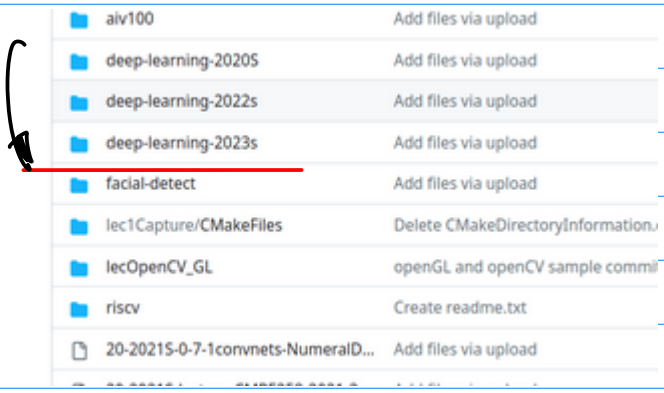
CMPE258
Fall 2023

August 22 (Tue)
Organizational meeting.
1. Class Material on github
github/hualili



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 (AI): (Human Intelligence)
is Symbolic Representation of
Learnt Experience.

CMPE258
Fall 2023

4. Projects. 2 pjts | team project (Semester Long) } ~~30%~~ 25% ~~30 pts~~

5. In-Person Class; CANVAS is utilized to post Homework/Project Requirements, and to collect the submission of the Homework, as well as for the exams.

NO

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

6.

Grading Information

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

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.

Ref: github.
2022F-103-NN-Intro-Python-v5-2022-8-25

8. Software Tools & Dev. Environment

Python. Pycharm.
Anaconda.
TensorFlow.

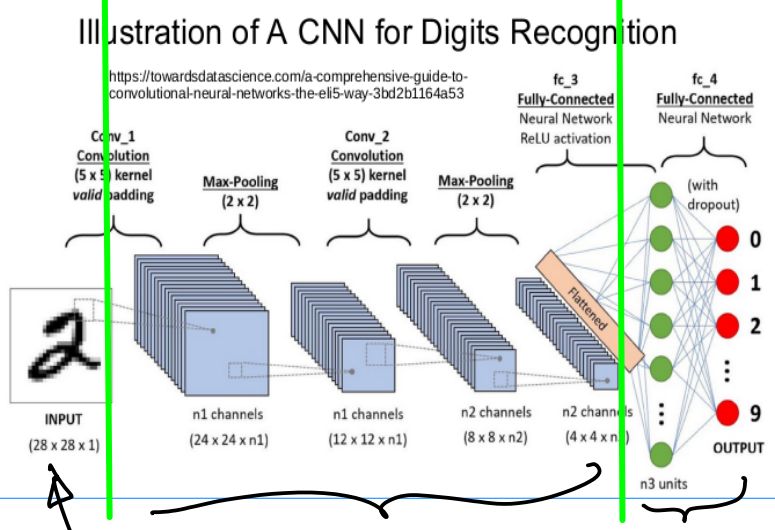
Note: Lab Space for the class Rm268.

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

August 24 (Thursday)

Example: Architecture Overview.

Note 1.

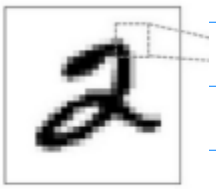


Preprocessing Computer Vision.

Convolution Layers
Feature Extraction
To produce Feature Vectors.
$\vec{u} = (u_1, u_2, \cdots, u_N)$

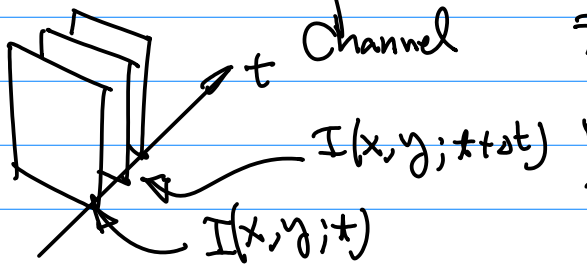
Feed Forward Neural Networks.
Decision Making

2. Image Definition



$I(x, y; t)$

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



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

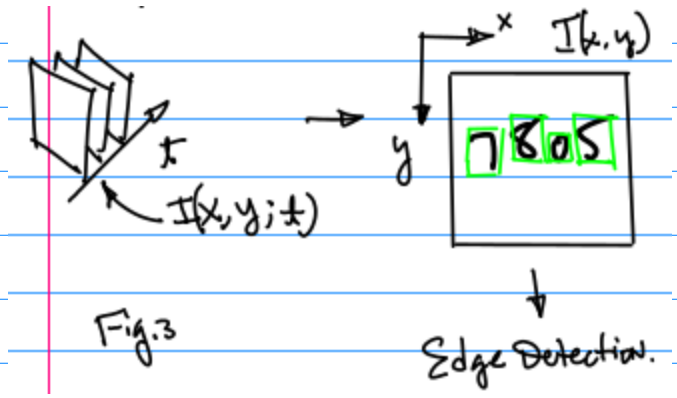


Fig.3

Comparison to Web Cam.

1080p Resolution: 1920x1080

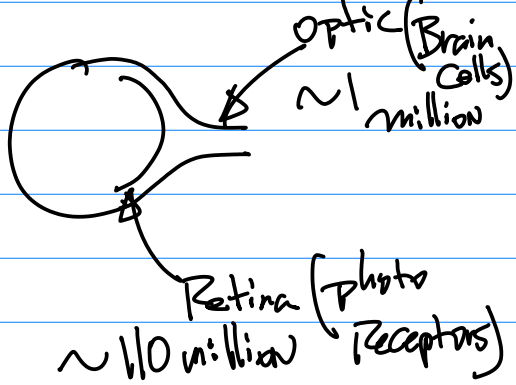
$\sim 2^{11} = 2 \cdot 2^{10} = 2K$

$\sim 2^{11}$

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

$= 4$ Meg

$\simeq 4$ million.



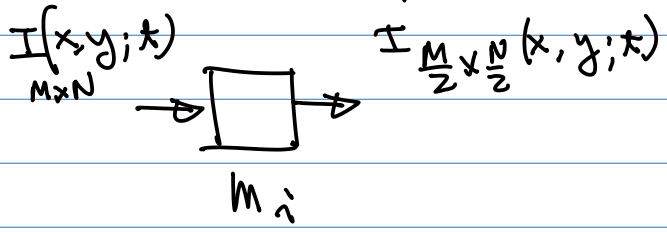
3. For the Convolutional Lay., we denote it as C_i, where $i = 1, 2, \cdots, N$

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

So, for the example, we have

$C_1 M_1 \rightarrow C_2 M_2$

Max pooling for Resolution Reduction / Feature Reduction.



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

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

M: No. of Neurons / Nodes

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

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 smartphone to take a photo, And upload the photo to your laptop to display.

Note: Sample code (Python) was posted on the github.

(2) Anaconda Installed on your Laptop.

Note: Readme for Anaconda installation was posted on the github.

(3) Create ChatGPT Account.

Python Interface to Chat GPT 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/thuaiki/OpenCV

↓

2022

Note1:

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

Image Kernel

...(1)

$n_1 = x$, $n_2 = y$

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

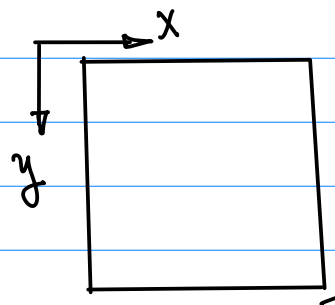


Fig.1

Note2. 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)

Note3. Kernel b(x,y) Can be rewritten k(x,y)

Size of A kernel is denoted as K x K

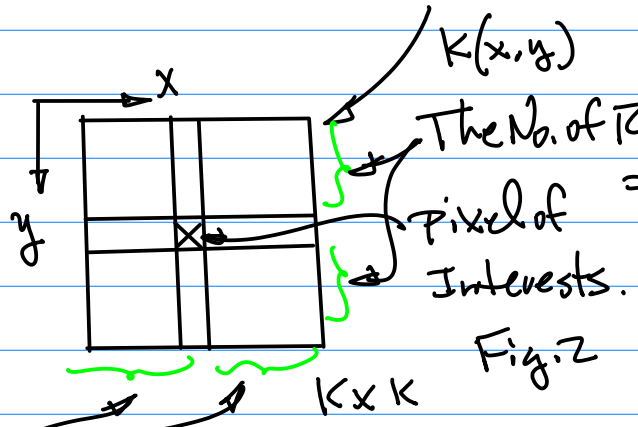


Fig.2

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.

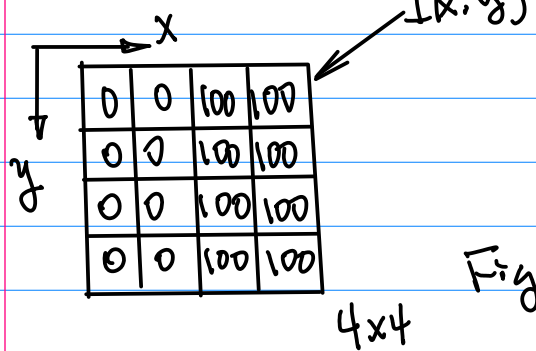


Fig.3a

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

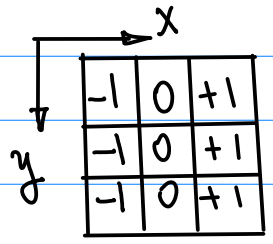


Fig.3b

Step1. Take the Kernel, and place it at the initial Position.

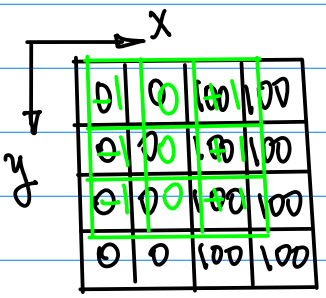


Fig.3c

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

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

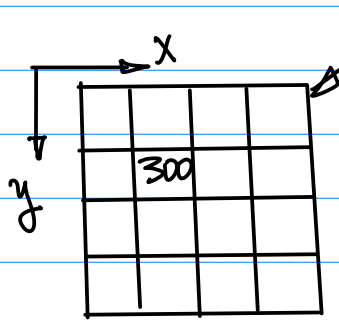


Fig.3d

Step2. Shift K(x,y) to the Right By 1 pixel (on the Same Row).

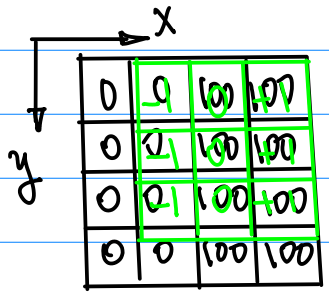


Fig.3e

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

August 31 (Th).

Note: 1° Honesty Pledge on CANVAS, Signed pdf due this Friday. (ON CANVAS). Homework ON CANVAS
2° will post/Anaconda Installation & OpenCV Installation, display an image.
(1) Screen Capture of the Activated CONDA Environment, with Personal identifier;
(2) Screen Capture of the OpenCV Display With P.I.D.
Sample Code for the display to Be Re-posted on github, 2023F

Example: Theoretical Aspects for 2D Convolution.

Ref: 1° PPT on the github.

2° 2023S-100-note-...
2023-03-21.pdf

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

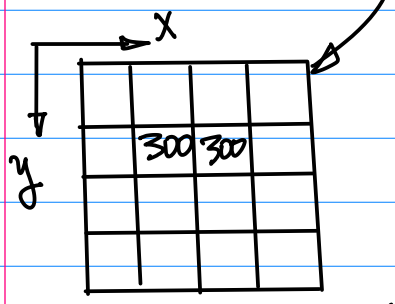


Fig.3F

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

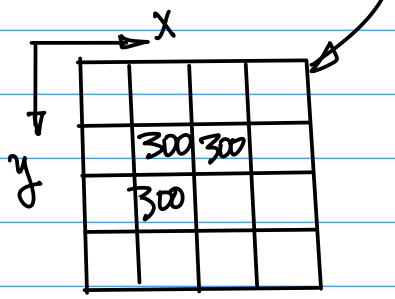


Fig.3G

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

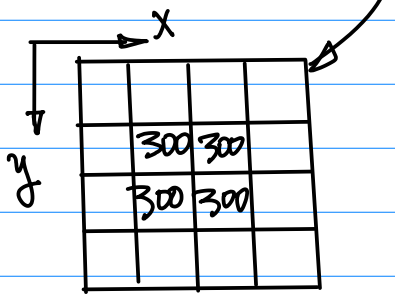


Fig.3H

Note: 2D Convolution (Continuous Case)

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

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

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

Image = Kernel

Note1. Ω : Image Plane(s)

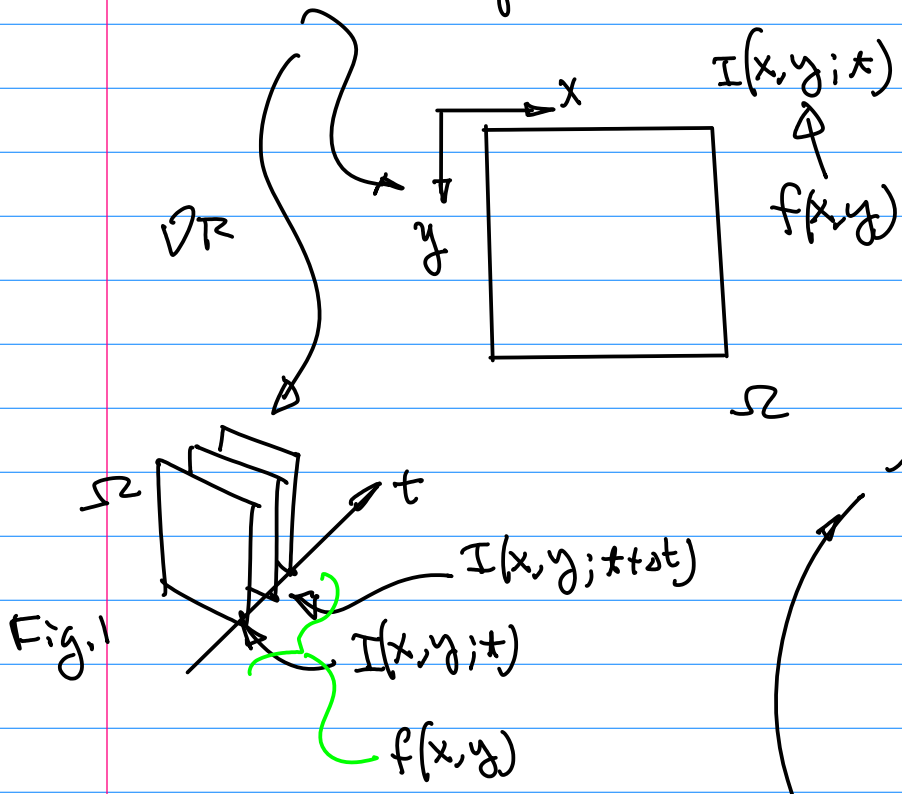


Fig.1

Note2 : Kernel

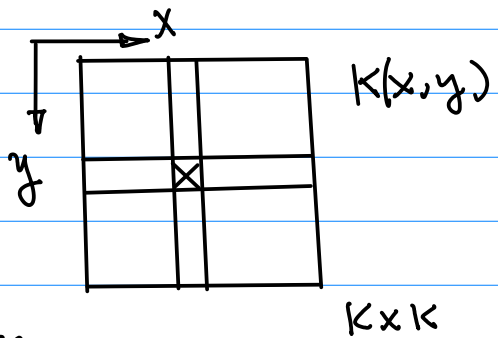


Fig.2a

① $K(-x, y)$

Flip the Kernel w.r.t y-axis

② Flip the Kernel w.r.t. x-axis.

(since $K(-x,y)$ Changed to $K(-x,-y)$)

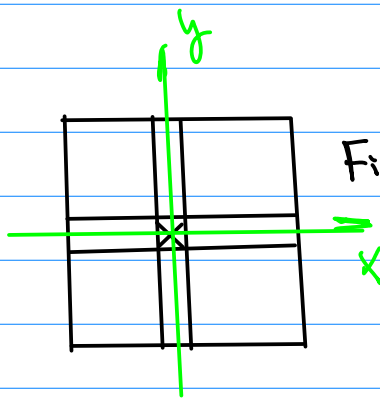
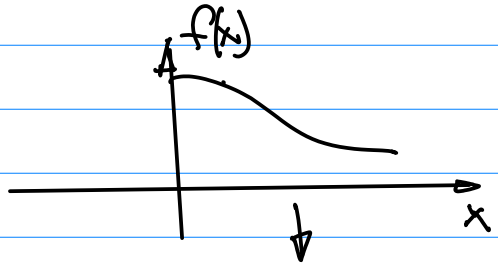
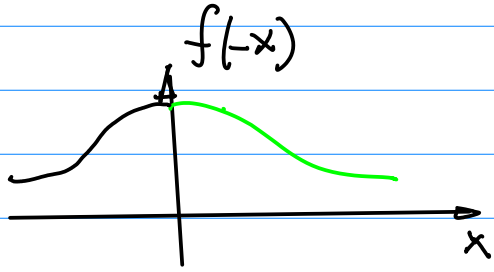


Fig.2c

Note: for 1D Case





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

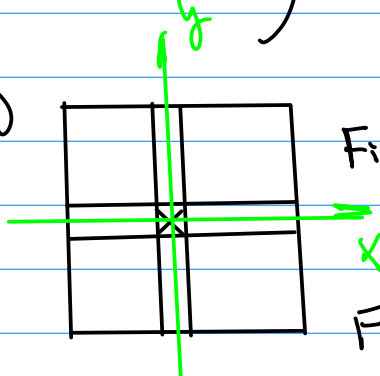


Fig.2b

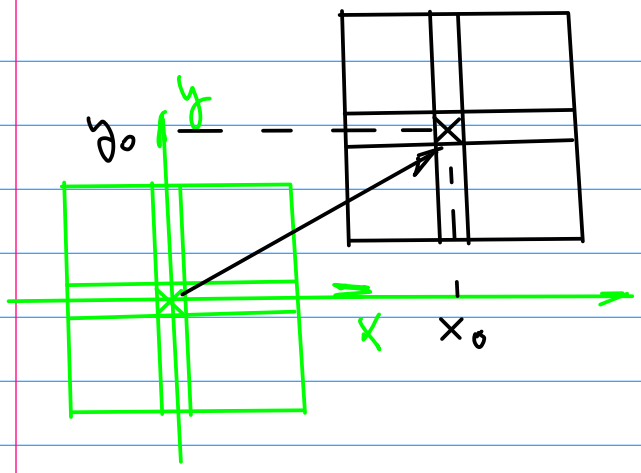


Fig. 2d

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/Image (OR Digital Feature plane)

Replace $\iint$ → by $\sum_{K_1} \sum_{K_2}$

$$c(n_1, n_2) = \sum_{K_1=-\infty}^{\infty} \sum_{K_2=-\infty}^{\infty} a(k_1, k_2) b(n_1 - k_1, n_2 - k_2)$$

Image Kernel ... (2)

Dimension Change Due to the Convolution.

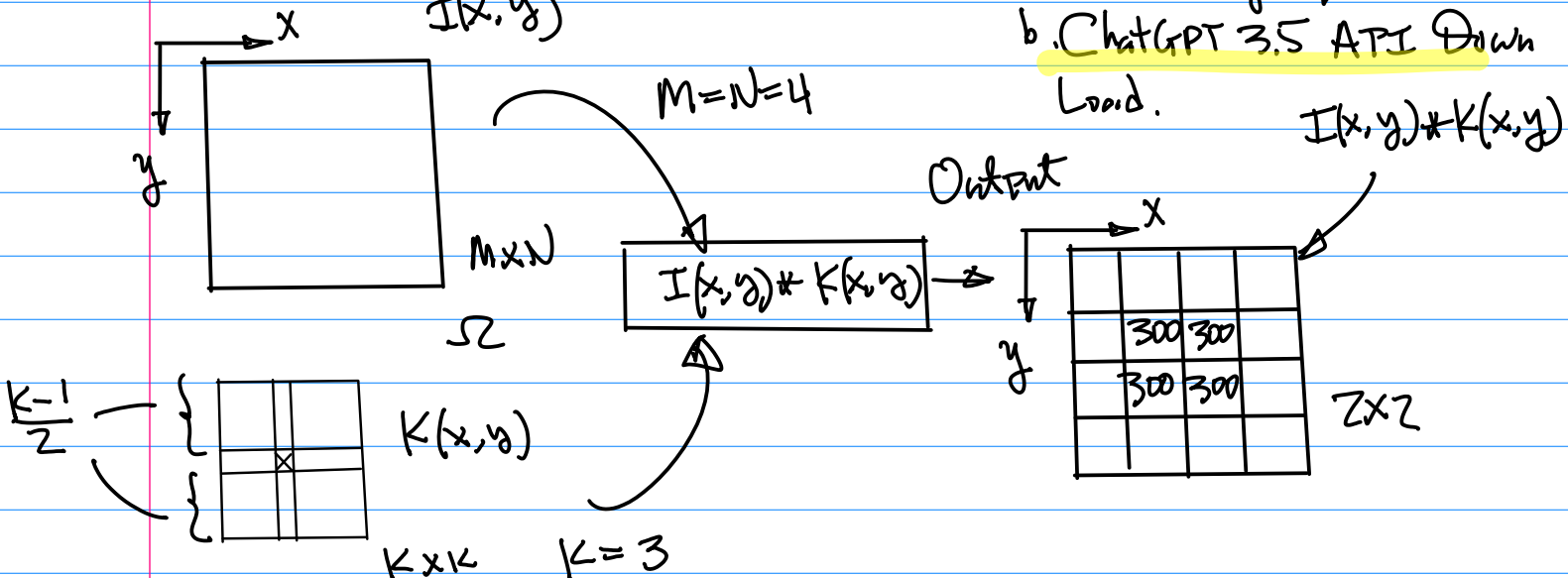
Sept. 5 (Tue)

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.

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

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

(Resolution difference.)

Now, the total No. of feature layers $= n|$

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

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

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

LoG Laplace of Gaussian.

Gabor Operators

$$G(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-m_x)^2+(y-m_y)^2}{2\sigma^2}} \quad \cdots (1)$$

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

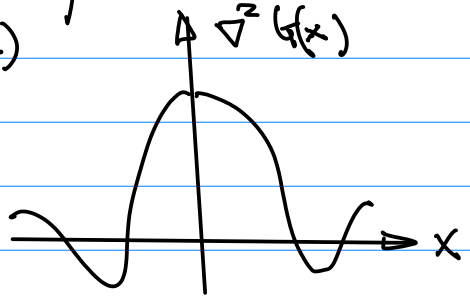


Fig 1.

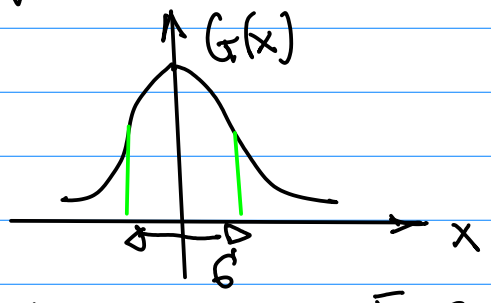


Fig.2

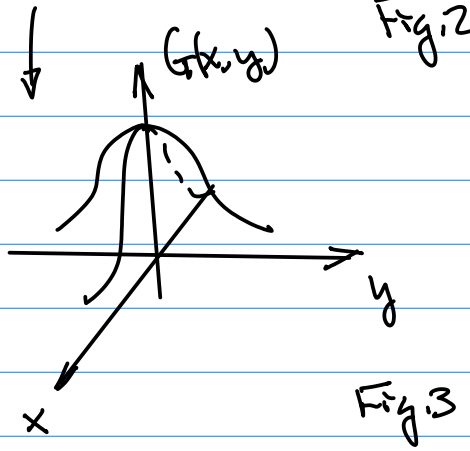


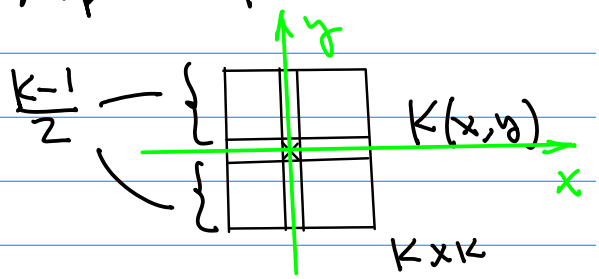
Fig.3

CmpE258

F2023

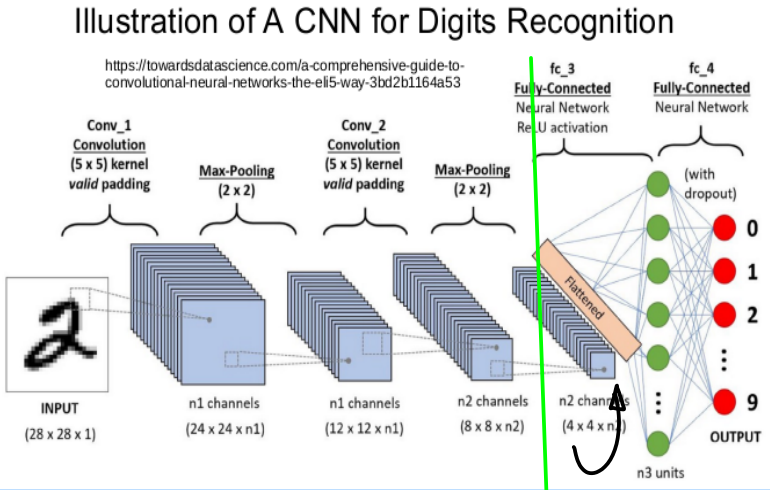
11/

Take G(x,y) in Fig.3, map it to KxK.



n1 Kernels produce n1 Layers.

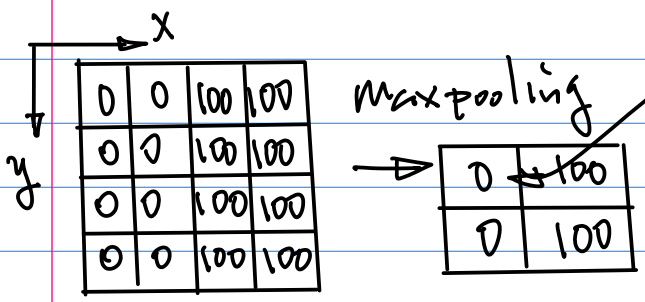
Now, consider the Architecture of MNIST



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

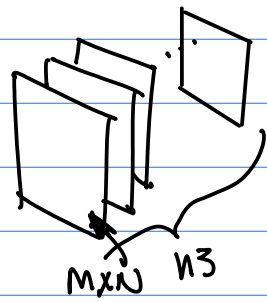
Now, consider a technique for feature Reduction,

given an image I(x,y)

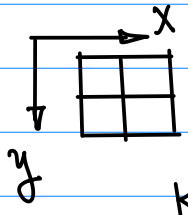


Feature Layer

Max{0,0,0,0}



Mapping



KxK
where K=2

Example: Sample code OpenCV for Displaying an image

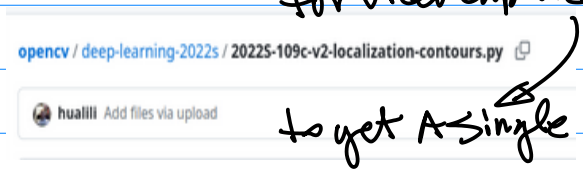
```
cv2.imread(imageName, cv2.IMREAD_COLOR)
cv2.imshow(window_name, src)
```

Sept. 7 (Th).

Note: 1° About OpenCV
Reference/Sample
Code.

On github.

a) for Video Capture



to get A single
Frame.

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

2022S-104d ~ Python Code

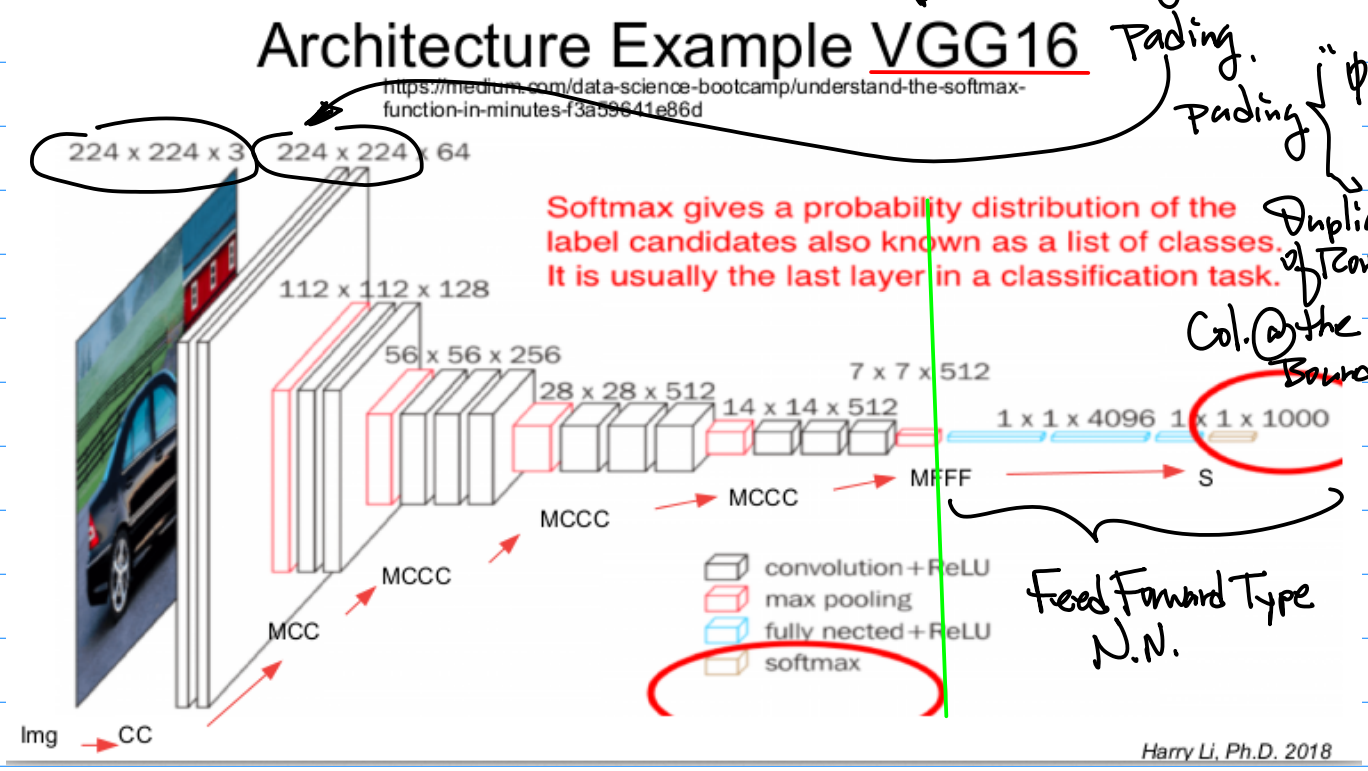
c) Script for Conda Environment
On github:
2022S-104b ~
" -104C ~

d) Readme for Creating Conda
Environment.

Note2. 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.

Note1. Architecture; Note2: Padding v.s. No Padding.

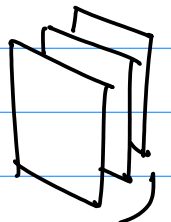


"0"s Padding → Duplication of Const Col. @ the Boundary

Feed Forward Type N.N.

Note3: Convolution Discussion.

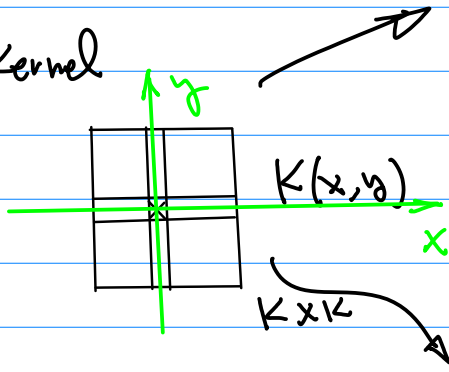
One Example

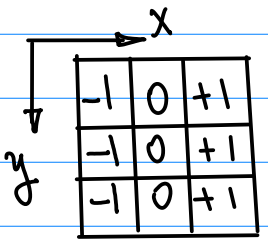


Color Image

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

Kernel





LoG $\nabla^2 G(x,y)$ → PP.10.

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

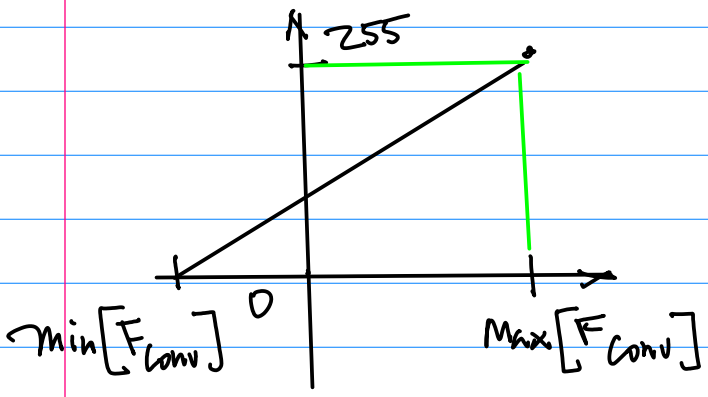
$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) G(x,y; \mu_x, \mu_y, \delta_x, \delta_y)$

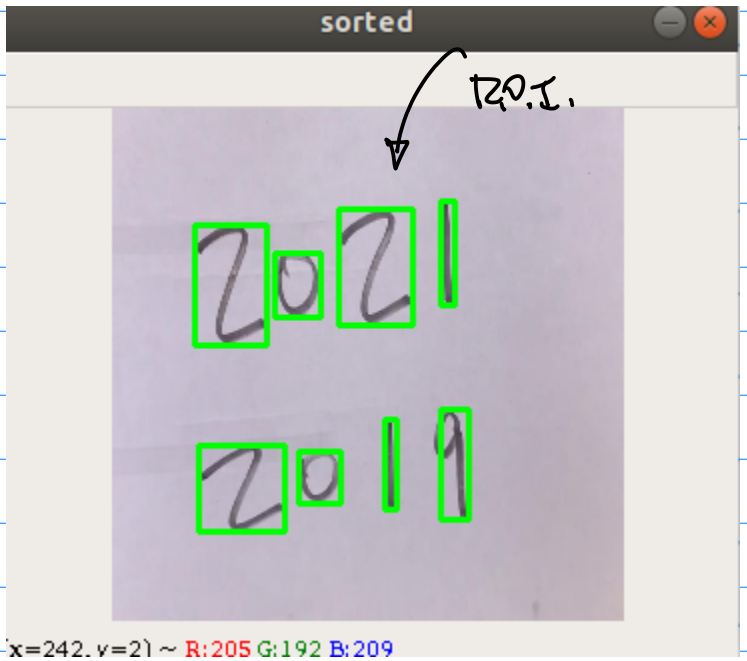
To perform Convolution. → $K(x,y)$

First, kernel is one channel in this case;

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

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





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

(1280×720 (M×N)
Col. Row.

1920×1080 (M×N)
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

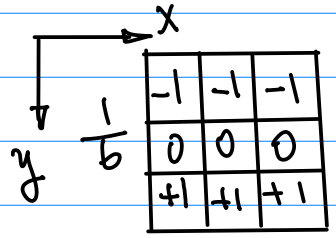
ON CANVAS as well.

Due on Nov. 27 (Monday) 11:59 P.M.

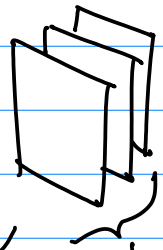
Example: Suppose a color image I(x,y) is given below, and a Convolution Kernel K(x,y)

is given as follows

A Color Image I(x,y)

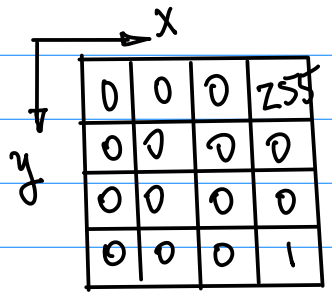


3x3.

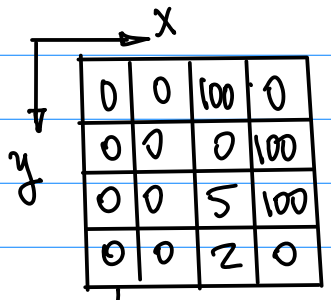


Colour Image

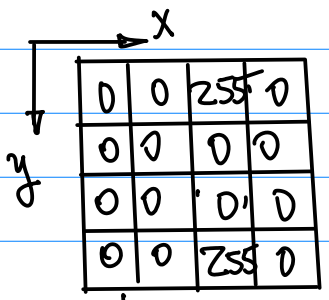
Find Convolution Result.



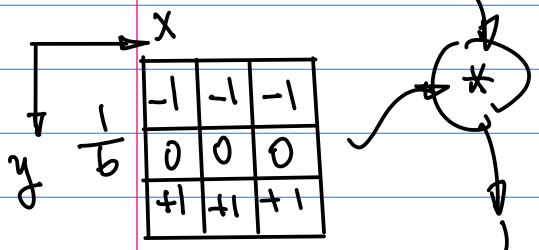
$I_r(x,y)$

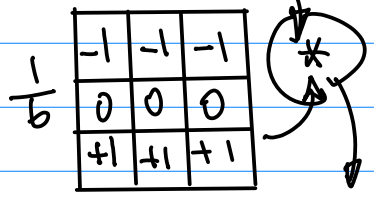


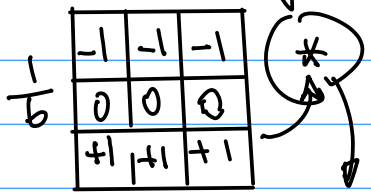
$I_g(x,y)$



$I_b(x,y)$







0	-255
0	1

-95	5
2	-98

-255	-255
255	255

Note: In the future, we have Need to Reduce the Number of feature Layers By introducing Newer Kernel(s) for Convolution.

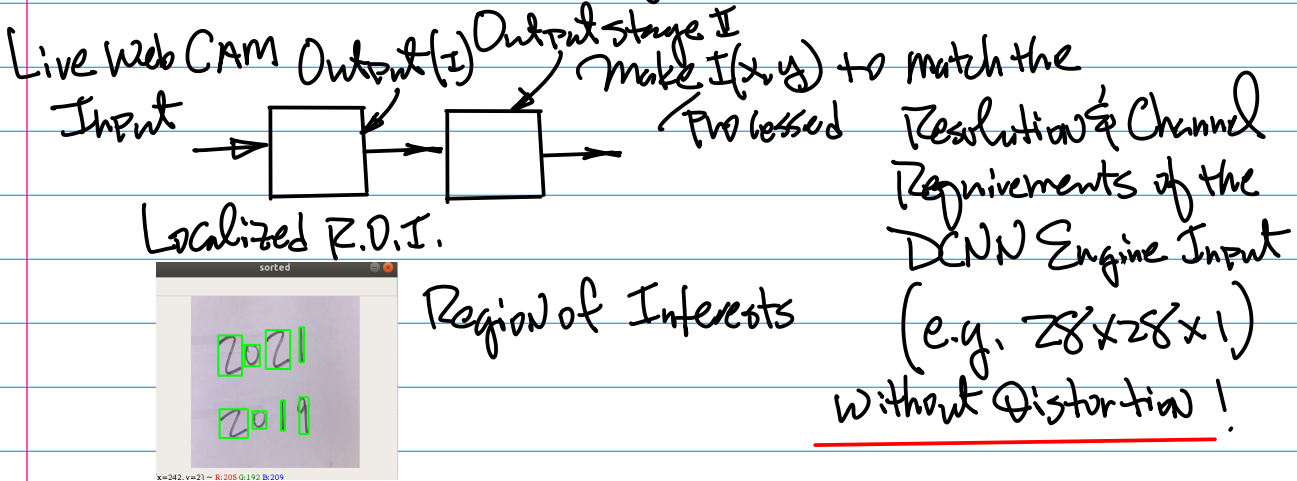
Homework: Due 1 week from Today.
(Sept. 14th).

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

2° OpenCV Code to Handle
a. Live CAM Input Video, (Web) 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.



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.

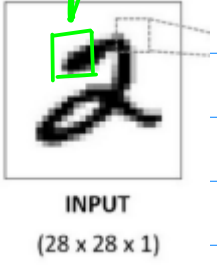
Submission:

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.



A. Binarization. To Obtain A Representation of A Digit.
Note: Binarization may lead to the Loss of useful information.

B. To Be Continued.
Edge Detection CANNY, LOG, Gabor
Note: Input Should be a grayscale (8 bits), Output Should be 8 bit (1 channel)

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

Example: Image Binarization.

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

Binarization is defined as

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

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

Example from PP. 18, Lecture Notes on Github, 2023S.

Note: Treat the top Left Corner as (1,1) for the Binarized Image Descriptors Calculation (to Void Skewed Result).

Most Important Descriptors are those Built Based on Moments.

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

(m, n: Order; Ω: Image Plane, $M \times N$; $B(x,y)$: Binary Image; Area, Size)

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

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

Where

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

and

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

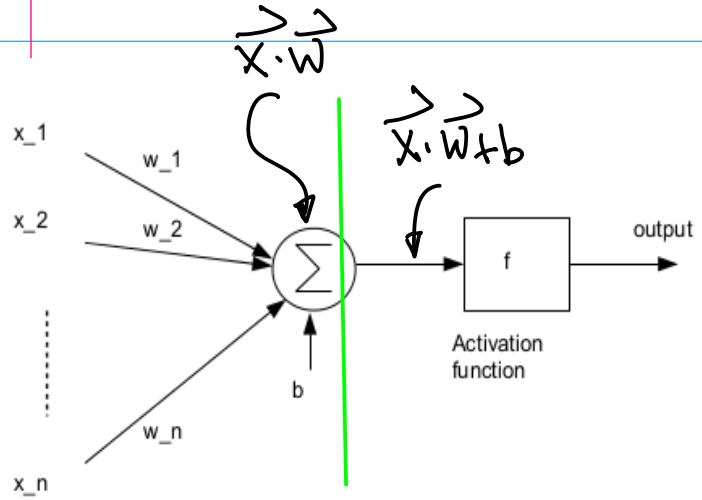
Sept 19 (Tue).

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

Ref: On the github. 2022S_103C~



Note 1. Feature vector, or Input, Excitation

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

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

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

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

b: Bias, e.g., offset

Note 2. $x_1 w_1$ → the Neuron input 1 to

$x_2 w_2$: input 2 to the Neuron.

$x_i w_i$: " i "

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

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

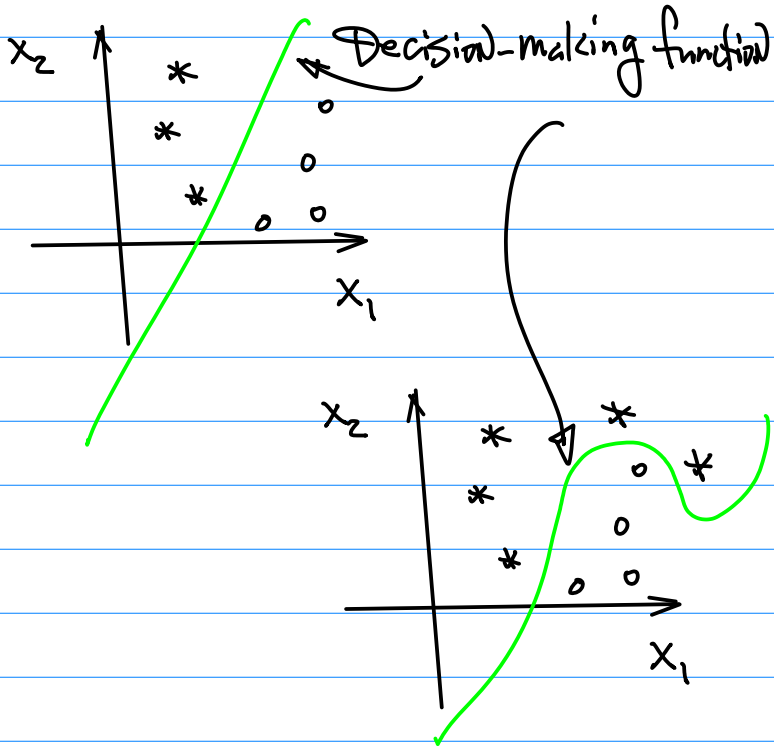
$$= \vec{X} \cdot \vec{W} \quad \cdots (3)$$

Together with b (Bias), we have

$$\sum_{i=1}^{n} w_i x_i + b = \vec{X} \cdot \vec{W} + b \quad \cdots (4)$$

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

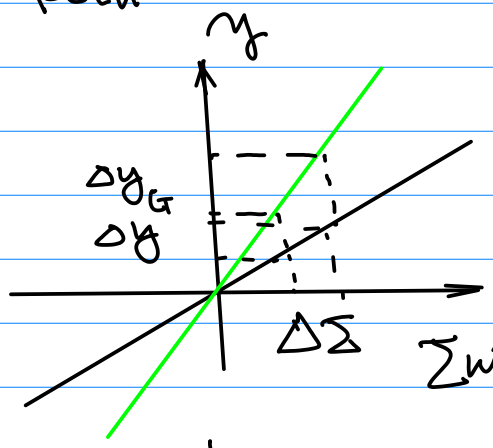
Feedforward NN.

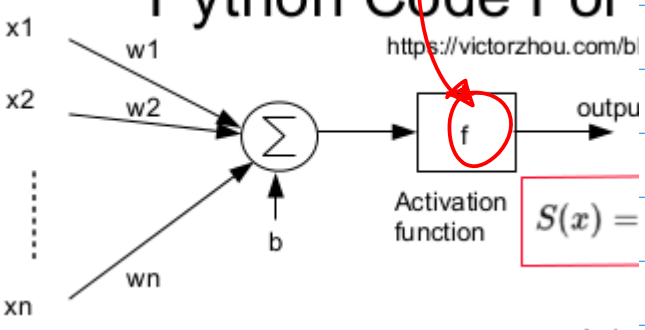


Note 3. Activation Function.

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

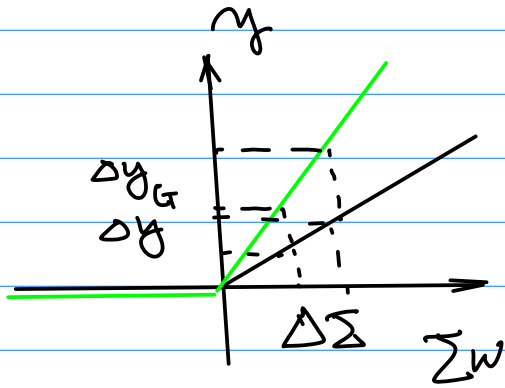
Note: ReLU





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

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



Output of the Neuron.

Note 4. Sigmoid function

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

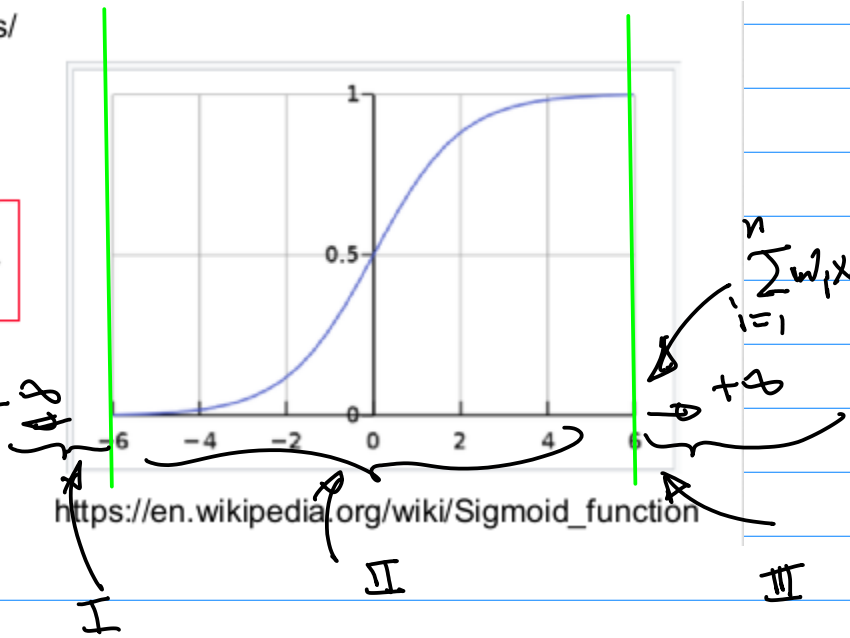
output

$\sum w_i x_i + b$.

n

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

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



CMPE258
F2023

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

Ref: github.

21/
Sample plain python code on github.
Example.

Note: 1. define activation function

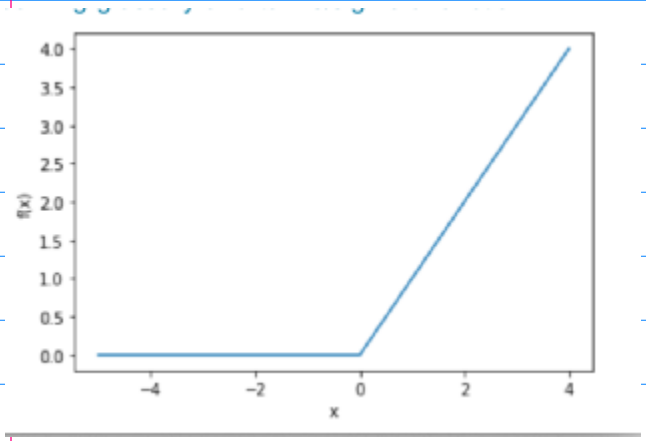
```
def sigmoid(x):
  # Our activation function: f(x) = 1 / (1 + e^(-x))
  return 1 / (1 + np.exp(-x))
```

A sigmoid mathemat characteri

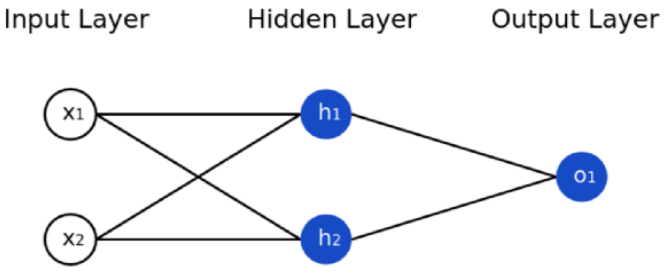
2. define a single neuron

```
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



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

Note: Comparison to Sigmoid interms of its Computation

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

from Taylor Expansion.

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

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

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/

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

$$\downarrow$$

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

$$\downarrow$$

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

$$\downarrow$$

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

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.

Note 2. y_{pred} / y_{True} ? $\rightarrow y_{pred} - y_{true}$

$y_{pred} - y_{true} \begin{cases} =0 & \text{the same} \\ >0 & y_{pred} > y_{true} \\ \leq 0 & \text{O/W} \end{cases}$

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).

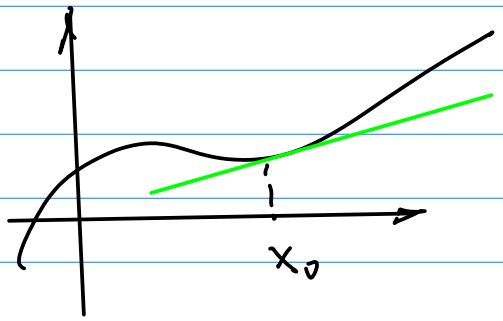
Example: Gradient Descend

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.

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

CMPE258
F2023

24/

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.

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

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

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

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

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

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

$$\vdots$$

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

$$\vdots$$

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

Now, verify the equation.

From Eqn (6).

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

Consider 1D Case

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

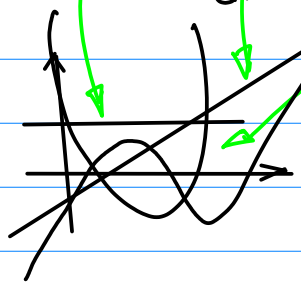


Fig. 1

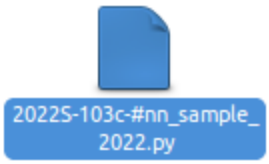
The multidimensional Taylor Expansion can be realized based on the Similar Principle.

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

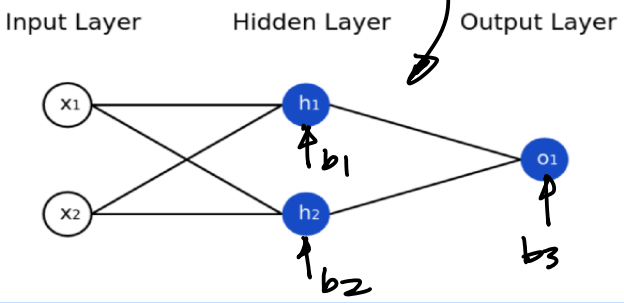
$y = ax + b \rightarrow y' = ax + b + c$, $b' = b + c$

Now, Review the Python Sample Code

Ref →



Sample code for this NN.



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

Start from here

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

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

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

Or,

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

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

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

Hence,

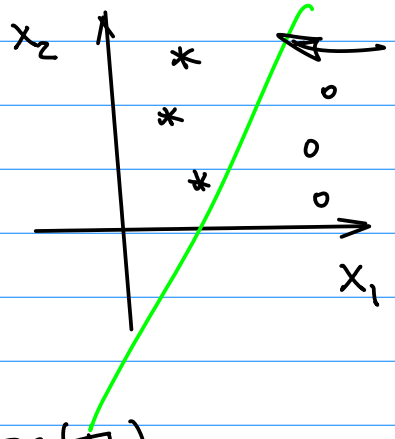
$$f(x_1, x_2) - f(a,b) = (x_1 - a, x_2 - b)\nabla f \quad (10)$$

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

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

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

!! "<"



Sept. 28 (Th).

Note 1. LLM (Large Language Model) Based on Chat GPT. focus API Package, on

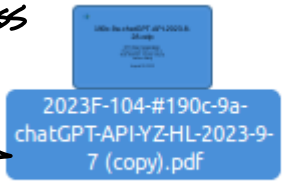
CMPE258

F2023

Note: the 1st of the 3 PPT.

Deploy API with Python.

Ref: ON the Class github.



ChatGPT API from OpenAI

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

Step 1. Log in ChatGPT, then click on API

OpenAI

Step 2. Design your prompt to program your model

ChatGPT →
Interact with our flagship language models in a conversational interface

DALL·E →
Create realistic images and art from a description in natural language

API →
Integrate OpenAI models into your application or business

Example: Conversation with API: a conversation is formatted with a system message first, followed by alternating user and assistant message, see below:

GPT-3.5 Turbo

GPT-3.5 Turbo is optimized for dialogue.

Learn about GPT-3.5 Turbo ↗

GPT-3.5-Turbo Pricing:

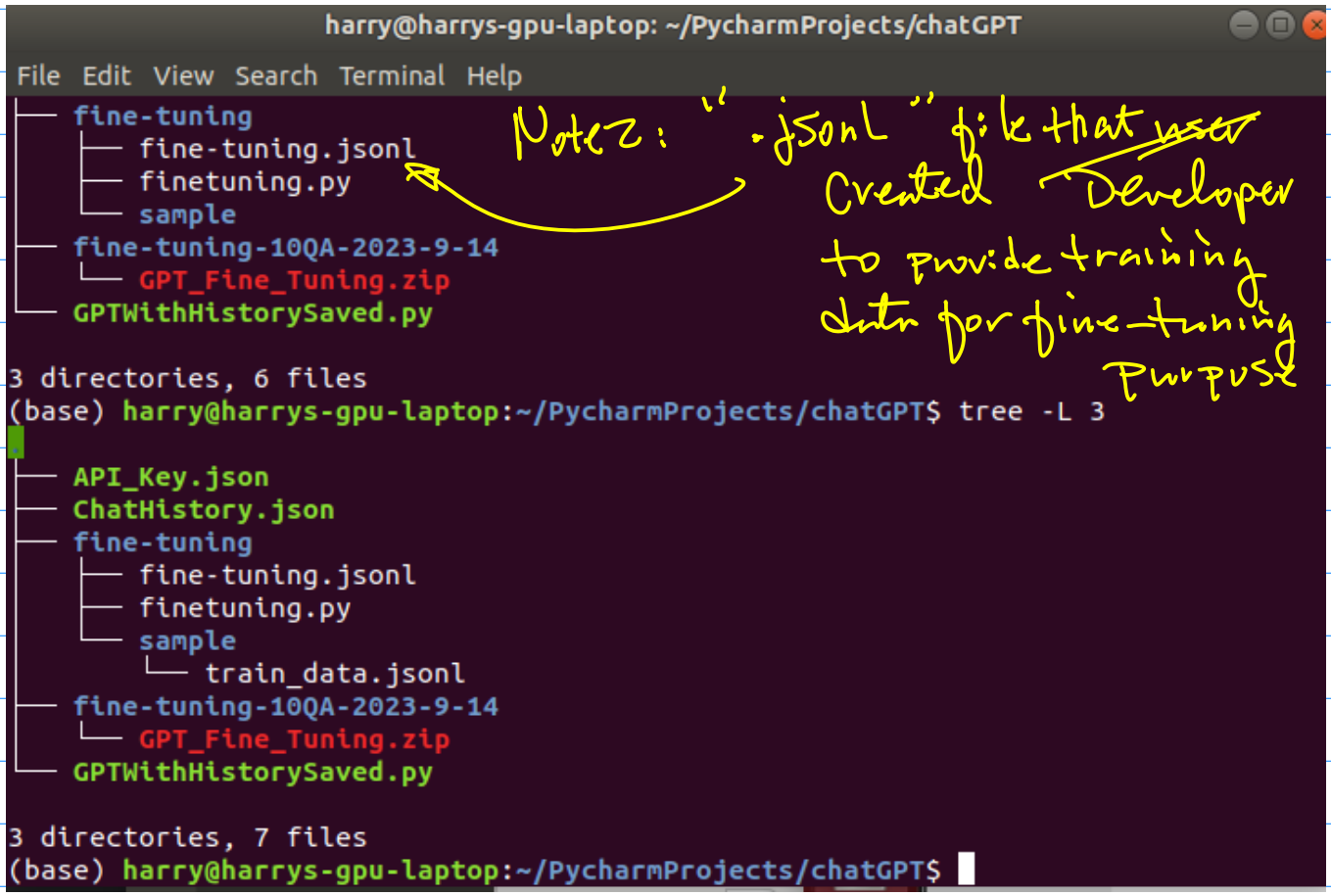
Model	Input	Output
4K context	$0.0015 / 1K tokens	$0.002 / 1K tokens
16K context	$0.003 / 1K tokens	$0.004 / 1K tokens

https://openai.com/pricing#language-models

```
import openai

openai.ChatCompletion.create(
  model="gpt-3.5-turbo",
  messages=[
        {"role": "system", "content": "You are a assistant."},
        {"role": "user", "content": "Who won the world series in 20?"},
        {"role": "assistant", "content": "The Los Angeles Dodgers."},
        {"role": "user", "content": "Where was it played?"}
    ]
)
```

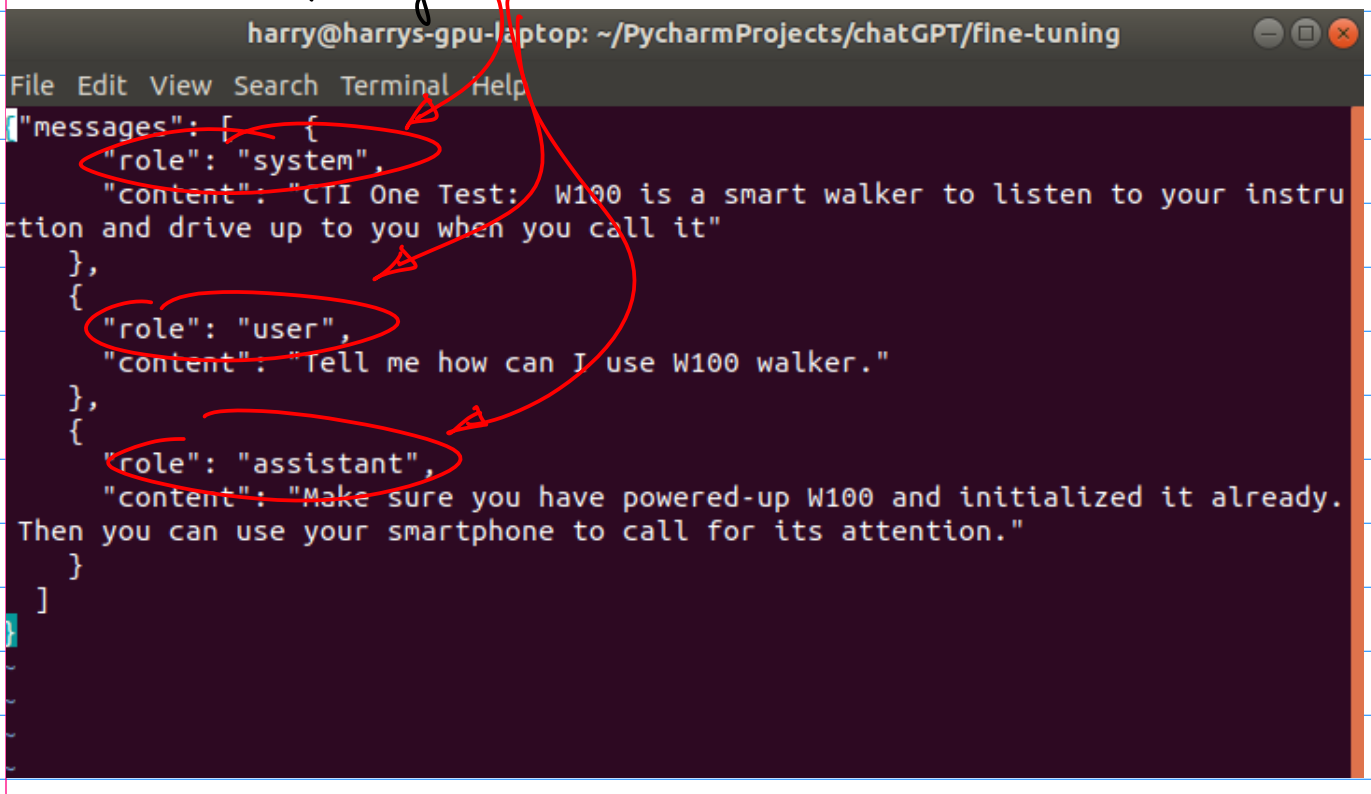
```
harry@harrys-gpu-laptop: ~/PycharmProjects/chatGPT
File  Edit  View  Search  Terminal  Help
(base) harry@harrys-gpu-laptop:~/PycharmProjects/chatGPT$ tree -L 2
.
├── API_Key.json
├── ChatHistory.json
├── fine-tuning
│   ├── fine-tuning.jsonl
│   ├── finetuning.py
│   └── sample
├── fine-tuning-10QA-2023-9-14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

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

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



Note2: ".jsonl" file that ~~user~~ Developer Created to provide training data for fine-tuning purpose

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



Homework, Due 1 week,
Oct. 5th.

1° Install openAI to
use ChatGPT API 3.5;

2° Create and Run "Hello,
the world" Python Code;

3° Screen Captures. for 1°
and 2° with Personal ID.

4° Submission: ON CANVAS.
as zip file. use the
following Naming Convention:
firstName_LastName_4SID_
ChatGPT1.zip.

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

Ref: on the github

2022S-106a-#2019S-38-lec13-BackProp-activation.pdf

Note1. Generalize the Architecture from a Single Neuron.

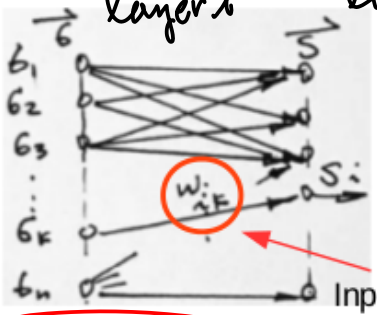
Back Propagation (1)

1. input and output neurons

Input $\vec{G} = \begin{pmatrix} G_1 \\ G_2 \\ \vdots \\ G_n \end{pmatrix}_{n\times 1}$

Output $\vec{S} = \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n_0} \end{pmatrix}_{n_0\times 1}$

The architecture



Input: k
output: i

2. weights w_ik

i for output k for input

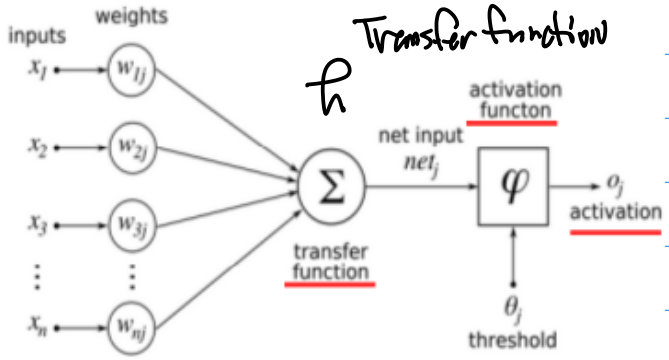
Note2: for the Weights Notation we adopted 2 subscripts

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

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

Other popular notation

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



Transfer function h

3. Activation function

$$S_i = f\left(\sum_k w_{ik} G_k\right) \quad \ldots (1)$$

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

Note3.

Harry Li, Ph.D.

Activation function.

$$S_i = f\left(\sum_k w_{ik} G_k\right)$$

"i" for the output layer.

Input, feature vector, Excitation.

4. Transfer function h_i

let

$$h_i = \sum_k w_{ik} G_k \quad \ldots (2)$$

Index i for the output neuron

Neuron output function S_i

$$S_i = f(h_i) = f[h_i(w_{ik})] \quad \ldots (2\text{-}1)$$

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

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

$$\zeta_i^\mu - S_i^\mu \quad \dots (2\text{-}2)$$

Harry Li, Ph.D.

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

$$D = \frac{1}{2} \sum_\mu \sum_i \left(\zeta_i^\mu - S_i^\mu\right)^2 \quad \dots (3)$$

MSE? (mean Square Error)

N : No. of Neurons

M : No. of Experiments / Epochs.

μ

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

Note: Rewrite the output S_i by its functional Form.

Activation function (Transfer function)

7. minimize the error wrt to w_ik

$$\frac{\partial D}{\partial w_{ik}} = \frac{\partial}{\partial w_{ik}} \cdot \frac{1}{2} \sum_\mu \sum_i \left[\zeta_i^\mu - f(h_i^\mu)\right]^2$$

$$= \sum_\mu \left[\zeta_i^\mu - f(h_i^\mu)\right] f'(h_i^\mu) \frac{\partial h_i}{\partial w_{ik}}$$

— Derivative Calculation is Based on "Chain Rule"

Note: Eqn(5) on TP. 24.

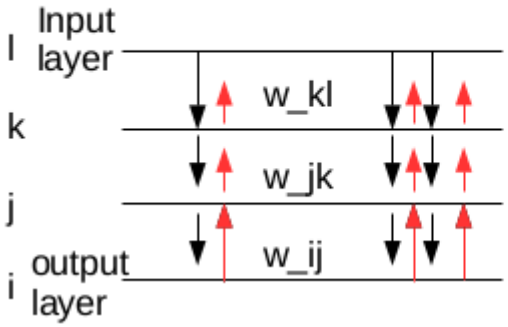
8. Training the NN by updating the weights

$$w_{ik}(t+1) = w_{ik}(t) + \delta w_{ij}(t) \quad \dots (4)$$

where

$$\delta w_{ij}(t) = -\epsilon \frac{\partial D}{\partial w_{ij}} \quad \dots (5)$$

10. feed forward NN (4 layers)



Black: for Experiments.
Red: Training Update

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

Math. Description.

Back Propagation (4)

11. chain rule, updating the weights (training)

For layer i

$$\frac{\partial D}{\partial w_{ij}} = \sum_n \left[s_i^n - f(h_i^n) \right] f'(h_i^n) \frac{\partial h_i^n}{\partial w_{ij}}$$

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

For layer j

$$\frac{\partial D}{\partial w_{jk}} = \sum_n \sum_i \left[s_i^n - f(h_i^n) \right] f'(h_i^n) \frac{\partial h_i^n}{\partial s_j} \cdot \frac{\partial s_j}{\partial w_{jk}}$$

For layer k

$$\frac{\partial D}{\partial w_{kl}} = \sum_n \sum_i \sum_j \left[s_i^n - f(h_i^n) \right] f'(h_i^n) \frac{\partial h_i^n}{\partial s_j} \cdot \frac{\partial s_j}{\partial s_k} \cdot \frac{\partial s_k}{\partial w_{kl}}$$

Project 1 Due 2 weeks from Next Monday.