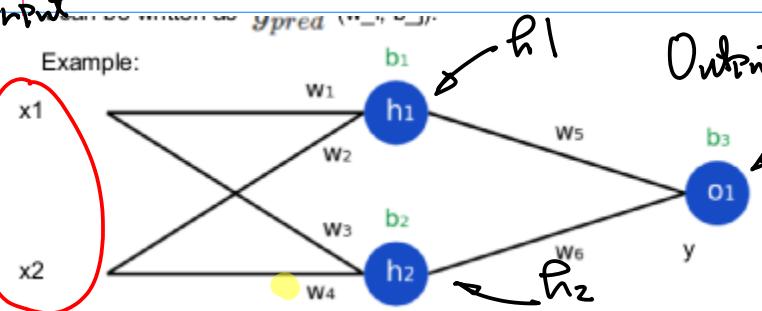


No. of Hidden Layers →

For the Simplicity in this example, Let's choose just 1 Hidden Layer.

Quadratic

\mathbf{I} Input



b. Excitation Range.
 $f(h(\mathbf{I} \mathbf{w}))$

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

Note: A Sigmoid Function. (1)
A sigmoid function is a

The Derivatives of the Activation Function. Done By hand Calculation of the Derivative.

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

$$\frac{d}{dx} \text{Sigmoid}(x) = \frac{d}{dx} \frac{1}{1+e^{-x}}$$

Define the Loss function

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

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

Mean Square Error
Ground Truth NN output
 \tilde{y}_i^j

Note: The Derivative Code in
 this Example Allows Training
 Implement By Gradient Descent.

```
# Biases
self.b1 = np.random.normal()
self.b2 = np.random.normal()
self.b3 = np.random.normal()

def feedforward(self, x):
    # x is a numpy array with 2 elements.
    h1 = sigmoid(self.w1 * x[0] + self.w2 * x[1] + self.b1)
    h2 = sigmoid(self.w3 * x[0] + self.w4 * x[1] + self.b2)
    o1 = sigmoid(self.w5 * h1 + self.w6 * h2 + self.b3)
    return o1
```

(24)

Define the F.F. Neural Network.

Output:

$$y = f(wx) \quad |_{f = \frac{1}{1+e^{-x}}}$$

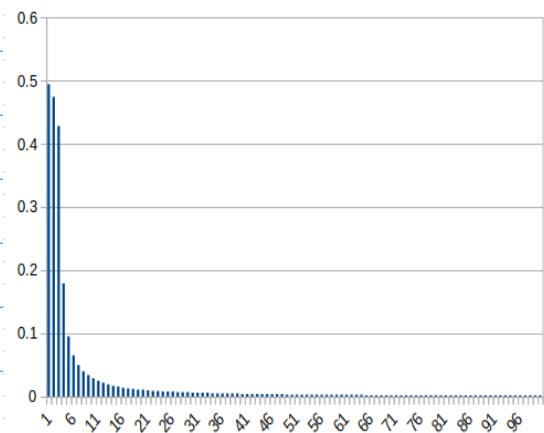
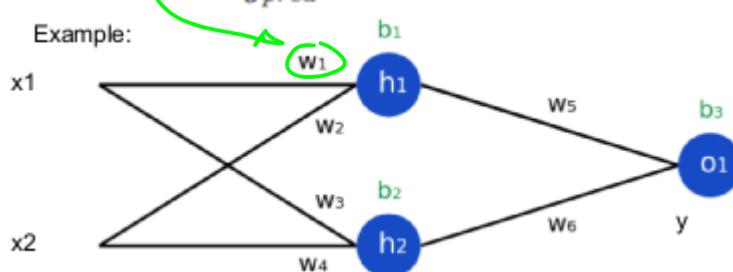
where x is from the
 hidden neurons, h_1 & h_2

https://github.com/hualili/opencv/blob/master/deep-learning-2022s/2022S-103c-%23nn_sample_2022.py

The weight initialization.

```
def __init__(self):
    # Weights
    self.w1 = np.random.normal()
    self.w2 = np.random.normal()
    self.w3 = np.random.normal()
    self.w4 = np.random.normal()
    self.w5 = np.random.normal()
    self.w6 = np.random.normal()
```

Example:



Sept 15 (Thu)

Today's Topics:

- 1° Preliminary Sample Code (Python)
- 2° Preprocessing Technique (Computer Vision / OpenCV) for Deep Convolutional NN. Handwritten Digits Recognition

Ref: From Python Sample Code on the github

loss is being Computed \rightarrow

98

 $d_L_d_{ypred} = -2 * (y_{true} - y_{pred})$

```
74     def train(self, data, all_y_trues):
157     #-----
158     # Define dataset and all_y_trues
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
```

a. Training will take the input from here

then, Compute partial derivatives to form gradient

then, update the weight(s) Based on gradient descent

b. Initialization. w

80 learn_rate = 0.1

81 epochs = 1000 # number of times to loop through

The Number of Trainings is defined with this as a upper Bound if the loss function during the training drops below the pre-set threshold then the training Process will terminate

The Data Point (e.g. $\mathbf{x}(x_1, x_2)$) is substituted into the code to evaluate Transfer function $f(w, x)$, then evaluate the Activation function $f(f(w, x)) = \frac{1}{1 + e^{-x}}$

$$f(f(w, x)) = \frac{1}{1 + e^{-x}} \quad |_{x=f(w, x)}$$

```
85     # --- DO A FEEDFORWARD (WE'LL NEED THESE VALUES LATER)
86     sum_h1 = self.w1 * x[0] + self.w2 * x[1] + self.b1
87     h1 = sigmoid(sum_h1)
```

The process propagates through each neuron & each layer till reaches the Output

```
92     sum_o1 = self.w3 * h1 + self.w6 * h2 + self.b3
93     o1 = sigmoid(sum_o1)
94     y_pred = o1
```

96 # --- Calculate partial derivatives.
97 # --- Naming: d_L_d_w1 represents "partial L / partial w1"
98 d_L_d_ypred = -2 * (y_true - y_pred)

Compute total Loss function

```
136
137     self.w1 := learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w1
138     self.w2 := learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w2
139     self.h1 := learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_h1
```

--- Calculate total loss at the end of each epoch

```
151     if epoch % 10 == 0:
152         y_preds = np.apply_along_axis(self.feedforward, 1, data)
153         loss = mse_loss(all_y_trues, y_preds)
154         print("Epoch %d loss: %.3f" % (epoch, loss))
```

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

Note: the ground truth is given in the code

```
178     all_y_trues = np.array([
179         1, # person A
180         1, # person A
181         1, # person A
182         0, # person B
183         0, # person B
184         0, # person B
185     ])
```

Annotation

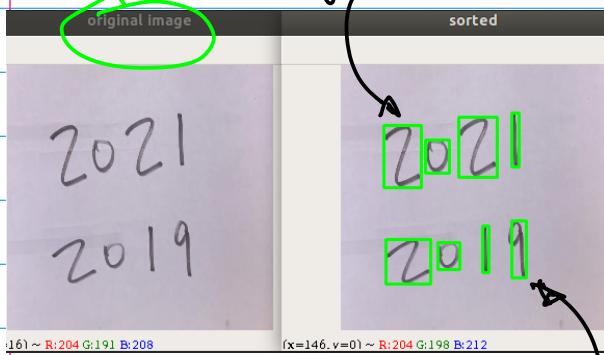
CMPT258
Sept. 15, 22

15

Example: Pre-processing Techniques.
for Handwritten Digits Recognition.

Objective: To Be Able to Localize
the ROIs. (Region of Interests).

a. ROI: Bounding Boxes



b. Localization: localized on
Each hand written digits
Ref: 1. pp.18 (Starting from)

2022S-101-cmpe258-2022-03-15-Note.pdf

2. 2022S-108-Intro-Binary.pdf PPT.
with math. Formulation.

Background on Color Images.

a. Video Clip.
(A Sequence of Images)
From Ref. 1.

b. A Single frame of
an image $I(x,y,t) \rightarrow I(x,y)$

c. ... (1) at time t

Each Frame of A color image

$I(x,y)$ consists of 3 channels.

Or. Primitive Color planes, Red,
green, Blue

In OpenCV, (Color) Images are organized
as B, G, R.

$$I_r(x,y) \in [0,255], I_g(x,y) \in [0,255] \\ 8 \text{ bit} \\ Z^8 = 256 \quad I_b(x,y) \in [0,255] \\ \dots (z)$$

Color Image \rightarrow Gray Scale Image.
Example in Fig. 1.

Fig. 1.

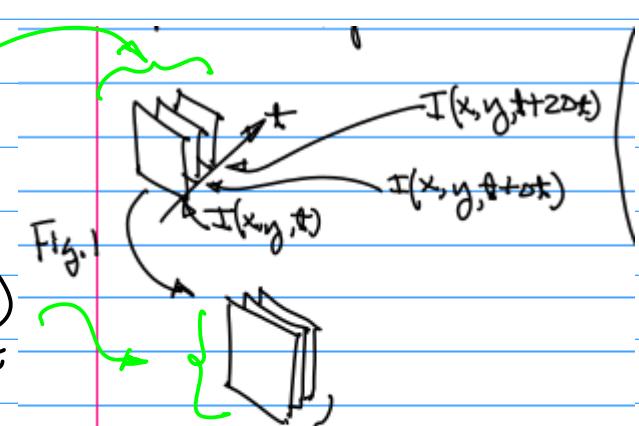
$I(x,y)$

$I_{gray}(x,y)$

$$I_{gray}(x,y) = \frac{1}{3} (I_r(x,y) + I_g(x,y) + I_b(x,y)) \\ \dots (3)$$

Binary Image.

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



3 Image planes (r, g, b)
Pixel depth, Color Depth: 8 bits per
Each Color plane
 $r \in [0,255], g \in [0,255], b \in [0,255]$

CMPE258
Sept. 15/22

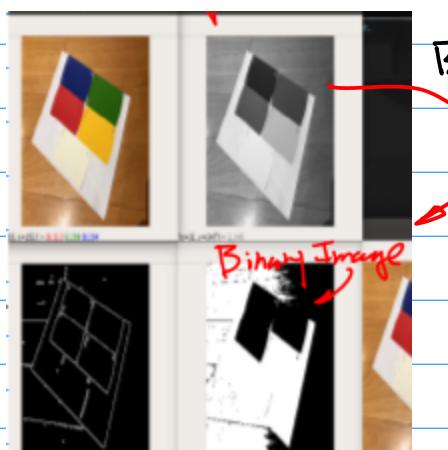
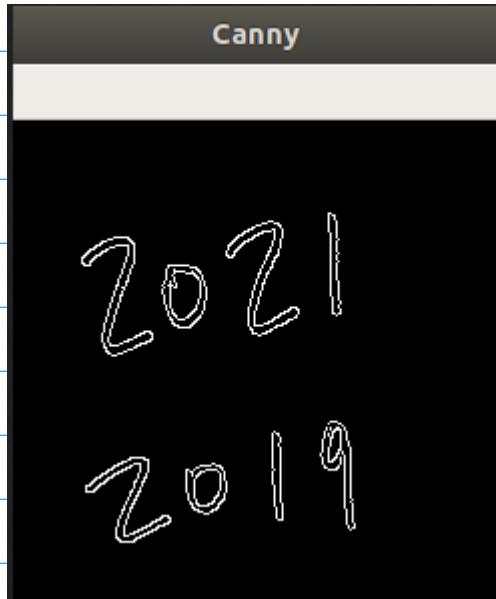
16

Example of A Binary Image
(This Binarized image is
Canny Edge Detection) Not Commonly used

From Eqn(4) Example:

2022S-108c-example-binary.pdf

Binarization Image.



By Eqn(4)

Binary Image

2022S-108-Intro-Binary.pdf

Operators for
Preprocessing (nb)

Sept 20 (Tuesday)

Note: 1° Next Class we will have in-class exercise, Regnies Run OpenCV Program, a) Have your Laptop Ready to Run OpenCV Program; b) Bring A blank Printer Paper to the Class, to Be used in the exercise; c) Bring your Smart phone; d) Check to Make sure your Video is working.

2° A New Homework will be posted on CANVAS tomorrow, please check.

3° For the Semester-Long Team Project, we will post the Rubrics & Requirements on Line (on CANVAS).

Pattern Recognition For Binary Images

The tool box for pattern recognition for binary images

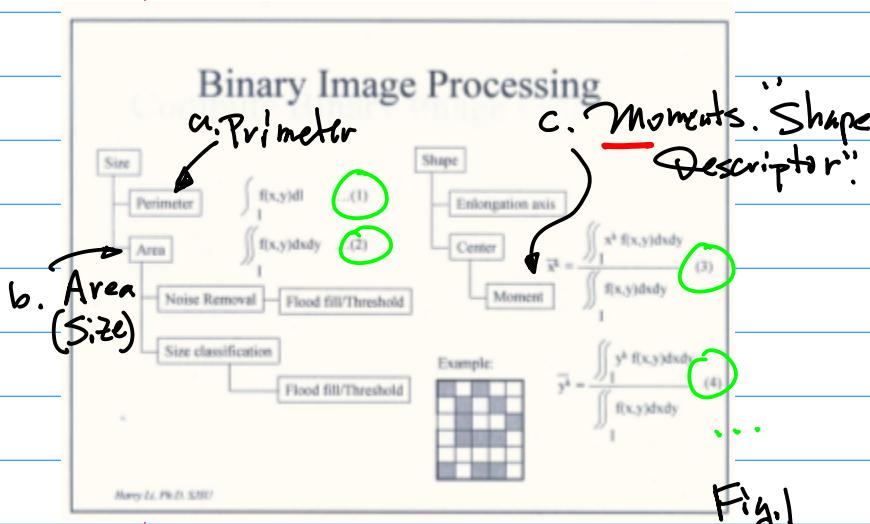
- 1. Size
- 2. Moments
 - \bar{x}
 - \bar{y}
 - x^k
 - y^k etc.
- 3. Perimeter
- 4. Orientation
- 5. Compositions of the above
 - Perimeter and moments: vector
- 6. Invariant operators
 - size invariant
 - orientation invariant
 - illumination invariant

Biologically inspired techniques

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

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

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



$f(0,0) < T$, Hence $B(0,0) = 0$;
 $\text{for}(0,1)$, Similarly, $f(0,1) = 70$, Hence,
 $B(0,1) = 0$;

...

...

$\text{for}(3,0)$, $f(3,0) = 21 \geq T$, Therefore

$$B(3,0) = 255$$

e.g. often this
255 in Hand
Calculation

Can be replaced by 1.
(Scaling factor)

Example: Calculation of Binary Image.
Based on Eqn (4)

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

$\text{for}(3,1)$, $f(3,1) = 209 \geq T$, Therefore

$$B(3,1) = 255$$

	7	$f(x,y)$
70	70	80
75	85	140
90	210	101
211	209	115

Given an image $f(x,y)$

Find a proper threshold

T , use the T to

Perform Binarization.

Selection of "T" is based on Heuristics,
e.g. Human Expert Knowledge.

if $f(x,y)$, at Location (x,y) , is
greater to Equal to T .

$$f(x,y) \geq T ?$$

$\text{for}(0,0)$, $f(0,0) \geq 135$; $\therefore f(0,0) = 70$,

		K_x
K_y	0	0 0 0 255
0	0 0 255 0	0 255 0 0
	255 255 0 0	

No. of pix \leq
Per Row
 \times
No. of
Rows per
frame.

Example: Calculate the Size, e.g. Area, of the Binary Image $B(x,y)$.

Definition 1. (AREA/Size)

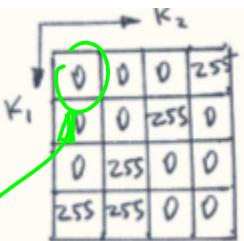
$$\iint_{\Omega} B(x,y) dx dy \dots (1)$$

Digitized Image / Discrete Function

$$\iint_{\Omega} \rightarrow \sum_{y=0}^{M-1} \sum_{x=0}^{N-1}$$

Hence,

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



$$\bar{y}^k$$

$$\frac{\iint_{\Omega} y^k B(x, y) dx dy}{\iint_{\Omega} B(x, y) dx dy} \dots (3)$$

where $k=1, 2, \dots, K$. When $k=1$, \bar{y} is a Centroid along y -axis.

In Addition, Extend Eqn. (2) & (3) to the following.

Sol:

$$\sum_{y=0}^{m-1} \sum_{x=0}^{N-1} B(x, y) = \sum_{y=0}^{m-1} \left(\sum_{x=0}^{N-1} B(x, y) \right) m_{pq}(x, y) = \frac{\iint_{\Omega} (x-\bar{x})^p (y-\bar{y})^q B(x, y) dx dy}{\iint_{\Omega} B(x, y) dx dy} \dots (4)$$

$$= \sum_{y=0}^{m-1} [B(0, y) + B(1, y) + \dots + B(N-1, y)] \Big|_{N=4}$$

$$= \sum_{y=0}^{m-1} [B(0, y) + B(1, y) + \dots + B(3, y)]$$

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

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

+ ...

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

$$= 5 \times 255 \quad (\rightarrow \text{Normalize it!})$$

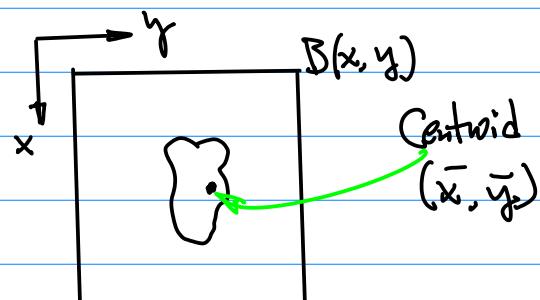
Hence, the size is 5.

Example: Computation of moments

Definition 2.

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

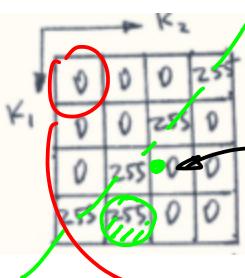
where $k=1, 2, \dots, K$; when $k=1$, \bar{x} "Centroid" along x -axis



where $T, q = 0, 1, 2, \dots, K$
 "Hu-moments" → Very Good Way to Describe Object Shape

Example: Find Centroid (\bar{x}, \bar{y}) of A given object

Since we now start from $(1,1)$,
we have



Visual Inspection
for (\bar{x}, \bar{y}) .

Sept. 22 (Thu).

Example: Pre-Processing Technique.

\bar{x}, \bar{y} Centroid.

From the previous Note.

$$\bar{x}^k = \frac{\iint_{S_k} x^k B(x, y) dxdy}{\iint_{S_k} B(x, y) dxdy} \dots (z)$$

Rewrite Eqn (z) as

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

\downarrow

AREA = 5

Now,

$$\begin{aligned} & \sum_{y=0}^3 \sum_{x=0}^3 x B(x, y) \\ &= \sum_{y=0}^3 [0 \cdot B(0, y) + 1 \cdot B(1, y) + \\ & \quad 2 \cdot B(2, y) + 3 \cdot B(3, y)] \end{aligned}$$

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

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

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

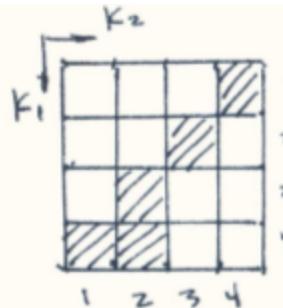
...
...

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

Therefore

$$\bar{x} = 14 / 5 = 14 / 5 = 2.8 \approx 3$$

For Visualizing
the Result



Keep this for
Continued Computation
in your program.

Note: for Smaller Images, like the one we have
for the Simplicity of our discussion we
often to make the index X and Y
starts from $(1,1)$ Not $(0,0)$.

Similarly,

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

See the handout on the Class
github, we have

$$\bar{y} = \frac{12}{5} = 2.4 \approx 2$$

For Visualization

Purpose Only.

Example: Consider 2D Convolution
of a given Image.



INPUT
(28 x 28 x 1)

$m \times n$, $m = n = 28$

A single channel.
"A grayscale image"

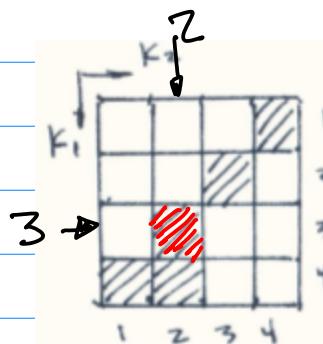


Fig. 1.

20	20	0	0
20	20	0	0
10	10	0	0
10	10	0	0

1	0	-1
1	0	-1
1	0	-1

A Kernel
 $K \times K$

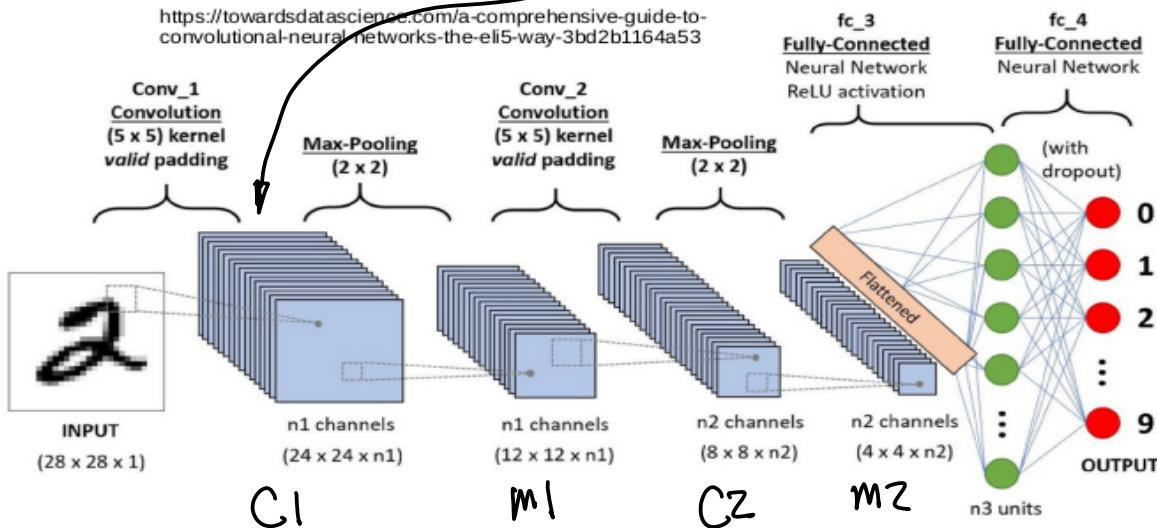
20	20	0	0
20	20	0	0
10	10	0	0
10	10	0	0

$I(x, y)$ $(m \times n)$ ($m = n = 4$)

Fig. 2

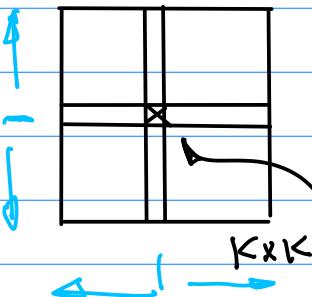
Now, consider Convolution Technique.

Illustration of A CNN for Digits Recognition



About the Kernel:

$K = \text{"Odd" Number.}$



Pixel of Interest

Fig. 3

$$\sum_{v=0}^{N-1} \sum_{u=0}^{M-1} I(u,v) g(x-u, y-v)$$

... (z)

a. Summation Index U, V ,

for $u=0, 1, 2, \dots, M-1$,
 $v=0, 1, \dots, N-1$.

Resolution of The Image $I(x,y)$
 $i \leq M \times N$.

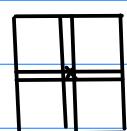
b. Kernel $g(x,y)$ is K -by- K ($K \times K$)
 in Size, K is always an "odd"
 Number.

The center pixel of the Kernel \rightarrow
 pixel of Interest;

"Odd" Number Makes No. of col. (Right
 & Left) equal, No. of Rows (Above
 & Beneath) equal \rightarrow Counting the
 Rows, Col. in a

"Un-Biased" fashion.

c. $g(x,y) \stackrel{(1)}{\rightarrow} Egn(z) \stackrel{(2)}{\rightarrow} \dots$
 Originally $g(u,v) \rightarrow g(-u,-v)$
 Outside $Egn(z)$ $\downarrow \stackrel{(4)}{\rightarrow}$



$g(c_1-u, c_2-v)$

$\downarrow \stackrel{(5)}{\rightarrow}$

$g(x-u, y-v)$

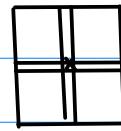


Fig. 1

for u, v

Sept 27 (Tue).

Note: Check Homework on CANVAS.

Due A week from Today.

Oct. 4th.

Midterm will be given After the
 Hand written Student ID Recognition
 Project.

Example: Continued from the Z0
 Convolution Discussion.

openCV / deep-learning-2022s / 2022S-111-#2019S-23-2DConvolution-2019-2-4.pdf

Suppose we have an image $I(x,y)$,

And a Kernel $g(x,y)$ ($K \times K$).

Then, ZD Convolution is defined
 as

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

$$= \iint_{\mathbb{R}^2} I(u,v) g(x-u, y-v) du dv$$

... (1)

For Discrete Cases, we have

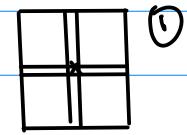
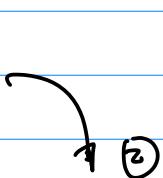
for x, y 

Fig. 2

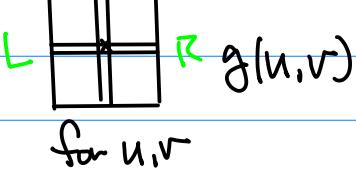
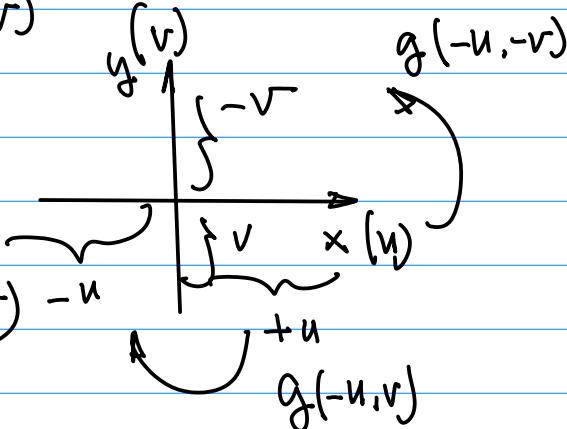
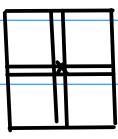
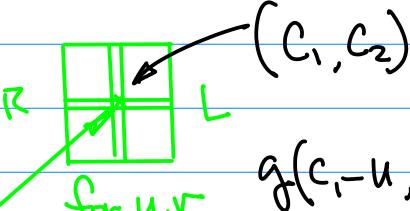
for u, v 

Fig. 3

for u, v

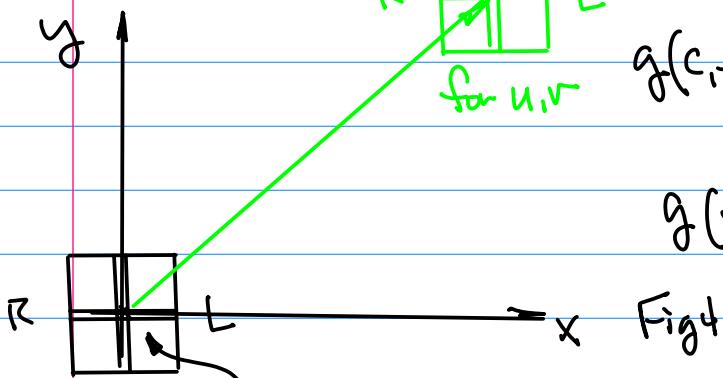
④

for u, v

$$g(c_1-u, c_2-v)$$

Let $c_1 = x, c_2 = y$.

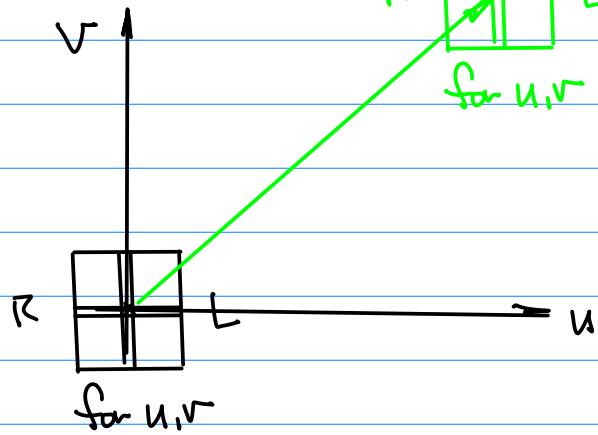
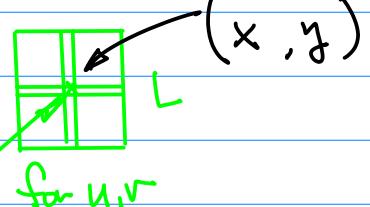
$$g(x-u, y-v)$$

Without "Shifting" e.g.
 $g(-u, -v)$ or

$$g(c_1-u, c_2-v) \Big|_{\substack{c_1=0 \\ c_2=0}}$$

Fig. 4

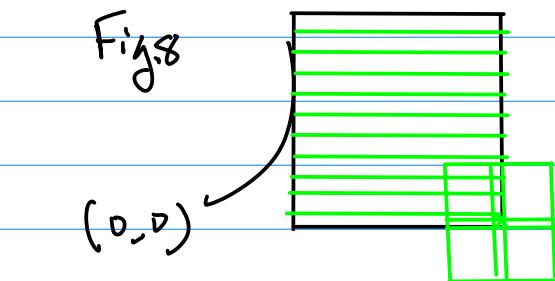
Fig. 5



Note: To perform 2D Convolution defined in Eqn(z) with the kernel $g(x, y)$.

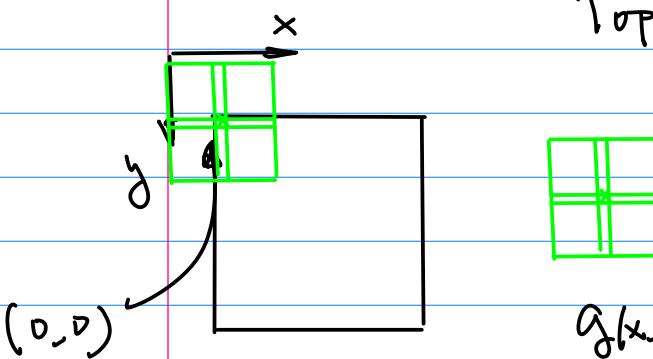
"Slide" the Kernel at $(0, 0)$ of place $I(x, y)$

Top Left Corner.



Note: The Computation of 2D Convolution is defined as

Sum of Products. See Eqn(z)



Initial position of 2D Convolution.

Fig.b

$g(x, y)$
Kernel of
 $K \times K$

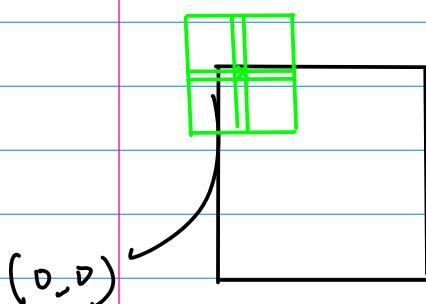


Fig.7

Slide the Kernel
One pixel at Time
from Left to Right

Now, put All these Together.
Compute the convolution By Hand.

$$\sum_{v=0}^{N-1} \sum_{u=0}^{M-1} I(u, v) g(x-u, y-v)$$

Addition

20	20	0	0
20	20	0	0
10	10	0	0
10	10	0	0

$\mathbf{I}(x, y)$

1	0	-1
1	0	-1
1	0	-1

$\mathbf{g}(x, y)$

A Kernel
 $K \times K$

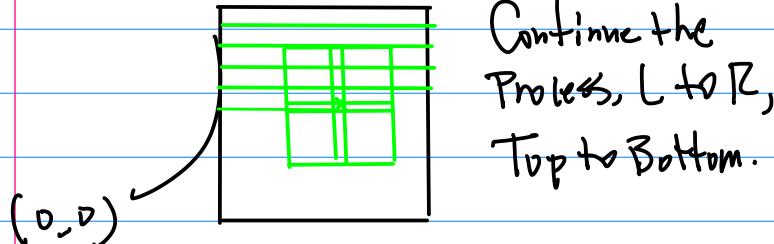


Fig.9

Continue the
Process, L to R,
Top to Bottom.

Step 1. Place $g(x,y)$ at $(0,0)$ on

Kernel

the Image.

1	0	-1	
1	20	20	0
1	20	20	0
1	10	10	0
1	10	10	0

PAD 0's

$$\begin{cases} (0,0) : 0 \times 20 \\ (1,0) : -1 \times 20 \\ (0,1) : 0 \times 20 \\ (1,1) : -1 \times 20 \end{cases}$$

Add Together

$$0 \times 20 + (-1) \times 20 + 0 \times 20 + (-1) \times 20 = 0 - 20 + 0 - 20 = -40$$

Convolution In Progress

40			

Step 2.

1	0	-1	
20	20	1	0
20	20	0	0
10	10	0	0
10	10	0	0

$$\begin{aligned} & 1 \times 20 + 0 \times 0 + (-1) \times 0 \\ & + 1 \times 20 + 0 \times 0 + (-1) \times 0 \\ & + 1 \times 20 + 0 \times 0 + (-1) \times 0 = 40 \end{aligned}$$

Convolution In Progress

40	40	"	"
"	"	"	"
"	"	40	"

$$\begin{aligned} & 1 \times 0 + 0 \times 0 + (-1) \times 0 \\ & + 1 \times 20 + 0 \times 20 + (-1) \times 0 \\ & + 1 \times 20 + 0 \times 20 + (-1) \times 0 = 20 + 20 = 40 \end{aligned}$$

Convolution In Progress

40	40		

Step 3. Carry out the Computation as illustrated, e.g.
Sliding the window from Left to the Right, Top to the Bottom, till finish the entire Image.

Sept. 28.

Note: 1° Brief Update from

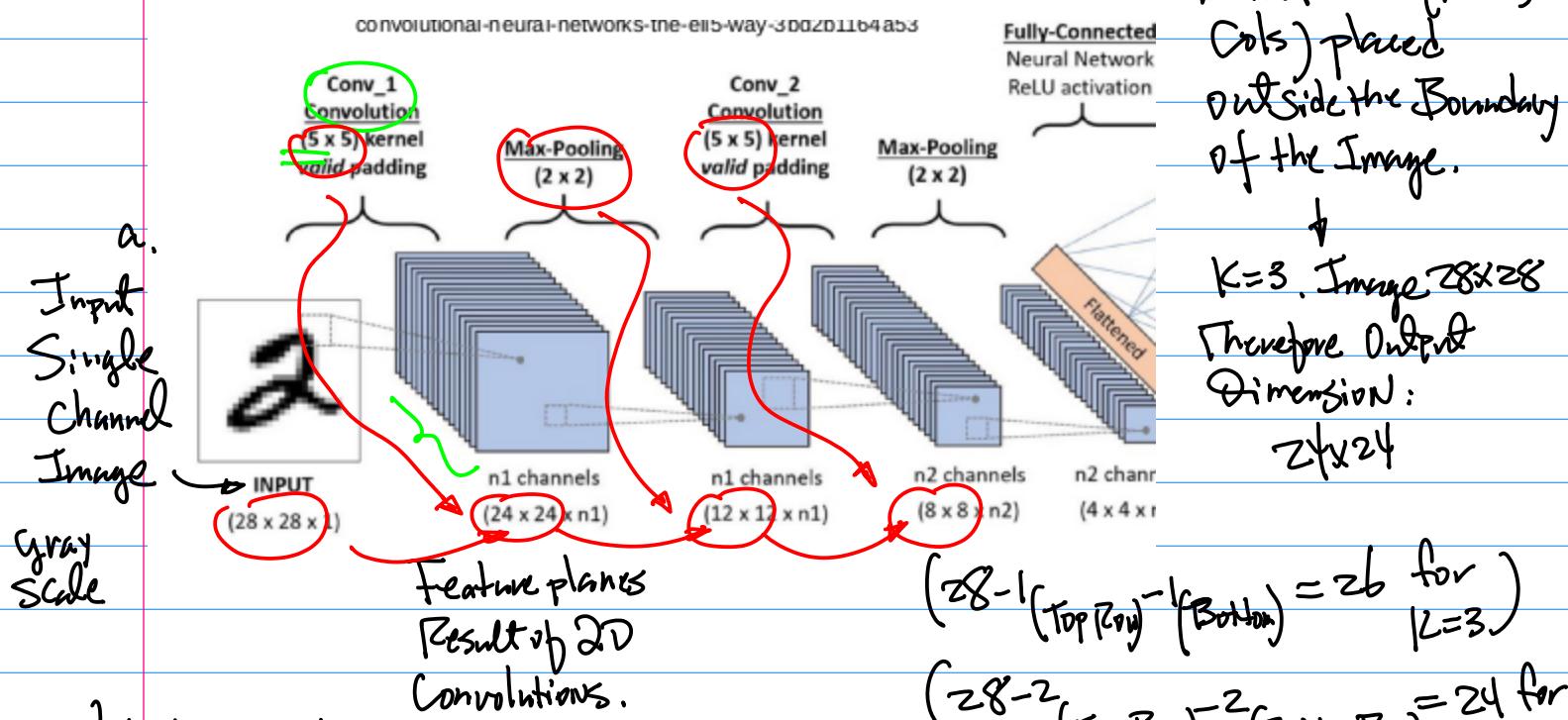
Each team, Status Activities.

On Thursday (A week from today)

2° First Project On DCNN (MNIST)

for Handwritten Student ID
recognition System Design is
due on Oct. 10th.

Example:



Note 1. Resolution

Change. Input: 28×28 Output (Convolution) 24×24

$$(28 - 1)(\text{Top Row} - 1)(\text{Bottom Row}) = 26 \text{ for } L=3$$

$$(28 - 2)(\text{Top Row})^2 (\text{Bottom Row}) = 24 \text{ for } L=5$$

$$\text{Image } M \times N, \text{ Kernel size } K$$

$$M - \left(\frac{K-1}{2}\right) \times 2 = M - (K-1) \text{ Col.}$$

$$N - \left(\frac{K-1}{2}\right) \times 2 = N - (K-1) \text{ Row.}$$

Output ... (1)

$$[M - (K-1)] \times [N - (K-1)]$$

Kernel ($K \times K$) is placed inside
Image Plane. See

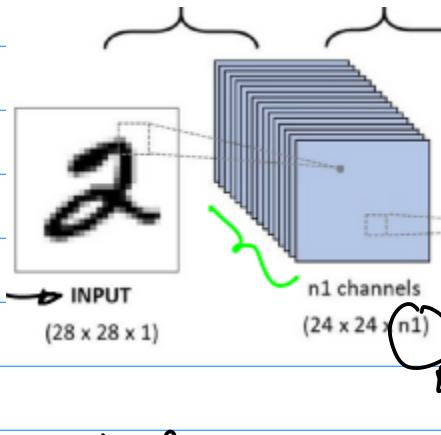
Fig. 1

Reduction of the
Rows (Top &
Bottom), Col.
(Left & Right)

Fig. 1.

in such a manner
None of them (Rows,
Cols) placed
Outside the Boundary
of the Image.↑
 $K=3$. Image 28×28
Therefore Output
Dimension:
 24×24

Example:



Note 2. No. of Output layers is equal to No. of 2D Convolution.

Note 3. Max Pooling

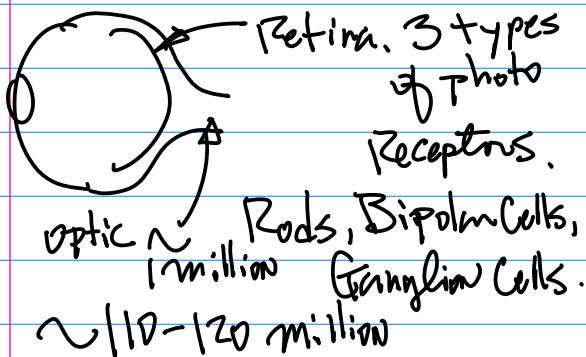
125	220
10	0

pixels

Max Pooling = 220

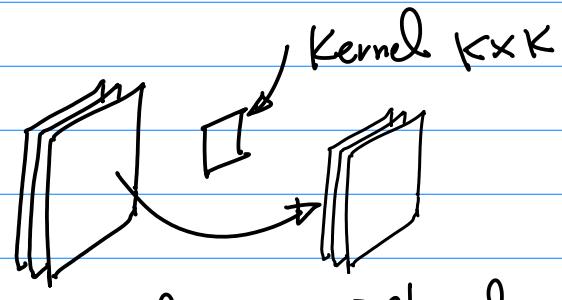
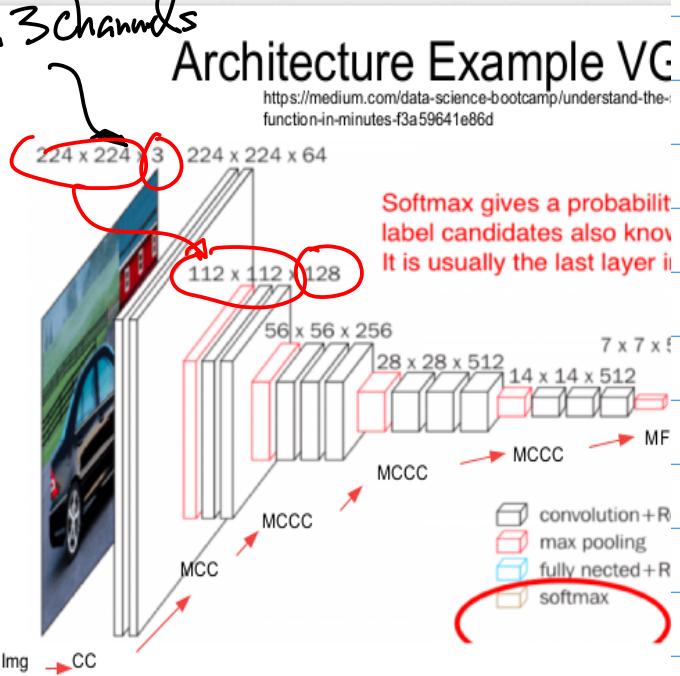
$\text{Max} \{ 125, 220, 10, 0 \}$

Biological Inspiration from Human Visual Perception System, Early Vision.



Change 2D Convolution "Step size" Stride to Penalize Both Function Convolution & Pooling.

a. 3 channels



Optional (Not Required.)

Example:



Feature Extraction
"Edge" Map.

Boundary of the object.
Build A Kernel for the purpose of Extraction of Boundaries .

Object Boundaries \rightarrow Sudden Change of Image Intensity.

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

Partial Derivative \hookrightarrow How to Detect Abrupt Change of the Intensity,

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

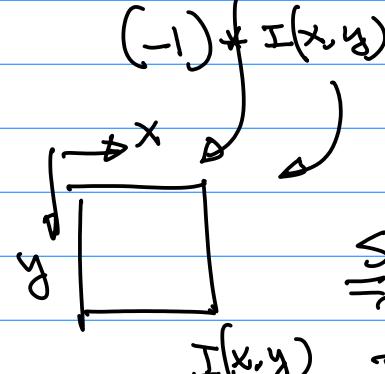
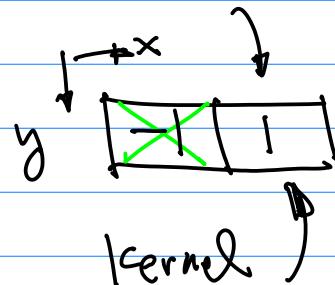
Similarly for y .

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

From Eqn(z), on Digital Image $\Delta x = 1$

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

$$\approx I(x+1, y) - I(x, y) = 1 * I(x+1, y) + (-1) * I(x, y)$$

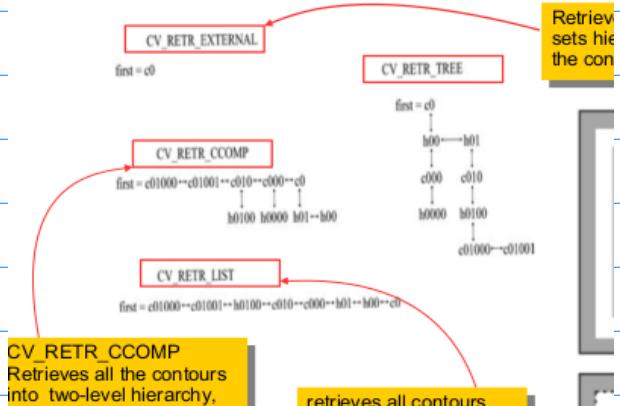


$\text{LoG}(x, y)$ Kernel, Laplace of Gaussian Kernel.

Now, consider Contour Analysis as Pre-processing Technique

Contours are Boundaries of the Objects, In OpenCV. They are Lists.

4 Contours N



Oct. 4th (Tue).

Handwritten Digits Recognition.

MNIST. Ref. from the Class

github

2022S-110-#lec5-1-mnist-hl-2021-3-1.pdf

2022S-110a-MNIST-digits-saveTrained.py

Background:

Step1. OpenCV \rightarrow Step2

Python Program.

To Capture Web CAM

Video

Preprocessing Task.

Goal: To Build

Bounding Boxes

for the Digits.

One Blank Printer

Paper + Marker

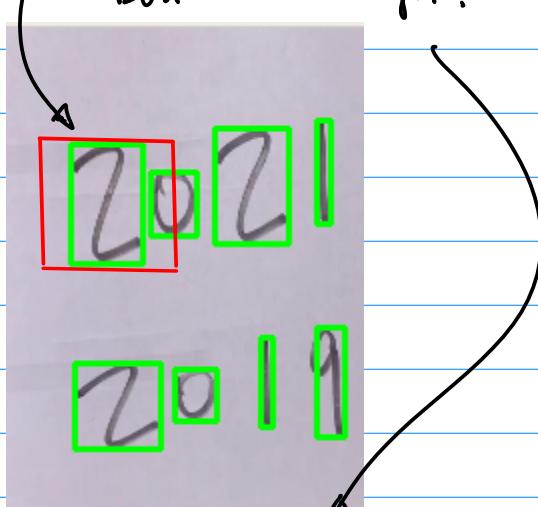
Write 4 Digits of yours

(One Box for One digit)

SID

Step3. (Also Belongs to Preprocessing) Make Each Bounding Box to Become

- ① A Squared Small Image by the Bounding Box for Each Digit.



Without Distortion (e.g. without Changing the Aspect Ratio of the Image)

- ② To prepare the Squared Image to the right size, and Right format for MNIST Input.

1 Channel
 $\underbrace{28 \times 28}_{28 \times 28 \times 1}$

Step4. To Run the MNIST Sample Code. (Sample Code from google TIF Tutorial, or from the Class Recommended, or the

the Sample code from this class's github).

- ① The Sample code will get the Dataset for MNIST Convolutional Neural Network
- Read the Dataset → Collection of Small Image Annotation file
- ② Will then Split the Dataset into Training set & Testing set. as well as its architecture

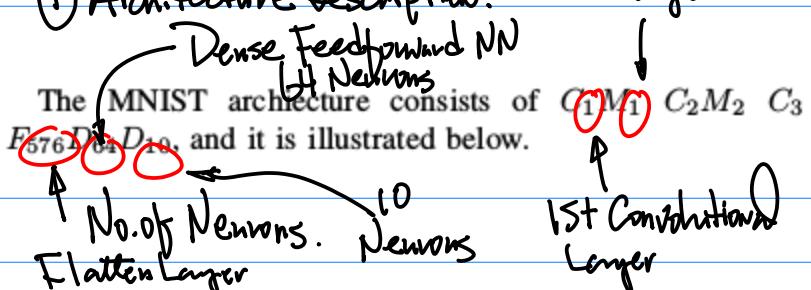
Step5. Save the trained DCNN. e.g. all Weights & Biases Once trained, are Saved into A file for the future use.

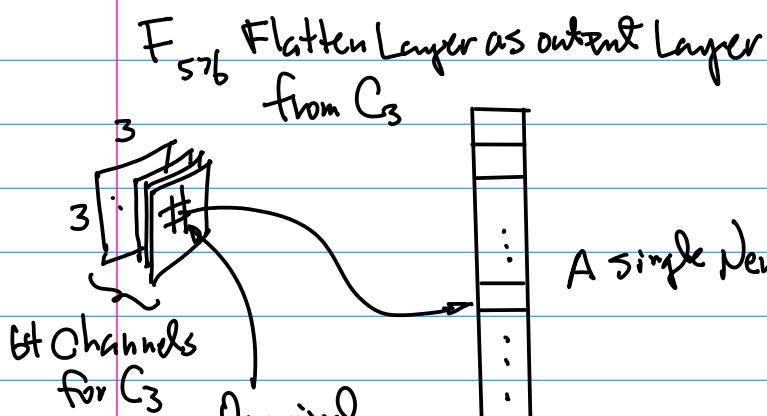
Step6. For Deployment purpose, Modify MNIST. (See Sample code), No Need to Download Dataset for Training, just Load Saved trained Result Back to the Defined Architecture.

Step7. Connect OpenCV Preprocessed Images (from Live Video, Web Cam) to the MNIST Code.

Example: On MNIST.

- ① Architecture Description.





Video Source from A Web CAM.

the 3rd part of the Homework:

Hand Calculation of (1) Binary Images. 1/2

A single Neuron. (2) 2D convolutions.

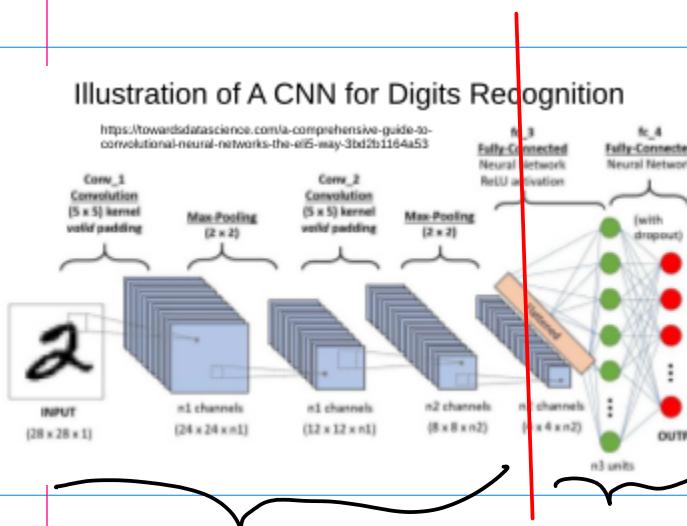
Team Project Quick Status checking,
Will continue on Thursday.

Fig.1

Oct.6 (Thu)

Note: 1° Project Announcement for
S19 Handwritten Digits Recognition
System, due Oct. 2nd, official
Announcement will be posted on CANVAS
Before End of the Day Friday;
2° Midterm after the project.

Example: DN MNIST.



Feature Extraction

Decision-Making

Note: N₃. of Neurons @ the Flatten Layer = 576



No. of the pixels @ C_3 Layer : $3 \times 3 \times 64$

Resolution

Channels / Layer
 $= 64$

Homework Due A week

from Today, Oct. 11th.
(Tuesday)

2 parts. 1st Part: OpenCV Programming
Write A program to Capture Display Live

a.

Algorithm 1 Build MNIST convolutional NN

Require: import keras
 from keras import layers
 from keras import model

Ensure: Build MNIST convolutional NN layers sequentially

```
model = models.Sequential()
model.add(layers.Conv2D(32,(3,3), activation='relu', input_shape=(28,28,1)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64,(3,3),activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64,(3,3),activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64,activation='relu'))
model.add(layers.Dense(10,activation='softmax'))
```

No. of the Neurons

Note: Eqn (1) "ReLU" is one of the very popular Activation Functions.

(4) Activation Function Softmax.

$$f(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad \dots (2)$$

Where $\sum_{j=1}^K$ No. of Classes/Cats
 in the context of Handwritten Digits Recognition: $K=10$
 $(0, 1, 2, \dots, 9)$

$$e^{z_j}, \text{ for } j=1, 2, \dots, 10$$

we have

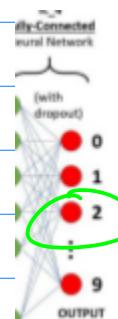
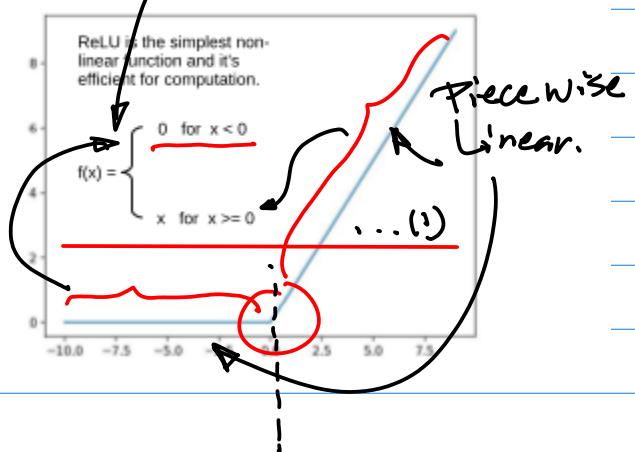
$$\sum_{j=1}^{10} e^{z_j} = e^{z_1} + e^{z_2} + \dots + e^{z_i} + \dots + e^{z_{10}}$$

Note, z_i is the output from the Neuron at Output layer, for $i=3$

① `models.Sequential`
 ② `model.add()`

`layers.Conv2D`
`layers.MaxPooling2D`
`layers.Flatten`
`layers.Dense`

③ Activation function.
 "ReLU", Math. Formulation

Rectified Linear Activation Function

Want to produce Output whose value can be formulated/integrated as probability value.

$$0 \leq \text{Prob}(x) \leq 1 \quad \dots (3a)$$

$$\int \text{Prob}(x) dx = 1 \quad \dots (3b)$$

$$\text{Now, } e^{z_1} + \dots + e^{z_{10}}$$

$$\frac{e^{z_i}}{e^{z_1} + \dots + e^{z_{10}}}$$

$$0 \leq \frac{e^{z_i}}{e^{z_1} + \dots + e^{z_{10}}} \leq 1$$

(match up to
Eqn 3a)

$$\frac{e^{z_1}}{e^{z_1} + \dots + e^{z_{10}}} + \frac{e^{z_2}}{e^{z_1} + \dots + e^{z_{10}}} + \dots +$$

$$\frac{e^{z_1}}{e^{z_1} + \dots + e^{z_{10}}} + \dots + \frac{e^{z_{10}}}{e^{z_1} + \dots + e^{z_{10}}}$$

$$= \frac{e^{z_1}}{e^{z_1} + \dots + e^{z_{10}}} = 1, \text{ Satisfies Eqn 3b}$$

Note: TensorFlow Keras Based
MNIST Sample Code.

- 2022F-105a-#6mnist-numerals-ch02 (copy...)
- 2022F-105b-#6mnist-numerals-ch02.py
- 2022F-105c-#load-deployment-MNIST.py

(5) Model.Save()

105a: \rightarrow 105b Saves the Trained Result (e.g. Both Architectural Information,
Baseline Code all weights/Biases)
For Training

```
-----save trained model-----
import h5py
model.save('harryTest.h5')
#-end
```

↓
105c.

Note: Save your Result in the following
Naming Convention:
"FirstName_LastName-SID(4 Digits)-P1.h5"

Deployment.

① Will visit this topics later.

Oct. 11, 22

1. The optimizer is "rmsprop". RMSprop is a gradient based optimization technique used in training neural networks. This technique balances the step size, e.g., momentum, which makes the step size decrease for large gradients to avoid exploding, and increase the step for small gradients to avoid vanishing. So here is the background of it. "RMSprop is unpublished optimization algorithm designed for neural networks, first proposed by Geoff Hinton in lecture 6 of the online course Neural Networks for Machine Learning. RMSprop lies in the realm of adaptive learning rate methods, which have been growing in popularity in recent years, but also getting some criticism[6]. Its famous for not being published, yet being very well-known; most deep learning framework include the implementation of it out of the box" [Bushaev, 2018].

2. The loss function or the objective function is "categorical_crossentropy", which minimizes the loss function based on the probability of the likelihood of the predicted output class.

4 Different Types Contours Implementation in OpenCV.

② Will visit this later, Oct. 11, 22

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

Motivation: Build Bounding Boxes → ROI (Region of Interest)

To Crop the Image
→
Further Processing

OpenCV Implementation for Contour Analysis.

4 Contours Models

Dr. EXTERNAL



CV_RETR_EXTERNAL

first = c0

CV_RETR_TREE

first = c0

b

TREE

h00

h01

c000

c010

c000

c000

c0100

c010

Oct 11, 22.

Note: 1° Whitepaper on MNIST.

2022S-110-#lec5-1-mnist-hl-2021-3-1.pdf — lec5-1-mnist-hl-2021-3-1

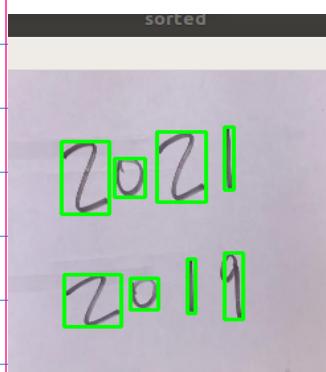
2° Midterm Exam Oct. 27

Thursday.

3° Two Topics in the white paper
to Be visited later.Example: Continuation ON Contour
Analysis. PPT. (ON github)

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

Motivation: Using Preprocessing Techniques

We have covered in
the class.Find Bounding Boxes
for the Digits, see
Green Boxes in Fig.1.Fig.1. We will Need Contour Analysis
Technique.

From the Sample Code for Contour Analysis.

1. Line 24, Resize function Can Change the

```

23
24 img = cv2.imread(image, cv2.IMREAD_COLOR)
25 img = cv2.resize(img, (256, 256))
26 cv2.imshow('original image', img)

```

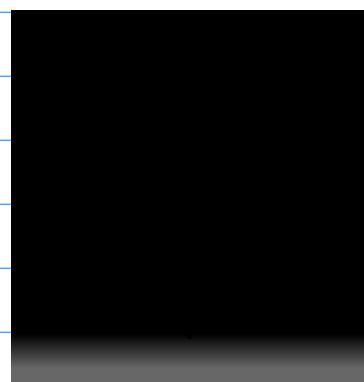


Fig.3a

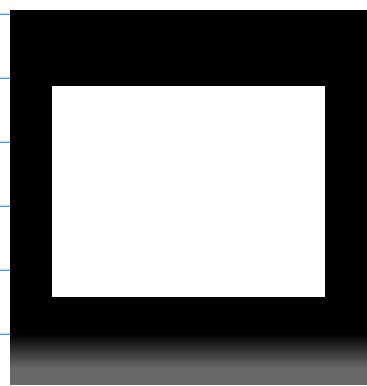
Add An
Object
(White)

Fig.3b

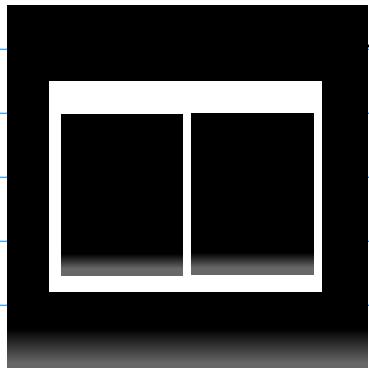
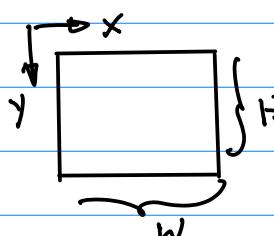
White Obj
has 2
Black "holes"

Fig.3c

Add 2
Foreground
White
Squares.

Fig.2



$$\text{Aspect Ratio} = \frac{W}{H} / \frac{W+H}{W+H} \dots (1)$$

When $W=H=256$ in Line 24,
it changes the A.R.

Aspect Ratio of An Image. where

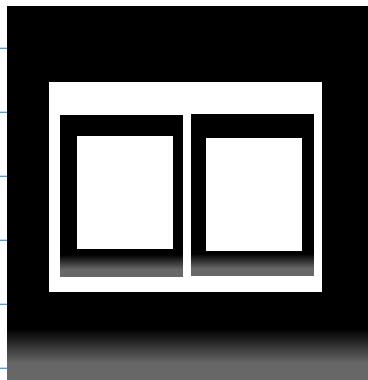


Fig.3d

Add 2 Black
Squares As
Background

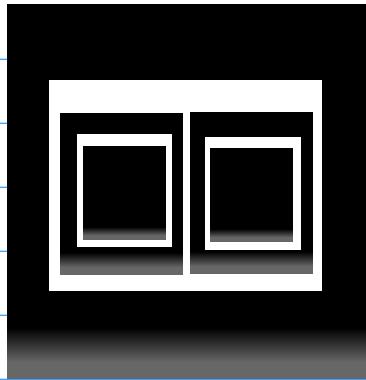
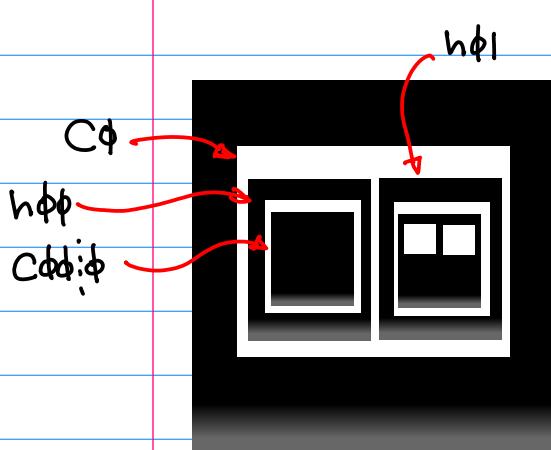


Fig.3e



Add 2 Smaller
White Squares
to the Right Blank
Square

Fig.3f

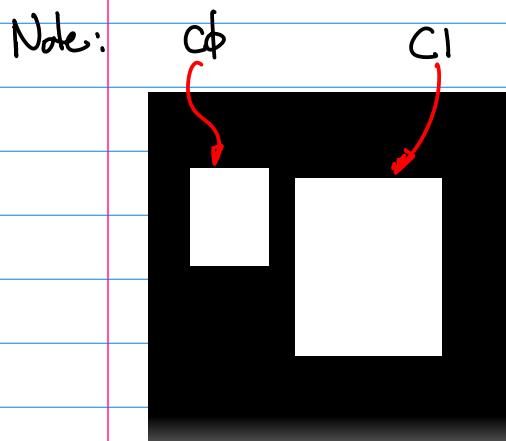
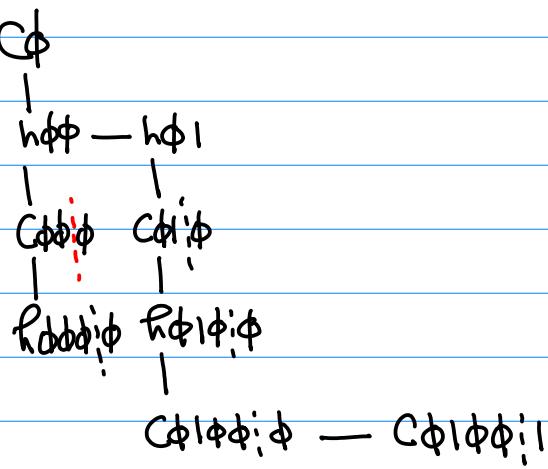


Fig.3g



External Contour

```

30     thresh = cv2.Canny(imggray, 100, 200)
31     cv2.imshow('Canny', thresh)
32
33 contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
34
35 # Draw contour
36 thresh1 = cv2.cvtColor(thresh, cv2.COLOR_GRAY2BGR)
37 cv2.drawContours(thresh1, contours, -1, (0, 255, 0), 5) # all contours
38 cv2.imshow('All contours', thresh1)
#Bounding box

```

From PPT, pp.8

Note: Use Size/Area of A Contour
to filter Out Random Noise

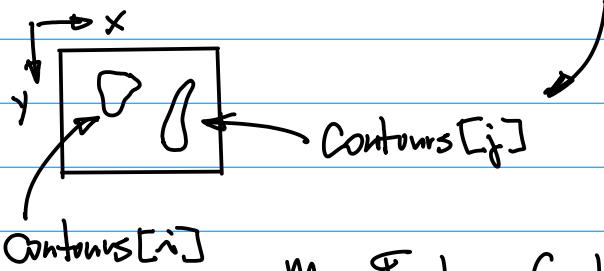
1. Moments

```

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

```

To find moments of
Any Contours, i or j

**2. Contour Area**

area = cv2.contourArea(cnt)

Set Threshold T = 10 (pixels)
if area[i] < T
then area[i] is Random Noise

Many Features Can be
derived from the moments.

