CMPE258
Fall 2023

August 22 (Tue)

Organizational meeting.

1. Class material on github
github/hualili

aiv100	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-NumeralD...	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 Engine
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 (HI): (Human Intelligence)
is Symbolic Representation of
Learnt Experience.

CMPE258

Fall 2023

2

4. Projects. 2
plus 1 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.

This course ~~is an online course~~ NO. The students must have Internet connectivity and Zoo
his/her machine. The students must participate in the class activities and submit all as
exams to SJSU CANVAS. The syllabus, faculty contact information on the syllabus, l
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%

Note:
1° CANVAS To Be up by the end of the day, Friday;
2° Tools & Software To be installed (will provide "Readme" as Ref)
OpenCV
Python.
T.F. Version 2.0 or higher

7. Textbooks & References

Textbook

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

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

8. Software Tools & Dev. Environment

Python. Pycharm.
AnnoConda.
TensorFlow.

Note: Rm268 Available ~~Per~~ ON Approved Basis.

August 24 (Thursday)

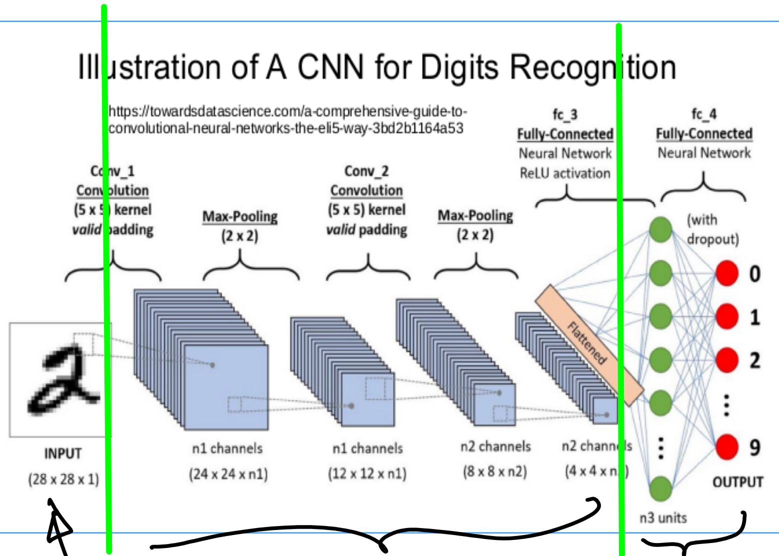
Ref: github.

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

Note: Lab Space for the Class Rm268.

Example: Architecture Overview.

Note 1.



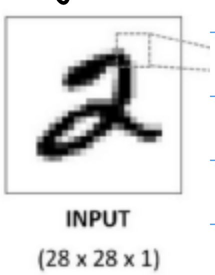
Preprocessing Computer Vision.

Convolution Layers
Feature Extraction
To produce Feature Vectors.

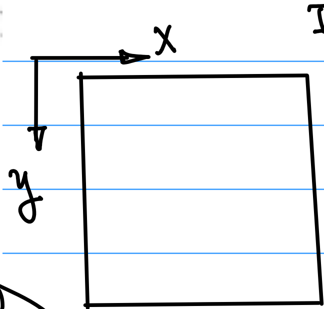
$\vec{u} = (u_1, u_2, \cdots, u_N)$

Feed Forward Neural Networks.
Decision Making

2. Image Definition

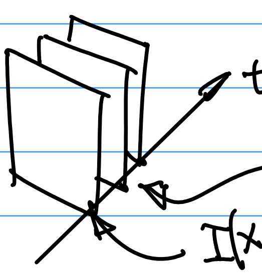


M N Channel



$I(x, y; t)$

Note: for x & y definition see Ref. Next page.



$I(x, y; t+\Delta t)$

$I(x, y; t)$

M x N

No. of col. Per Row.

No. of Rows per frame

Ref: on the github.

2023S-101-Note-cmpe258-2023-03-16.pdf

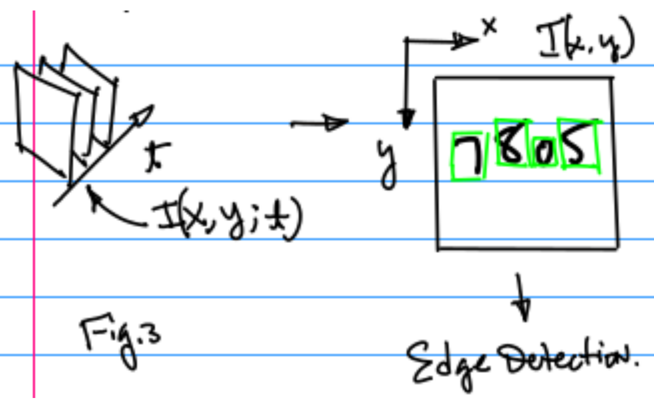


Fig.3

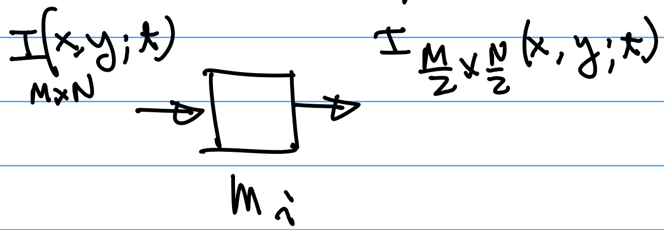
3. For the Convolutional Layer, we denote it as C_i, where $i = 1, 2, \cdots, N$

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

so, for the example, we have

$C_1 M_1 \to C_2 M_2$

Maxpooling for Resolution Reduction / Feature Reduction.



Note: BioLogic Inspiration, Human Visual Perception System, Retina, ~ 110 million photo Receptors

Comparison to Web Cam.

1080p Resolution: 1920x1080

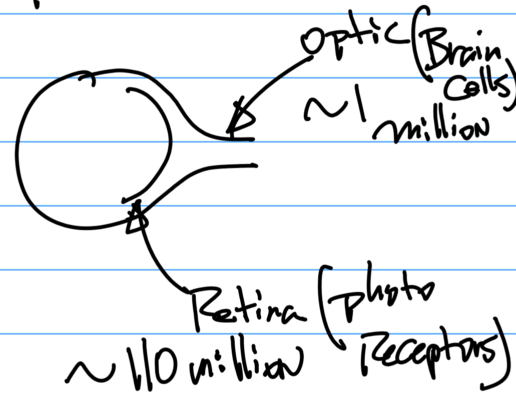
$1920 \to \sim 2^{11} = 2 \cdot 2^{10} = 2K$

$1080 \to \sim 2^{11}$

$\underbrace{2^{11} \times 2^{11}}$

$2^{22} = 2^2 \cdot 2^{20} = 4$ Meg

$\simeq 4$ million.



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

M: No. of Neurons / Nodes

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

CMPE258

F2023.

August 29 (Tue)

Note: 1° CANVAS is up.

2° Homework Assignment

a. Honesty pledge to Be Signed / Signed Copy has to be uploaded to CANVAS.

b. Software Tools Installation. (Opt)

① OpenCV. By Friday

Next Tuesday. Bring Your Laptop w/ OpenCV installed.

↓

Please use Smart phone 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° Form A Team for the Semester Long Project.

Example: Consider 2D Convolution Technique.

Ref: github/hualili/opencv

↓

2022

Note 1:

$$c(n_1, n_2) = \boxed{\sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty}} a(k_1, k_2) b(n_1 - k_1, n_2 - k_2) \quad \text{...(1)}$$

Image Kernel

$n_1 = x$, $n_2 = y$

Summation Index: k_1, k_2 are for x and y.

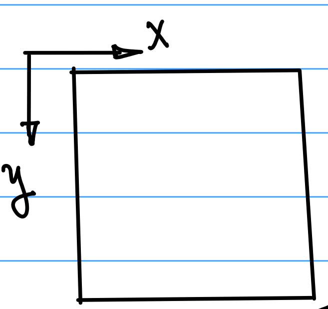


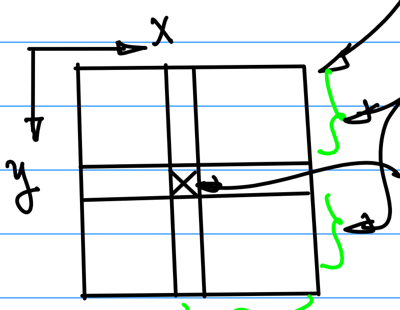
Fig. 1

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 $K \times K$



K(x,y)

The No. of Rows Above the P.O.I. = No. of Rows Beneath the P.O.I.

Pixel of Interests.

K x K

Fig.2

K = 3, 5, 7, ... (odd Number)

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

I(x,y)

0	0	100	100
0	0	100	100
0	0	100	100
0	0	100	100

4x4

Fig.3a

8 bit Grayscale: $2^8 - 1 = 255$

-1	0	+1
-1	0	+1
-1	0	+1

Fig.3b

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

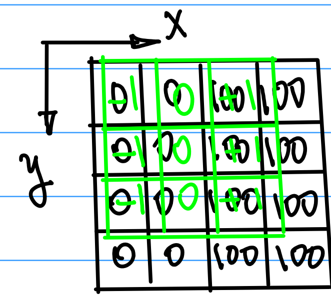


Fig.3c

$-1 \times 0 + 0 \times 0 + 1 \times 100 +$
$-1 \times 0 + 0 \times 0 + 1 \times 100 +$
$-1 \times 0 + 0 \times 0 + 1 \times 100$
$= 300$

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

	300		

Fig.3d

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

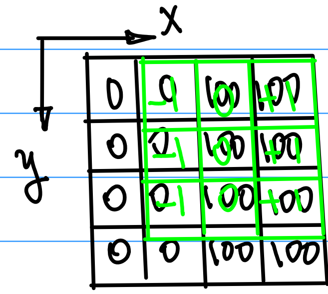


Fig.3e

$-1 \times 0 + 0 \times 100 + 1 \times 100 +$
$-1 \times 0 + 0 \times 100 + 1 \times 100 +$
$-1 \times 0 + 0 \times 100 + 1 \times 100 = 300$

CmpE258

F2023

August 31 (Th).

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

2° Will post Homework ON CANVAS 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

Example: Theoretical Aspects for 2D Convolution.

Ref: 1° PPT on the github.

2° 2023S-100-note-... 2023-03-21.pdf

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

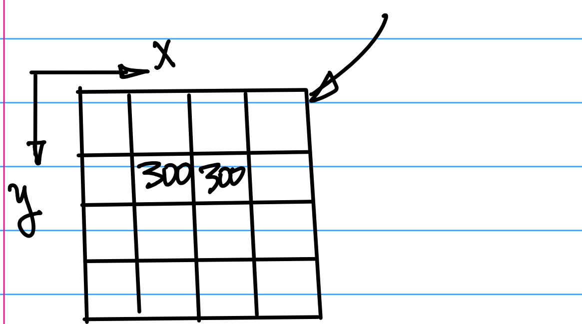


Fig.3F

Step3. $I(x,y)*K(x,y)$

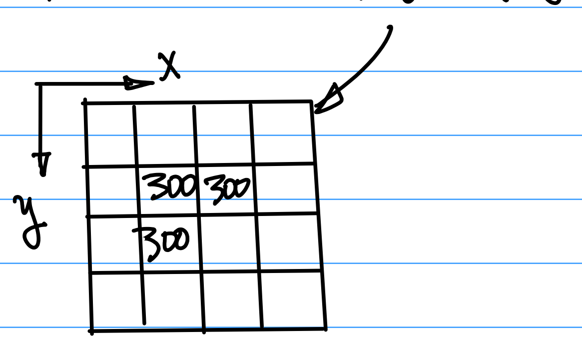


Fig.3G

Step3. $I(x,y)*K(x,y)$

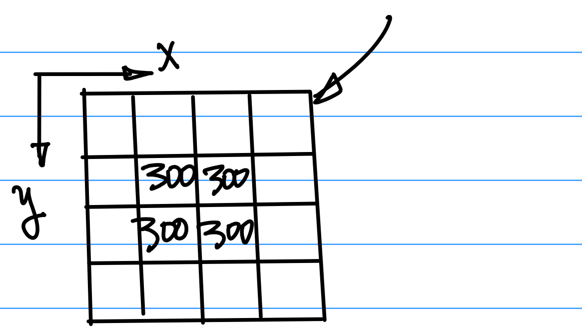


Fig.3H

Note: 2D Convolution (Continuous Case)

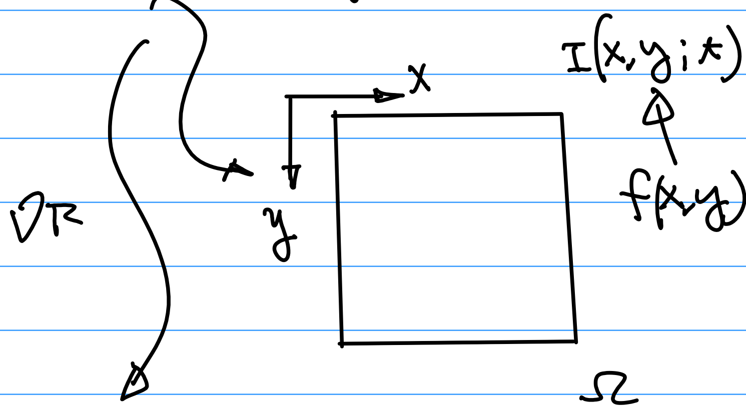
given Image $f(x,y)$
a Kernel $h(x,y)$

$$\iint_{\Omega} f(u,v)\, h(x-u, y-v)\, du\, dv \quad \cdots (2)$$

$$\iint_{\Omega} \underline{f(u,v)}\, \underline{h(x-u,y-v)}\, du\, dv \quad \cdots (1)$$

Image · Kernel

Note 1. Ω : Image Plane(s)



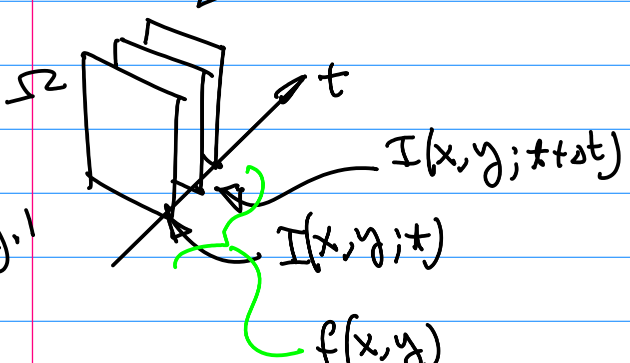
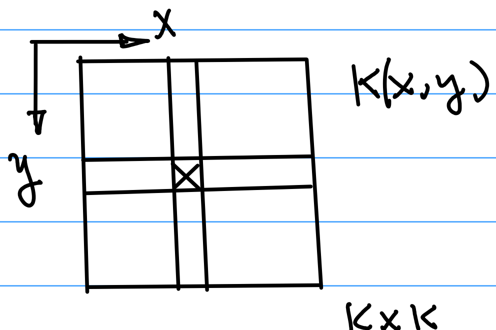


Fig.1

Note 2: Kernel



K×K

Fig.2a

↓ (1) $K(-x,y)$

Flip the Kernel w.r.t y-axis

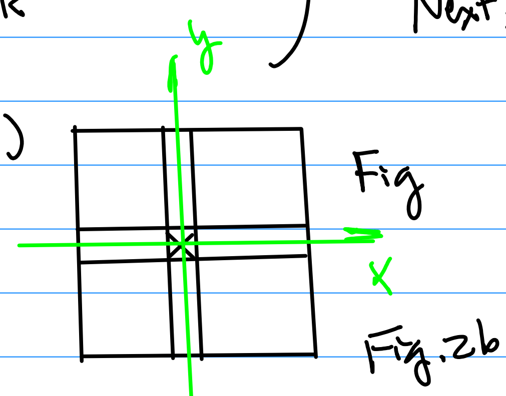


Fig.2b

(2) Flip the Kernel w.r.t. x-axis.
(since $K(-x,y)$ changed to $K(-x,-y)$)

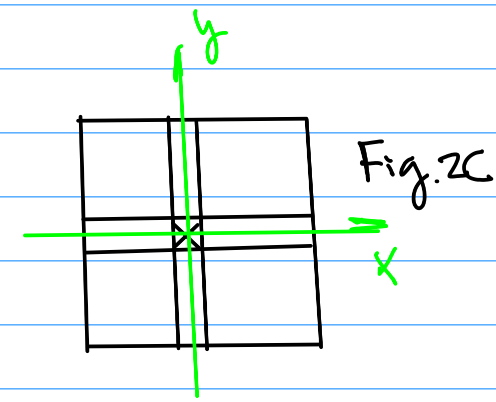
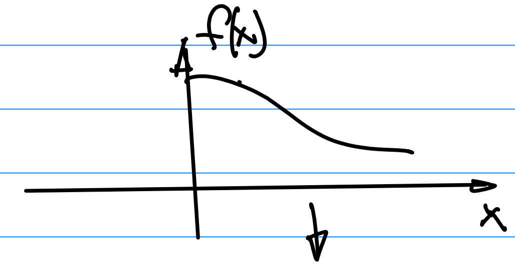
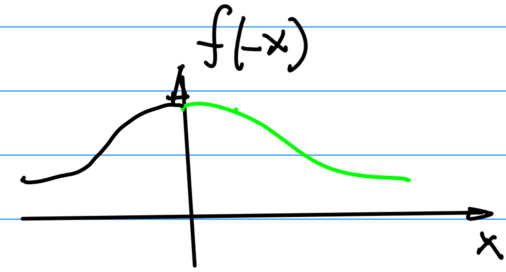


Fig.2C

Note: for 1D Case





Next, make $K(-x,-y)$ to $K(x_0-x, y_0-y)$

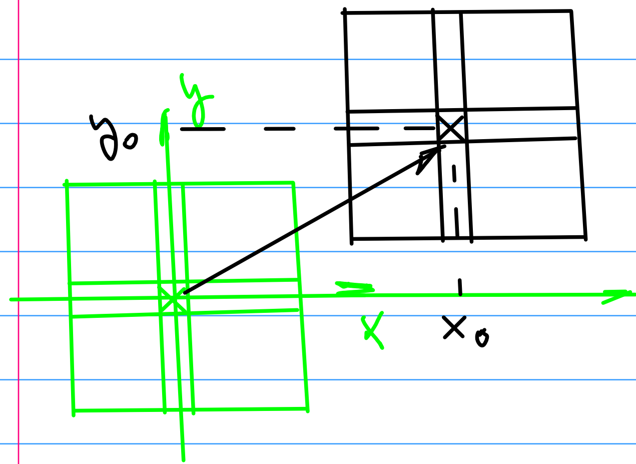


Fig. 2d

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.

Therefore,

Input Dimension $M \times N$ → Output After Convolution $M-(K-1) \times N-(K-1)$ ... (3)

Now, for Discrete/Digital Image (OR Feature plane)

Replace $\iint$ → by $\sum_{k_1} \sum_{k_2}$

$$c(n_1, n_2) = \boxed{\sum_{k_1=-\infty}^{\infty}} \boxed{\sum_{k_2=-\infty}^{\infty}} a(k_1, k_2) b(n_1 - k_1, n_2 - k_2) \quad \text{... (2)}$$

Image Kernel

Dimension Change Due to the Convolution.

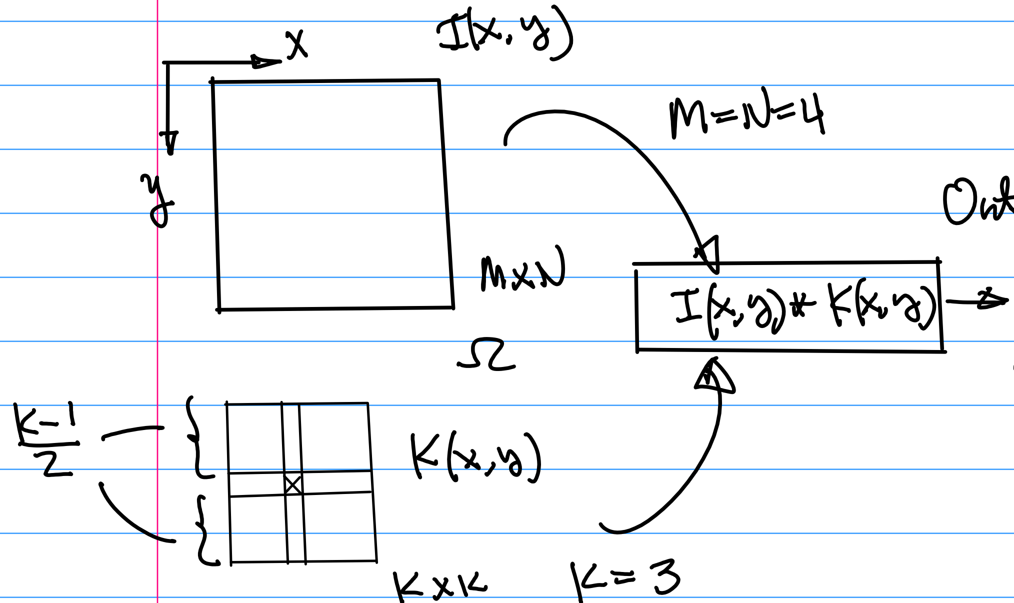
Sept. 5 (Tue)

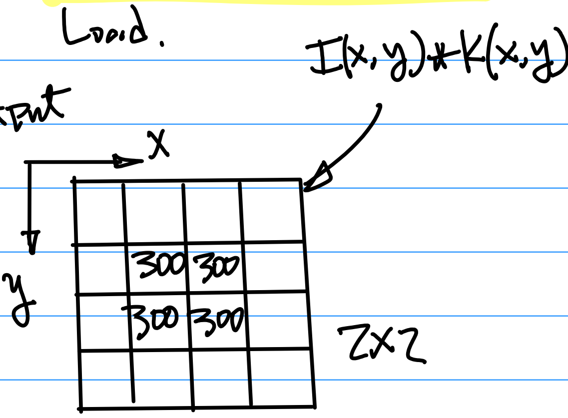
Note: 1° CANVAS OpenCV, Anaconda Installation;

2° 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 Hand written Digits Recognition.
Based on Eq(3), pp.9.

$$M \times N \text{ Image} \xrightarrow[\text{Convolution}]{K \times K} M-(K-1) \times N-(K-1)$$

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

Now, the total No. of feature layers $= n1$

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

Therefore
$n1$ Convolutional feature Layes is the result of $n1$ Convolutions, e.g. $n1$ different Kernels.

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

LoG Laplace of Gaussian.
Gabor Operators

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \quad \dots (2)$$

$$G(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{\cancel{2\sigma^2}\, 2(\sigma_x^2+\sigma_y^2)}} \quad \dots (1)$$

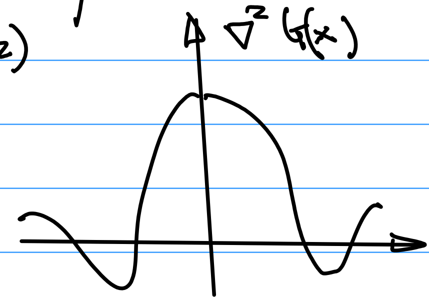


Fig 1.

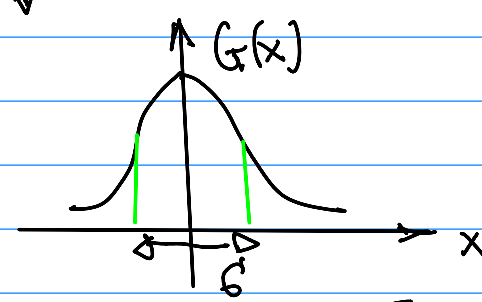


Fig 2

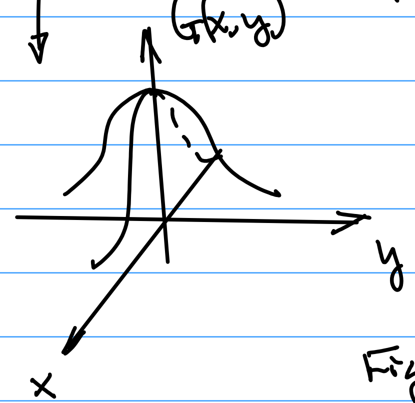
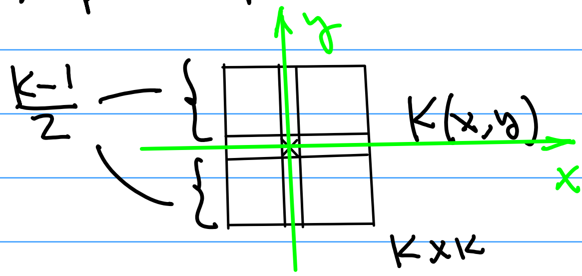


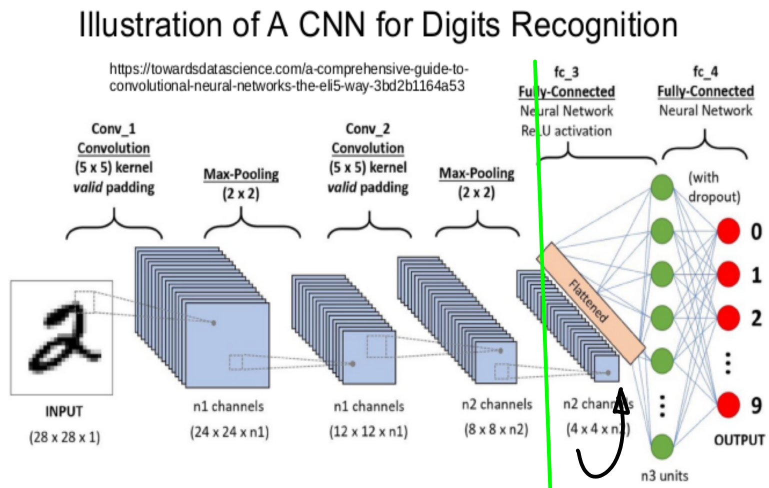
Fig 3

Take G(x,y) in Fig.3, Map it to K×K.



n1 Kernels Produce n1 Layers.

Now, consider the Architecture of MNIST



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

Now, consider a technique for feature Reduction,

Given an image I(x,y)

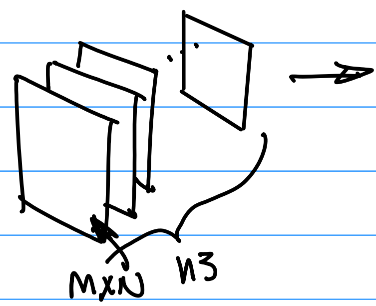
0	0	100	100
0	0	100	100
0	0	100	100
0	0	100	100

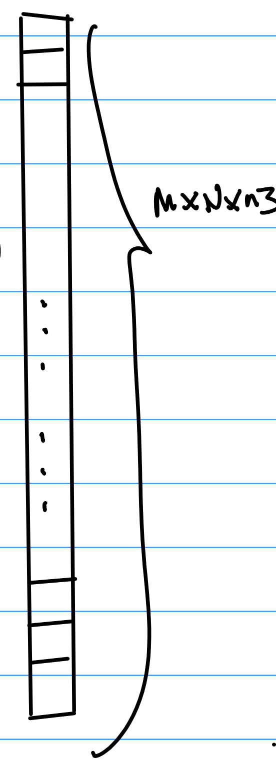
Feature Layer

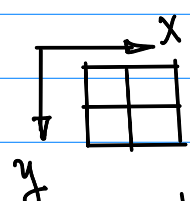
maxpooling →

0	100
0	100

Max{0,0,0,0}







K×K where K=2

Example: Sample code OpenCV for Displaying an image

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

Sept. 7 (Th).

Note: 1° About OpenCV Reference/Sample Code.

a) On github.

opencv / deep-learning-2022s / 2022S-109c-v2-localization-contours.py

hualili Add files via upload

for Video Capture 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)

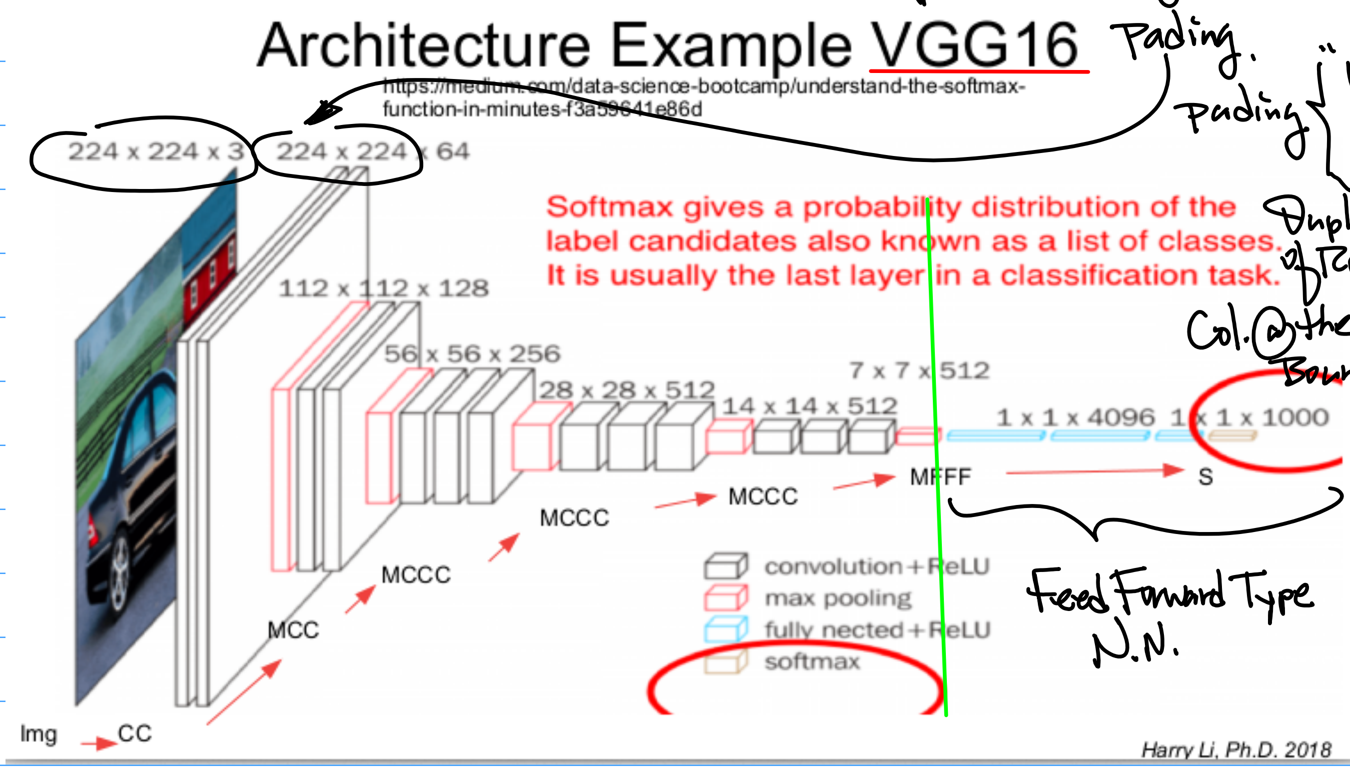
Submission on CANVAS.

Example: Convolutional Layer, Architecture Analysis.

Note 1. Architecture; Note 2: Pading v.s. No Pading.

"ϕ"s pading

Duplication of Const + Col. @ the Boundary

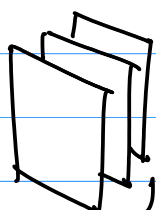


Feed Forward Type N.N.

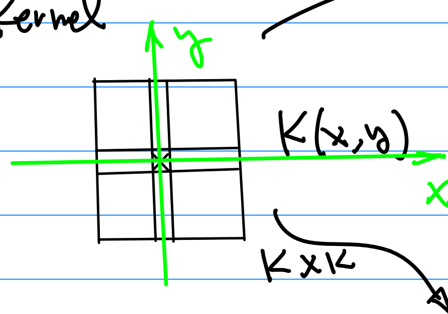
Note 3: Convolution Discussion.

One Example

Kernel



Colour Image



	x		
y	-1	0	+1
	-1	0	+1
	-1	0	+1

LoG $\nabla^2 G(x,y) \rightarrow$ PP.10.

$$\nabla^2 G(x,y; \mu_x, \mu_y, \delta_x, \delta_y)$$

$$\rightarrow \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) G(x, y; \mu_x, \mu_y, \delta_x, \delta_y)$$

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

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

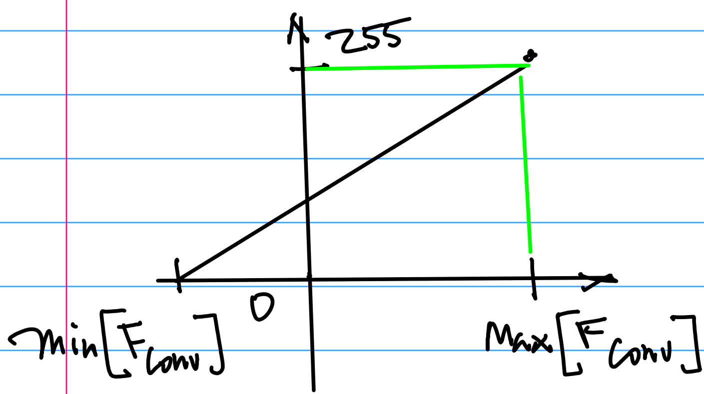
First, Kernel is one channel in this Case;

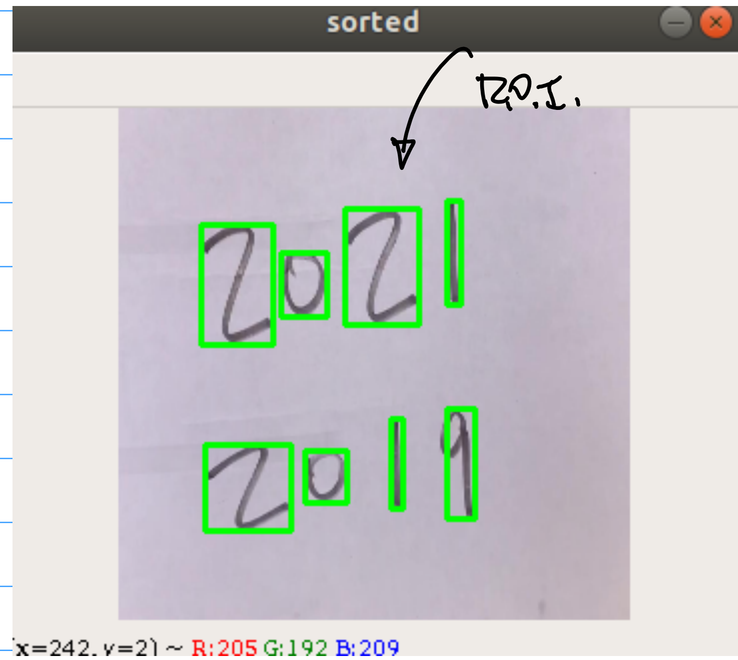
Then, Convolutions. $r(x,y) * K(x,y)$, $g(x,y) * K(x,y)$, $b(x,y) * K(x,y)$

CmpE258

F2023.

$$F_{conv}(x,y) = \frac{1}{3}\left[r(x,y) * K(x,y) + g(x,y) * K(x,y) + b(x,y) * K(x,y)\right]$$





Region of Interests.

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

Requirements:

1° Live CAM Input,
720P or 1080P
(1280x720 (MxN))
Col. Row.
1920x1080
(MxN)
Col. Row.

2° Printer Paper (Blank/White)
Black mark to Writ 4 Digits

3° Localize R.O.I. ON Each Digit

Sept. 12 (Tue).

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

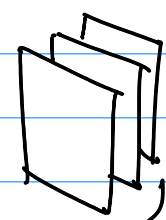
Note 2. Team Project.
Ref: 2023F-103-project-team-2023-9-12.pdf
ON CANVAS as well.
Due on Nov. 27 (Monday) 11:59 P.M.

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

$$\frac{1}{6}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$$

3x3.

A color Image $I(x,y)$



Colour Image

Find Convolution Result.

$$\begin{bmatrix} 0 & 0 & 0 & 255 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \\ 0 & 0 & 5 & 100 \\ 0 & 0 & 2 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 255 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 255 & 0 \end{bmatrix}$$

$I_r(x,y)$ $\qquad$ $I_g(x,y)$ $\qquad$ $I_b(x,y)$

$$I_r(x,y) * \frac{1}{6}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & -255 \\ 0 & 1 \end{bmatrix}$$

$$I_g(x,y) * \frac{1}{6}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} \rightarrow \begin{bmatrix} -95 & 5 \\ 2 & -98 \end{bmatrix}$$

$$I_b(x,y) * \frac{1}{6}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} \rightarrow \begin{bmatrix} -255 & -255 \\ 255 & 255 \end{bmatrix}$$

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° Installation of T.F., Screen Capture the installation Result.
(w/ Personal ID).

2° OpenCV Code to Handle
a. Live Web CAM Input Video, and Display it;
b. file input, MPEG4 format Video and display it.

Note: Please Use A Printer paper, Write with a Black marker, 4 Digits of your SID.

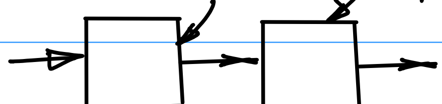
Submission: the Code & Video Clips.

Preprocessing for Hand Written SID Recognition System Design.

Live Web CAM Input

Output (I)

Output stage II

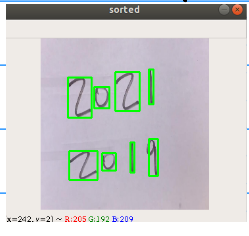


Make I(x,y) to match the Resolution & Channel Requirements of the DCNN Engine Input (e.g. 28x28x1)

Processed

without Distortion!

Localized R.O.I.



Region of Interests

CMPE258

F2023

Ref:

Homework: Due 1 week from today. Sept. 21st.

Use your OpenCV for the 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 $B(x,y)$.

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 → Gray Scale Image.

C. Filtering Operation, Conducted by Convolution w/ Predefined Kernel Coefficient.

B. To Be Continued.

Edge Detection CANNY, LoG, Gabor

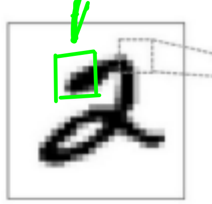
Note: Input Should be a grayscale (8 bits), Output Should be 8 bit (1 Channel)

255		
	200	
		11

A.

Binarization. To Obtain A Representation of A Digit.

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



INPUT

(28 x 28 x 1)

Example: Image Binarization.

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

Binarization is defined as

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

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

Example from P.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.

$$\frac{\iint_{\Omega} (x-\bar{x})^{m} (y-\bar{y})^{n} B(x,y)\,dx\,dy}{\iint_{\Omega} B(x,y)\,dx\,dy} \quad \dots (2)$$

order (m, n)

Image Plane. $M \times N$ (Ω)

Binary Image ($B(x,y)$)

Area, Size (denominator)

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

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

Where

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

and

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

Sept 19 (Tue).

Example: Consider the Architecture of MNIST CNN.

Roadmap:

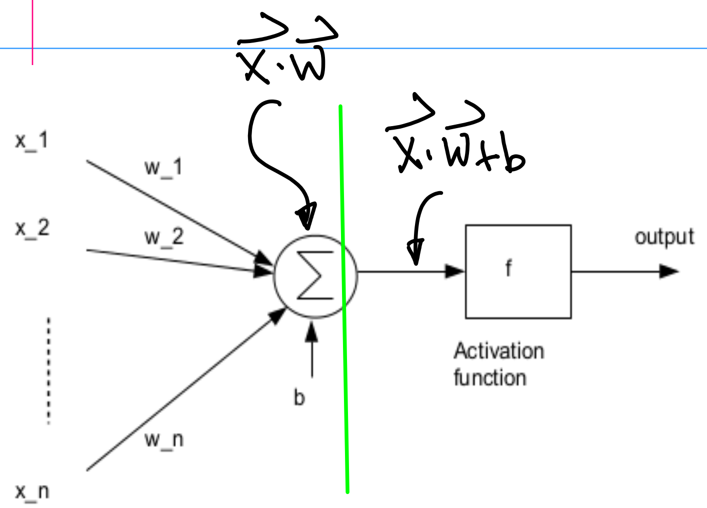
MNIST CNN with Preprocessing. To generate Bounding Boxes → Yolo. Automate Bounding Boxes Creation R.O.I. → Yolact Semantic Segmentation

CMPE258

F2023.

19/

Ref: On the github 20223-103C~



Note 1. Feature vector, or Input, Excitation

$\vec{X} = (x_1, x_2, \cdots, x_n)$ ... (1)

$\vec{W} = (w_1, w_2, \cdots, w_n)$ ... (2)

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

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

b: Bias, e.g., offset

Note 2. $x_1 w_1 \rightarrow$ the Neuron input 1 to

$x_2 w_2$: input 2 to the Neuron.

$\vdots$

$x_i w_i$: " i "

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

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

$= \vec{X}\cdot\vec{W}$ ... (3)

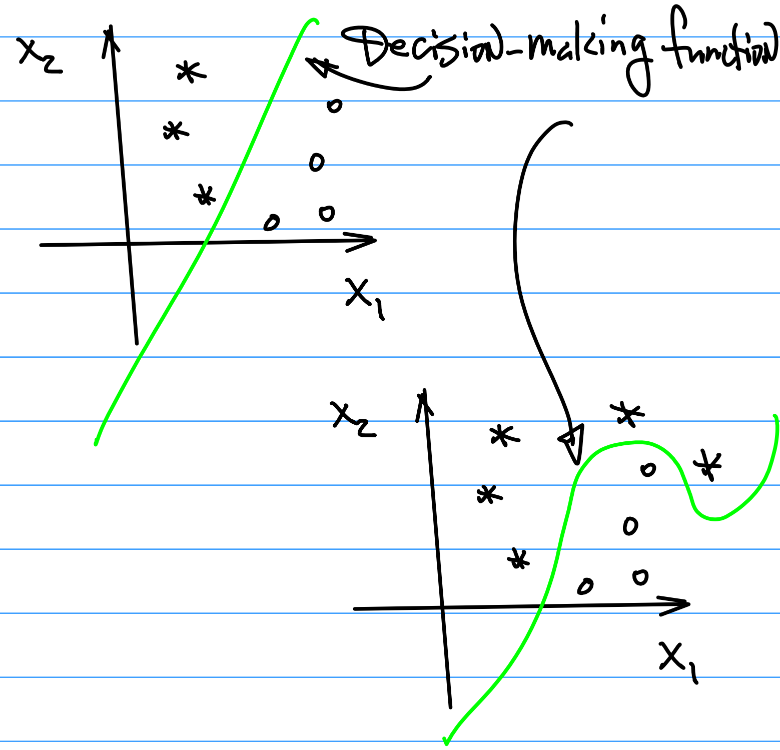
Together with b (Bias), we have

$\sum_{i=1}^{n} w_i x_i + b = \vec{X}\cdot\vec{W} + b$

... (4)

Note 2: The "Bigger Picture" of this Part of the Architecture.

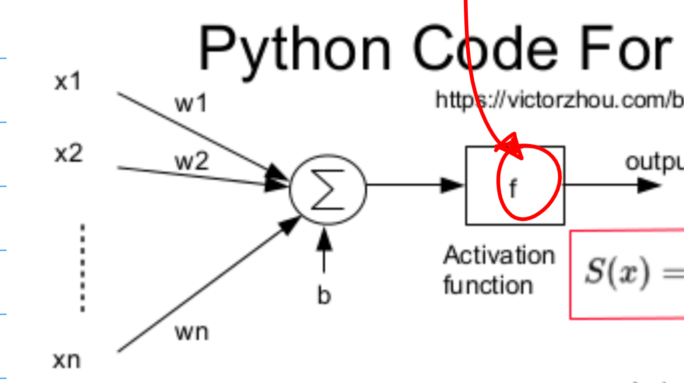
Feedforward NN.



Note 3. Activation Function.

$$f\left(\sum_{i=1}^{n} w_i x_i + b\right) \quad \cdots (5)$$

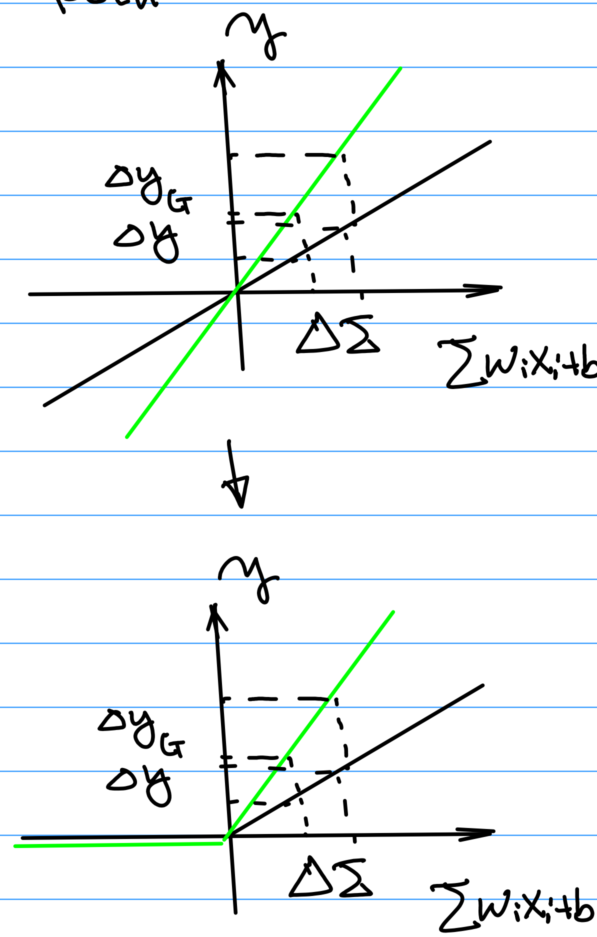
Note: RELU



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

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

Output of the Neuron.



Note 4. Sigmoid function

orzhou.com/blog/intro-to-neural-networks/

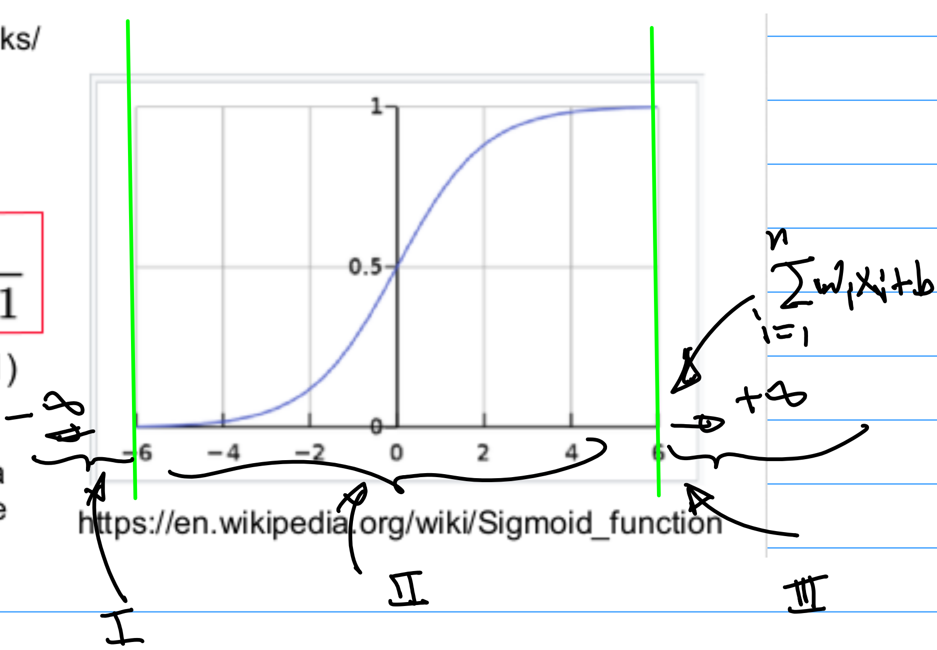
output

$\Sigma w_i x_i + b$.

n

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

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



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

CMPE258
F2023

Sept. 21 (Th).
Continuation on the Formulation of Feedforward NN.

Ref: github.

2022S-103c-NN-Intro-Python-v5.pdf

Sample plain python code on github.
Example.

A sigmoid
mathemat
characteri

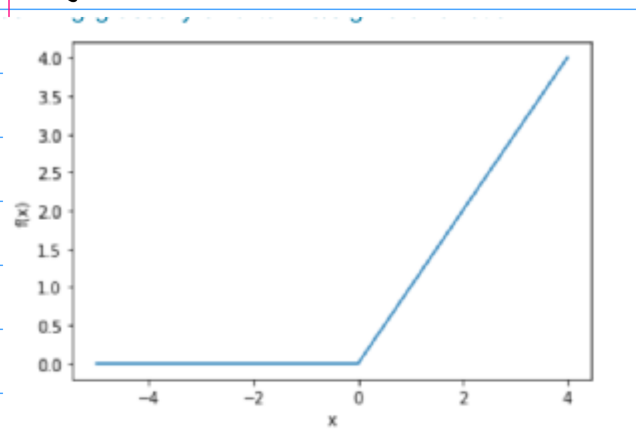
Note: 1. define activation function

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

2. define a single neuron

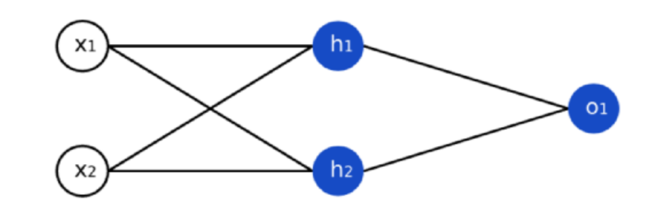
```
class Neuron:
  def __init__(self, weights, bias):
    self.weights = weights
    self.bias = bias

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

Fig. 1.



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

Can be applied to the Neurons in the following illustration



$$f(x) = \begin{cases} \alpha x & x \geq 0 \\ 0 & o/w \end{cases} \quad \ldots (1)$$

Note: Comparison to Sigmoid in terms of its Computation

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

from Taylor Expansion.

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

$$+ \cdots + \frac{d^k f}{dx^k}(x-x_0)^k + \cdots + R_n(x) \quad \ldots (3)$$

CMPE258

F2023.

Note 1. y output of a Neuron. $\begin{cases} y_{pred} \\ y_{true} \end{cases}$ Compare the two, to evaluate the performance of a Network.

Define Loss Function

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

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

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.

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

$$\downarrow$$

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

$$\downarrow$$

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

Note 2. y_{pred}/y_{True}?

$y_{pred} - y_{true}$ $=0$ the Same

$y_{pred} - y_{true}$ $\begin{cases} >0 & y_{pred} > y_{true} \\ <0 & o/w \end{cases}$

Note 3. Comparision for a large dataset

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

$\tilde{y}_n - y_n$.

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

y ground Truth.

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

To Avoid Cancellation of errors. Let's Square them.

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

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

Average of the Accumulative Comparision for the entire Data Set.

M.S.E (mean square error).

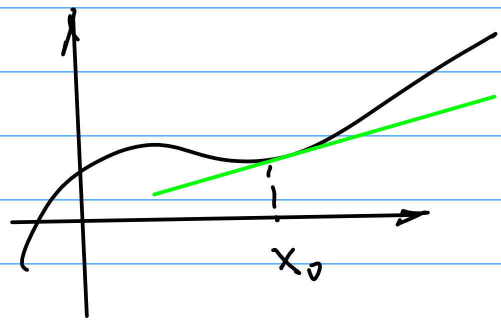
Example: Gradient Descend

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

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

Consider One dimensional Case first.

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \quad \cdots (5)$$



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

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

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

$$\vdots$$

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

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

$$\begin{pmatrix} \frac{\partial f(x_1, x_2, \cdots, x_n)}{\partial x_1} \\ \frac{\partial f(x_1, x_2, \cdots, x_n)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x_1, x_2, \cdots, x_n)}{\partial x_n} \end{pmatrix} \quad \cdots (6)$$

Sept. 26 (Tue).

Note 1. New Homework Posted on the CANVAS.

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

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

Example: On the gradient descent Technique.

Ref:

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

gradient.

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

Note 1. Weights, $\vec{W} = (w_1, w_2, \cdots, w_n)$ where $\vec{w}$ matches to the feature Vector $\vec{X} = (x_1, x_2, \cdots, x_n)$

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

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}^{t} \quad \cdots (1)$$

"t" transpose $\rightarrow$

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

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

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

$$\vdots$$

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

$$\vdots$$

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

Now, verify the equation. From Eqn (6).

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

Consider 1D Case

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

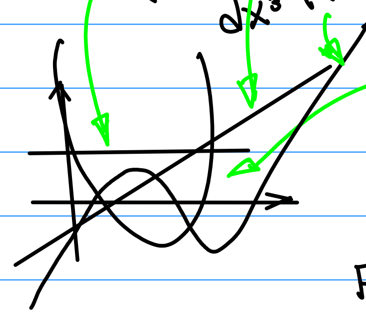


Fig. 1

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

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

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

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

Start from here

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

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

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

Or,

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

$\rightarrow \nabla f(x_1, x_2)$

Hence,

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

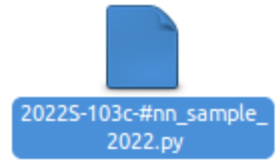
$$f(x_1,x_2) - f(a,b) = (\Delta x_1, \Delta x_2)\nabla f = -(f_{x_1}^2 + f_{x_1}^2) \qquad (12)$$

The error becomes smaller each iteration as long as $\Delta x_1, \Delta x_2$ defined by Eqn (5)

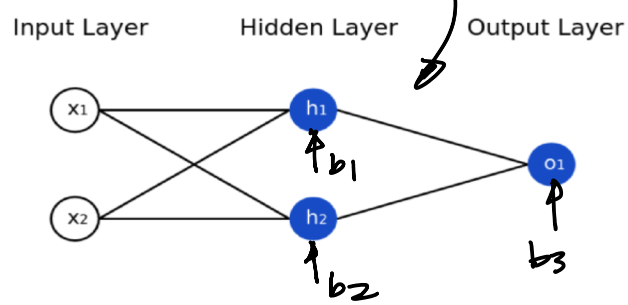
!! "<"

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

Now, Review the Python Sample Code

Ref → 

Sample code for this NN.




```
#------------------------------------
data = np.array([
    [1, 2.5],      # person A
    [1, 3],        # person A
    [2.1, 3.4],    # Person A
    [2.1, 1], # person B
    [3.3, 1], # person B
    [3, 2.3], # person B
# HL 2020-9-7 Part E
# for the testing of adaptive learning
# (1) create a new program based on this o
```