CMPE258

F22, HL

1/

August 23 (Tue)

First Day of the Class

1. Organizational meeting

"Green sheet"

Repo: github/hualili/opencv/deep-learning-2022s/

Email: hua.li@sjsu.edu

(650) 400-1116 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: CMPE255 & CMPE257

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. (Heavy Theoretical Book, Good Reference for OpenCV Algorithms. Good Theoretical Foundations)

5. Projects { Mandatory Assigned Project; Team Project (Mandatory)

4-Person Team. Presentation By the End of the Semester.

August 23 (Wed)

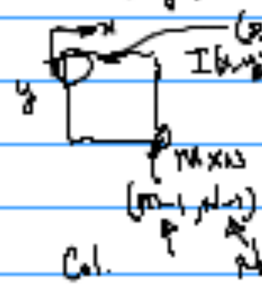
Note: 1. The lecture note will be posted on to the github.

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 Image → pdf → zip 2 pdf.

Example: (github: 2022F-103-)

MNIST Architecture for Handwritten Digits Recognition.



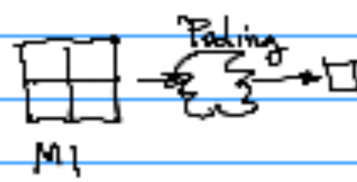
Single channel "gray scale" Image from the PPT. 28x28x1 → Square Image

Gray scale image → 1 Channel → 8 bit → [0, 255]

First Layer of the MNIST Architecture

n1 Channel/plane of the 1st Convolution Layer C1

Next, pooling → Reduction of Resolution → 2x2 becomes 1

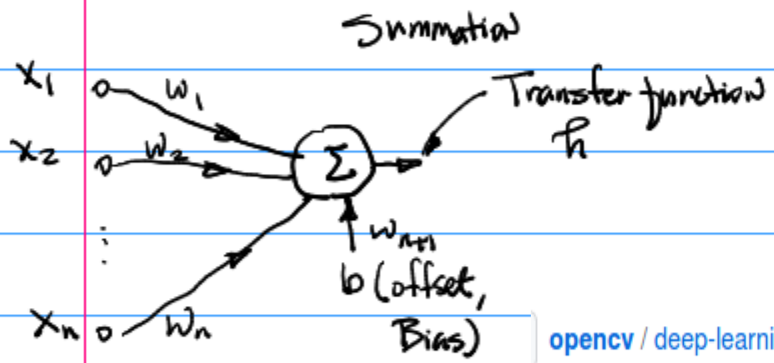


From the Architecture diagram:

C1 M1 → C2 M2 → Flatten → FFNN (FF)

To generalize the quick inspection of the the CNNs, we have to investigate the Behavior of Each Single Neuron as the Basic Building Block.

CmpE258
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

Input / Excitation in Vector Form: $X = (x_1, x_2, \cdots, x_n)$ ... (1)

Weights, links each excitation to the Neuron

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

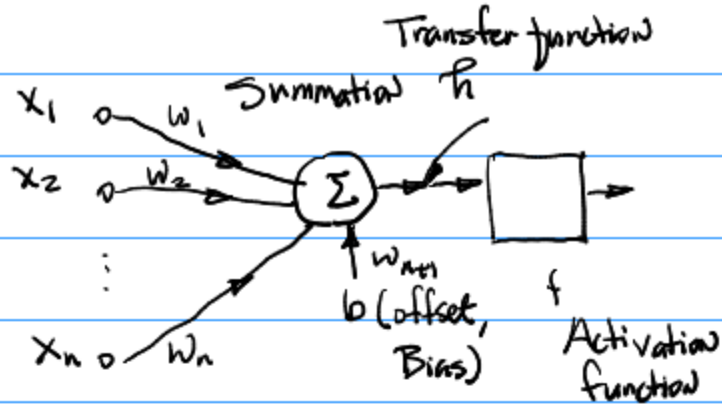
Ref 2. Code
https://github.com/hualili/opencv/blob/master/deep-learning-2022s/2022S-110b-%232019S-31-6mnist-numerals-ch02.py

$x_1 w_1 + x_2 w_2 + \ldots + x_i w_i + \cdots + x_n w_n + b w_{n+1} = h$

Example:

$\sum_{i=1}^{n} x_i w_i + b w_{n+1} = h(x_i w_i)$ or Simply $h(XW)$ ... (3)

$h(XW; b)$, $h(X, W)$



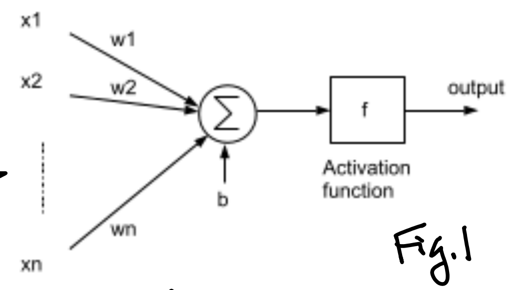


Fig. 1

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

$f(h(XW)) = f\left(\sum_{i=1}^{n} x_i w_i + b w_{n+1}\right)$ ... (4)

$h(XW) = \sum_{i=1}^{n} x_i w_i + b w_{n+1}$

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

Transfer function $h(\cdot)$.

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

Examples of Different Activation functions include RELU. A piecewise Linear.

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

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

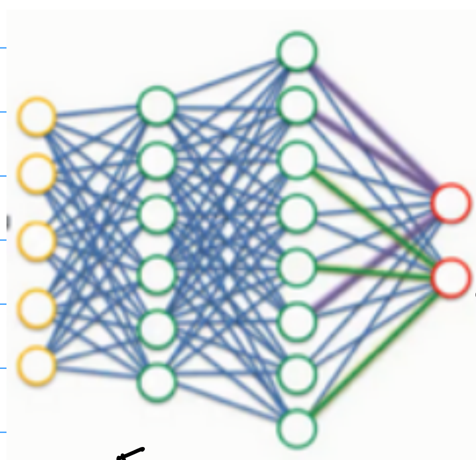
Note: Be Able to Build A Single Neuron per a technical Specification, Such as h(·), Activation f(·), Draw a Block

CMPE258

Aug. 30.

Consider the output of the Neuron
y from Eqn(1).

Output of A Single Neuron.
For multiple Neuron Output, see Fig.2



y_1

y_2

Fig.2

y_i ...(1)

Subscript: $i=1,2$
No. of Output at the Output Layer.

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

In practical Application,

y_i^j ...(2)

Superscript
$j=1,2,\cdots,P$ 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
Comparision of the Similarity or diffence between f and g.

$$\begin{cases} f-g \\ f/g \end{cases}$$

Difference Between Two Functions.
Take this Approach to define Loss function,

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

Ground Truth.

Output (prediction) from the Neuron

Outputs

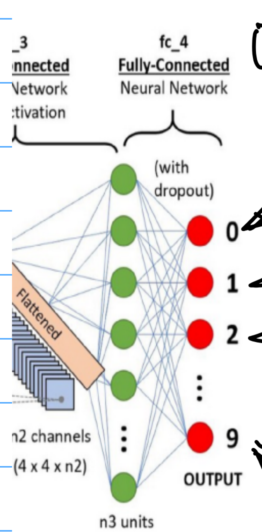


Fig3.

y_0

y_1

y_2

y_9

$\hat{y}_i$

$i=0,1,2,\ldots,9$

$y_i - \hat{y}_i$ ... (4-a)

To measure All the output for Each Training/Experiment

$$\sum_{i=0}^{q} (y_i - \hat{y}_i) \quad \text{... (4-b)}$$

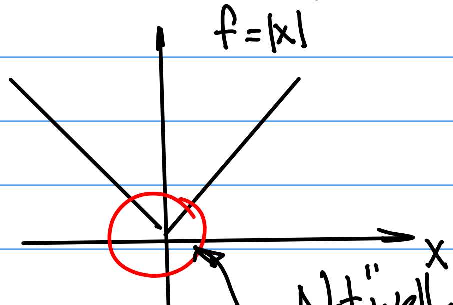
Expand this to Experiment/Training up to "P" Times

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

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

Fix: AbsoluteValue? $\rightarrow$ Squared Instead.

$f = |x|$



Not "well-behaved"!

"Well-Behaved" System (Function) $\rightarrow$ derivative / Partial Derivative up to order "K".

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

$\rightarrow$ J, or Φ

Ground Truth

$$L_{total} = \frac{1}{2} \sum_{j=1}^{P} (\tilde{y}^j - y^j)^2. \qquad (23)$$

For a Single Neuron @ the Output Layer

Training Based "Steepest Gradient Descent" SGD.

Example: Given A function $f(x) = x^2$, Find its Derivative

$$\frac{df}{dx} = 2 \cdot x$$

To get rid of Coefficient from the derivative,

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

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

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

Sept 1st (Th).

Ref:

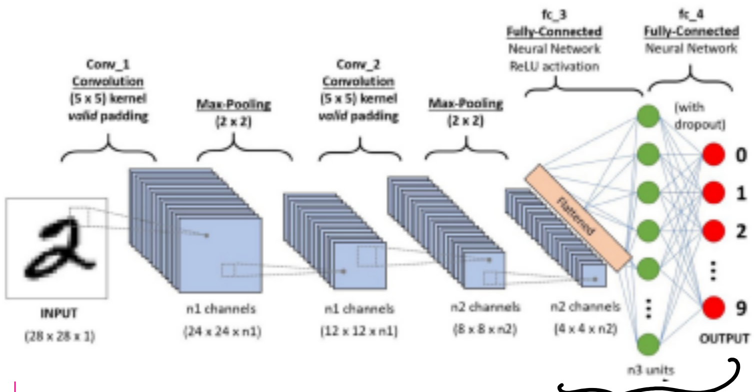
2022S-103a-notation-neuro-loss-function-2022-2-8.pdf

Example: Background on "Learning" of A NN.

2022F-103b-NN-Intro-Python-v5-2022-8-25.pdf

CMPE258

Sept 1st.



Fully Connected Feedforward NN

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

$L(\cdot) \to L(W_i)$

Independent Variables

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

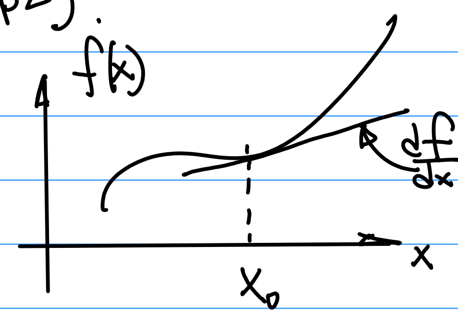
↓

Since the Loss function in Eqn (23) is formulated with a ground Truth y, this define 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 indication to give us the description of the function behavior Next Step Ahead (Very small tiny Step Δ).



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

↓ Intuition

$f(x+\Delta x) - f(x) \to > 0$ (derivate)

$\Delta x \approx 1$ unit

if the derivative is greater than 0

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.

CMPE258

Sept. 1st.

$$f(x_1,x_2)\begin{cases}\dfrac{\partial f(x_1,x_2)}{\partial x_1} \to \text{Single Neuron}\\[2ex] \dfrac{\partial f(x_1,x_2)}{\partial x_2}\end{cases}$$

In Case of Training They are "Weights" w_1, w_2.

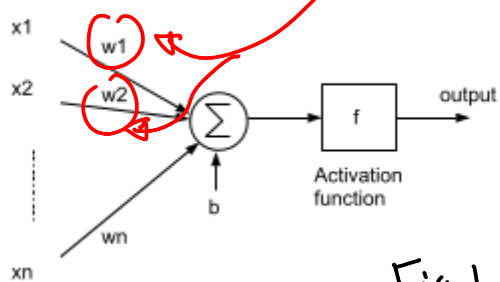


Fig. 1

$\frac{\partial f}{\partial x_1} \to \frac{\partial f}{\partial w_1}$ In the Context of Training.

Conclusion: Use the Partial Derivative With respect to the Weight w_i as an indicator to measure if the Loss function is got reduced or not.

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

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

Steps for Development of this technique:

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° Installation of TensorFlow. Version 2.0 or higher.

2° Screen Capture to show the installation is successful

Note: All Different Development Tools | Environments including google co-lab, 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.
4. Submission to CANVAS.
   ① Python Code;
   ② Original & Processed image Side by Side with your Name + SID.
   ③ Create One pdf file to Cover the Source Code, And Screen Captured Images.
5. Naming Convention

   HW_CV_First_LastName_Cmpe258_SID.Zip

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

Loss function

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

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

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

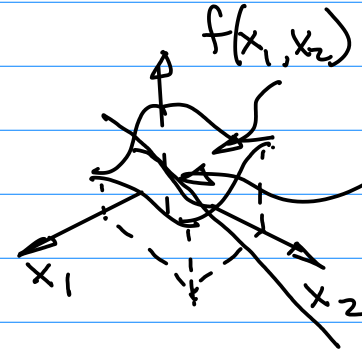
Higher Dimensional Function

$$\underbrace{f(x_1, x_2, \cdots, x_n)}_{\text{Weights, } w_1, w_2, \cdots, w_N} \quad \dots (2)$$

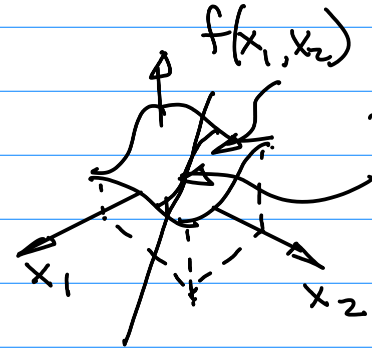
Partial Derivatives:

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

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



$\frac{\partial f}{\partial x_1}$ Changing rate of f w.r.t x_1



$\frac{\partial f}{\partial x_2}$, Changing rate of f w.r.t x_2

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

Consider the minization 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.

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \cdots \\ \frac{\partial f}{\partial x_i} \\ \cdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \quad \cdots (2a)$$

for n=2,

$$\nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} \quad \cdots (2b)$$

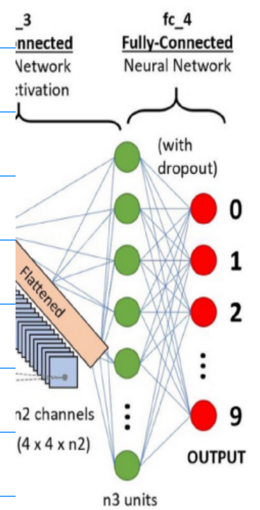
for n=3

$$\nabla f(x_1, x_2, x_3) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \end{bmatrix} \quad \cdots (2c)$$

d. On the Right hand side of Eqn(5):

(x_1^k, x_2^k) $\begin{cases} \text{Dimension } n=2,\ (x_1, x_2) \\ \text{Time Index "k", Superscript} \end{cases}$

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

$$-\nabla f = -\eta \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} \quad \cdots (3)$$

IV. Gredient Steepest Descent for Minimization

Conclusion:

a. Loss function f

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

b. n=2

e.g. $f(x_1, x_2, \cdots, x_n) \rightarrow f(x_1, x_2)$

c. Gradient

$\nabla f(x_1, x_2)$, or ∇f

Background:

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

$$f(x) = \underbrace{\text{Constant}}_{\text{Term}} + \underbrace{\text{A Linear}}_{\text{Term}} + \underbrace{\text{A Quadratic}}_{\text{Term}} + \text{A Cubic Term} + \cdots \quad \cdots (4)$$

CmpE258

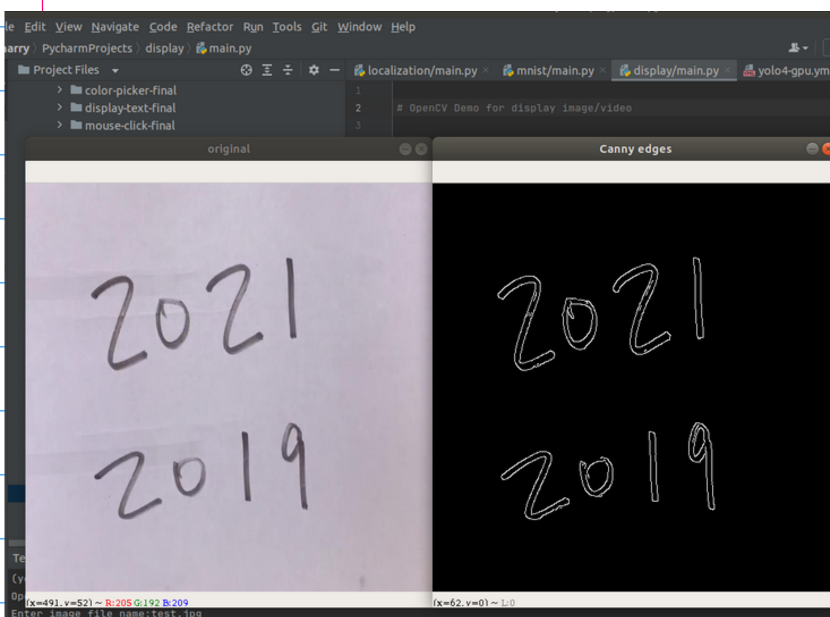
Sept 6, 22

Taylor Expansion:

$$f(x) = f(x_0) + \frac{df}{dx}\cdot(x-x_0) + \frac{d^2f}{dx^2}(x-x_0)^2 + \dots + R_n(x) \quad \dots (4)$$

$$f(x_1,x_2) = \underbrace{f(x_1,x_2)\Big|_{\substack{x_1=x_{10}\\ x_2=x_{20}}}}_{\text{Constant}} + \underbrace{\frac{\partial f}{\partial x_1}(x_1-x_{10}) + \frac{\partial f}{\partial x_2}(x_2-x_{20})}_{\text{Linear Term}} + \underbrace{\frac{\partial^2 f}{\partial x_1^2}(x_1-x_{10})^2 + \frac{\partial f^2}{\partial x_2^2}(x_2-x_{20})^2}_{\text{2nd Order}} + \dots + \underbrace{R_n(x_1,x_2)}_{\text{Higher Order Terms}} \quad \dots (5)$$

Note: The screen capture for your homework reference.



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

Step 1. Taylor Expansion → Step 2. Simplify the Taylor Expansion By just using up to the Linear terms → Step 3. Re-arrange the Taylor Expansion in the form of Training formula (In Eqn(5), in Handout 1) → Step 4. Analyze the re-arranged formula, to Reach the Observation which lead to the Conclusion, e.g., using gradient descent, we can Redefine the Loss

Sept. 8 (Thu)

Note: 1° 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 $n=2$

CmpE258

Sept. 8, 22

function through each step of the trainning.

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

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

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

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

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

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

Therefore,

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

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

$$\Downarrow$$

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

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

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

Loss function at the Current Step.

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 (e.g. 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

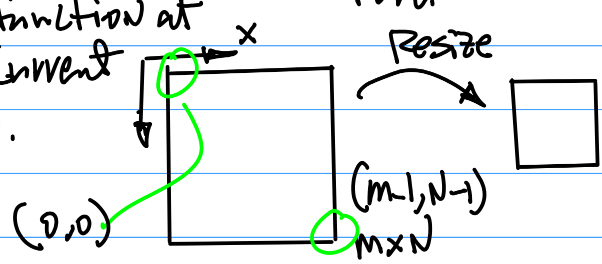
```
#main(sys.argv[1:])
window_name = 'Display Image'

imageName = sys.argv[1] #get file name from command line

src = cv2.imread(imageName, cv2.IMREAD_COLOR)

if src is None:
    print ('Error opening image!')
    print ('Usage: pdisplay.py image_name\n')

ind = 0

while True:
    cv2.imshow(window_name, src)

    c = cv2.waitKey(500)
    if c == 27:  #ESC
```

a.

b.

c.

Note: OpenCV Resize Function.



CmpE258
Sept. 8, 22


```python
import numpy as np
#import argparse
import cv2

img = input('Enter image file name:')

image = cv2.imread(img, cv2.IMREAD_COLOR)

if image is None:
    print('Error opening image!')
    print('Usage: pdisplay.py image_name\n')

image = cv2.resize(image, (512, 512))

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray,100,200)
```

a., b.

a. Conversion to Gray-scale image;
b. Canny Edge Detection.

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

$V = (V_1, V_2)$

$\downarrow$

$X = (x_1, x_2)$

$\downarrow$

$X_i = (x_{i1}, x_{i2})$

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

X_1
X_2
$\vdots$

Signs	Mij	Mpq	Sign
V1	133	65	Stop
V2	160	72	Right
V3	152	70	Right
V4	120	60	Stop

b. Ground Truth y_i Corresponds to X_i

c. Preprocessing of the dataset

Name	x_1 Weight (minus 135)	x_2 Height (minus 66)	Gender
Alice	-2	-1	1
Bob	25	6	0
Charlie	17	4	0
Diana	-15	-6	1

"Shift" Down the data w.r.t "0"

You can find the mean for x_i, $i = 1, 2$, then "Shift" the Data Accordingly By Subtracting it from the mean.

make the ground truth Value as Numerical value for easy handling.

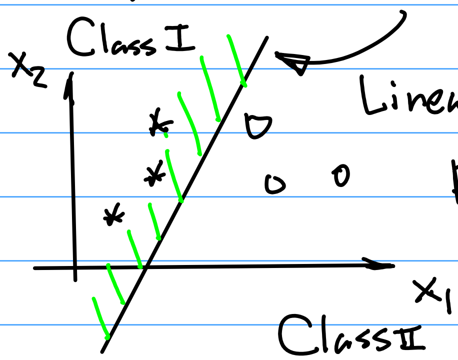
Now, Consider Design for a Simple feedforward NN.

Step 1. Match the Dimension of the Feature Vector X_i to the Input Neurons

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

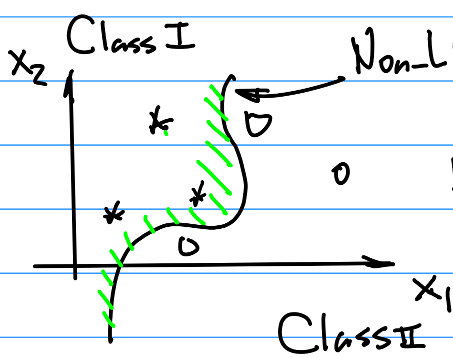
Also, Determine the Output Layer.
Inspection of the Nature of this Prediction, Hence, Choose One Single Neuron Output

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



Linear Function

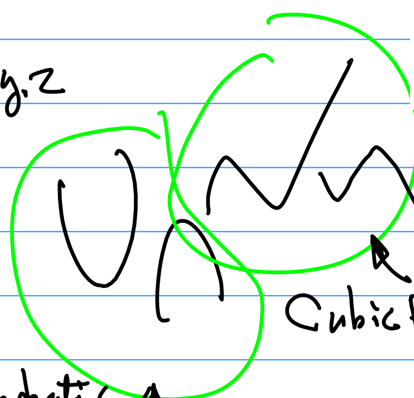
Fig. 1.



Non-Linear Function

Fig. 2

No. of Hidden Layers →

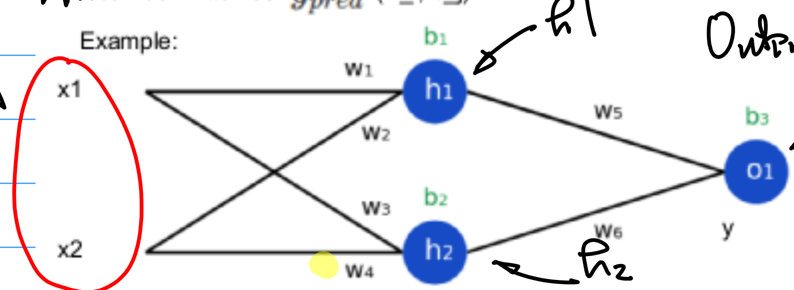


Quadratic

Cubic Functions

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

X Input

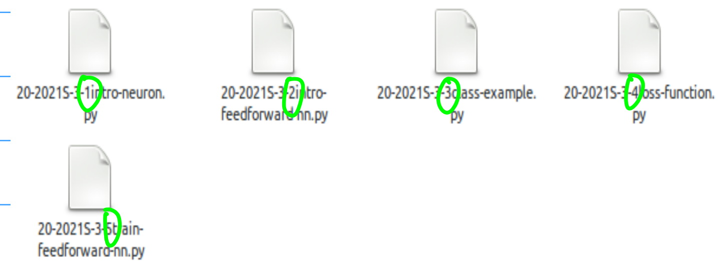


h1

h2

Output

Now, Let's use Python Code to Build this Simple FFNN



Define An Activation function.

$$y = f(h(w_i x_i; b)) = f(XW)$$

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

a. Range [0,1]

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

b. Excitation Range.
$f(h(XW))$

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

Note: A Sigmoid Function.

A sigmoid function is a

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

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

CmpE258
Sept 13, 22

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

Define the Loss function

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

```
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 $\tilde{y}_i$

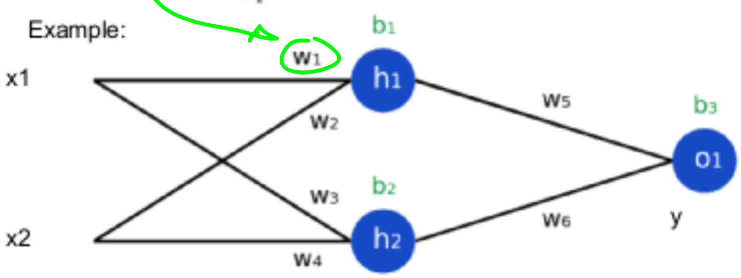
Note: The Derivative Code in this example Allows Training Implement By Gradient Descent.

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()
```

it can be written as y_{pred} (w_, b_).

Example:




```
  # 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
```

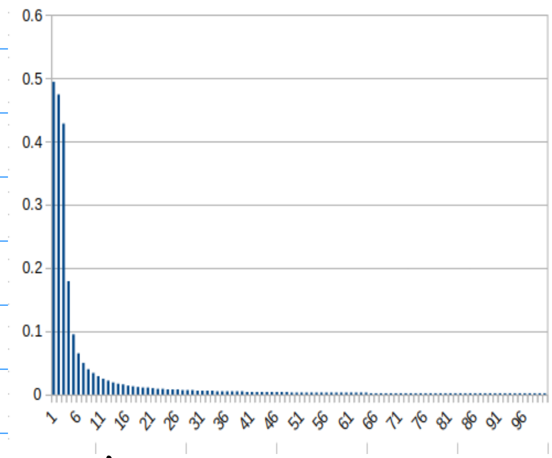
Define the F.F. Neural Network.

Output:

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

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



Sept 15 (Thu)

Today's Topics:

1° Preliminary Sample Code (Python)

2° Preprocessing Technique (Computer Vision / OpenCV) for Deep Convolutional NN. HandWritten Digits Recognition

CmpE258
Sept. 15, 22

Ref: From Python Sample Code on the github

```
def train(self, data, all_y_trues):
```

a. Training will take the input from here

```
#----------------------------------
# Define dataset and all_y_trues
#----------------------------------
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
```

loss is being computed

```
      d_L_d_ypred = -2 * (y_true - y_pred)
```

then, compute partial derivatives to form gradient

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

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

```
      self.w1 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w1
      self.w2 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w2
      self.h1 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_h1
```

b. Initialization. η

```
    learn_rate = 0.1
    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. $\underline{X}(x_1, x_2)$) is substituted into the code to evaluate Transfer function $h(w,x)$, then evaluate the Activation function $f(h(w,x)) = \left.\dfrac{1}{1+e^{-x}}\right|_{x=h(w,x)}$

Compute total Loss function

```
    # --- Calculate total loss at the end of each epoch
    if epoch % 10 == 0:
      y_preds = np.apply_along_axis(self.feedforward, 1, data)
      loss = mse_loss(all_y_trues, y_preds)
      print("Epoch %d loss: %.3f" % (epoch, loss))
```

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

Note: the ground truth is given in the code

```
    # --- Do a feedforward (we'll need these values later)
    sum_h1 = self.w1 * x[0] + self.w2 * x[1] + self.b1
    h1 = sigmoid(sum_h1)
```

```
all_y_trues = np.array([
  1, # person A
  1, # person A
  1, # person A
  0, # person B
  0, # person B
  0, # person B
])
```

Annotation

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

```
    sum_o1 = self.w5 * h1 + self.w6 * h2 + self.b3
    o1 = sigmoid(sum_o1)
    y_pred = o1
```

CmpE258
Sept. 15, 22

Example: Pre-processing Techniques for Handwritten Digits Recognition.

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

a. ROI: Bounding Boxes

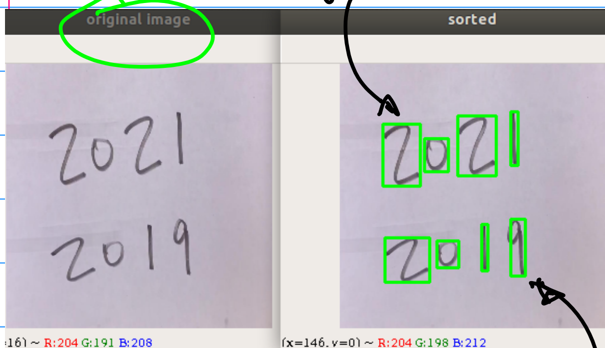


Fig. 1.

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 $I(x,y,t) \rightarrow I(x,y)$ @ time t ... (1)

c. Each Frame of A color image $I(x,y)$ Consists of 3 channels. Or, Primitive Color planes, red, green, Blue

In OpenCV, (Color) Image Planes is Organized as B, G, R.

$I_r(x,y) \in [0,255]$, $I_g(x,y) \in [0,255]$, $I_b(x,y) \in [0,255]$ ... (2)

8 bit
$2^8 = 256$

Color Image Example in Fig. 1. $I(x,y)$ $\rightarrow$ Gray Scale Image. $I_{gray}(x,y)$

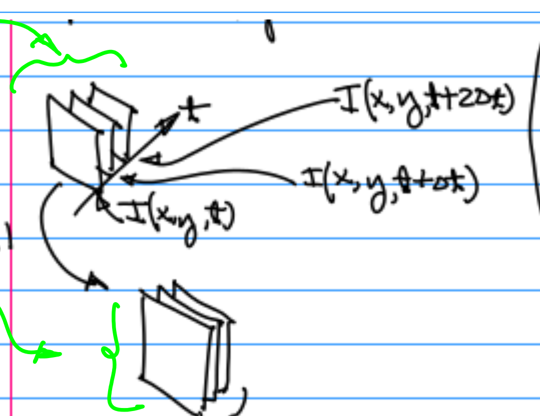
$$I_{gray}(x,y) = \frac{1}{3}\left(I_r(x,y) + I_g(x,y) + I_b(x,y)\right) \quad \dots (3)$$

$\downarrow$

Binary Image.

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

Fig. 1



3 Image Planes (r, g, b)

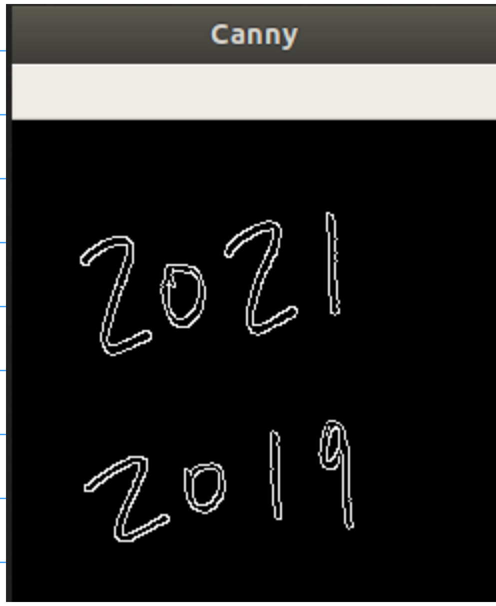
Pixel depth, Color Depth: 8 bits for Each Color plane

$r \in [0,255]$, $g \in [0,255]$, $b \in [0,255]$

CmpE258
Sept.15, 22

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

From Eqn(4)



Example:

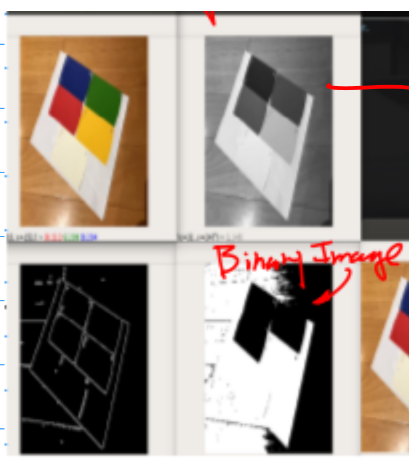
2022S-108c-example-binary.pdf

Sept. 20 (Tuesday)

Note: 1° Next Class we will have in-Class exercise, Requires 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.

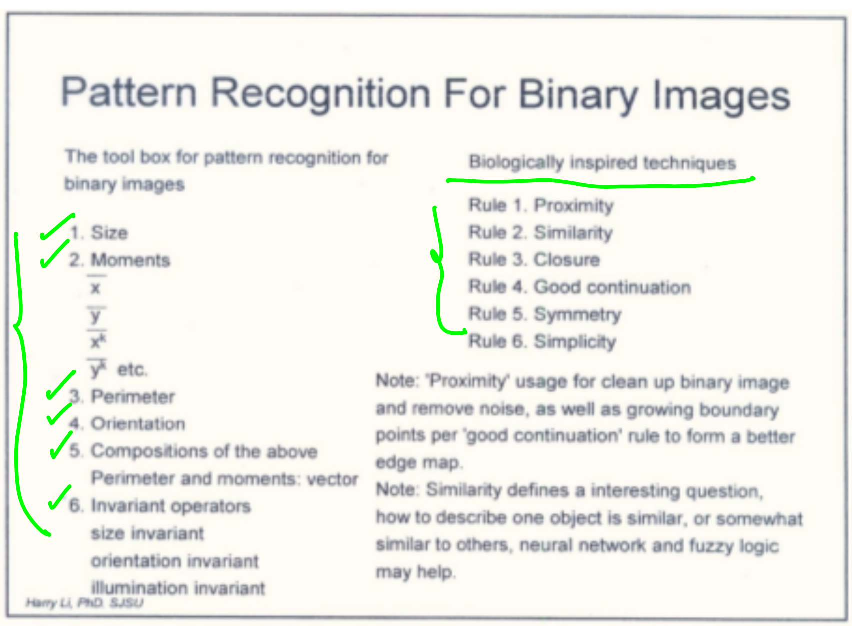
3° For the Semester-Long Team Project, we will post the Rubrics & Requirements on Line (on CANVAS).



By Eqn(4)

2022S-108-Intro-Binary.pdf

Operators for Preprocessing (1~6)



2022S-108-Intro-Binary.pdf

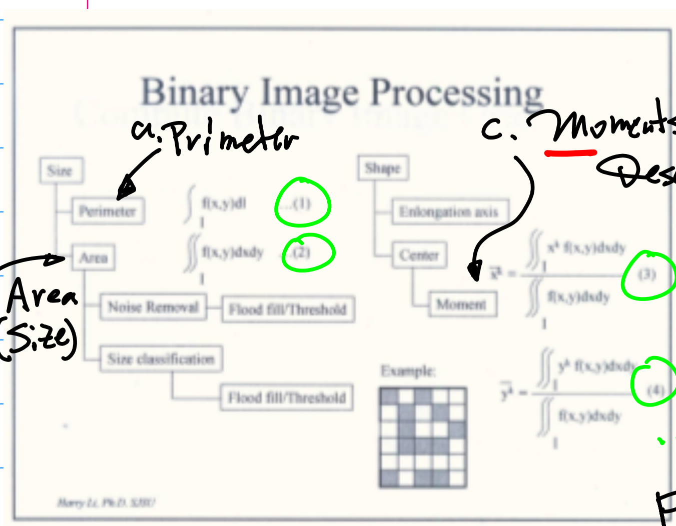


Fig.1

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

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

2022S-108c-example-binary.pdf

$f(x,y)$

70	70	80	159
75	85	149	121
90	210	101	90
211	209	115	85

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.

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

for (0,0), $f(0,0) \geq 135$, $\because f(0,0) = 70$,

$f(0,0) < T$, Hence $B(0,0) = 0$;

for (0,1), Similarly, $f(0,1) = 70$, Hence, $B(0,1) = 0$;

...

...

for (3,0), $f(3,0) = 211 \geq T$, Therefore $B(3,0) = 255$

e.g. often this 255 in Hand Calculation Can be replaced by 1. (Scaling factor)

for (3,1), $f(3,1) = 209 \geq T$, Therefore $B(3,1) = 255$

K_2 / K_1

0	0	0	255
0	0	255	0
0	255	0	0
255	255	0	0

M x N

No. of Pix. s per Row

No. of Rows Per frame.

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

Definition 1. (AREA/Size)

$$\iint_{\Omega} B(x,y)\,dxdy \quad \ldots (1)$$

Digitized Image / Discrete Function

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

Hence,

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

$K_2 \rightarrow$, $K_1 \downarrow$

0	0	0	255
0	0	255	0
0	255	0	0
255	255	0	0

Sol:

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

$$= \sum_{y=0}^{M-1}\left[B(0,y)+B(1,y)+\cdots+B(N-1,y)\right]\Big|_{N=4}$$

$$= \sum_{y=0}^{M-1}\left[B(0,y)+B(1,y)+\cdots+B(3,y)\right]$$

$$= \cancel{B(0,0)}^{0}+\cancel{B(1,0)}^{0}+\cancel{B(2,0)}^{0}+B(3,0)$$

$$+ \cancel{B(0,1)}+\cancel{B(1,1)}+B(2,1)+\cancel{B(3,1)}$$

$$+ \cdots$$

$$+ B(0,3)+B(1,3)+\cancel{B(2,3)}+\cancel{B(3,3)}$$

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

Hence, the size is 5 "

Example: Computation of moments

Definition 2.

$$\bar{x}^K = \frac{\iint_\Omega x^K B(x,y)\,dx\,dy}{\iint_\Omega B(x,y)\,dx\,dy} \quad \ldots (2)$$

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

$$\bar{y}^K \quad \frac{\iint_\Omega y^K B(x,y)\,dx\,dy}{\iint_\Omega B(x,y)\,dx\,dy} \quad \ldots (3)$$

where $K=1,2,\ldots,K$. when $K=1$, $\bar{y}$ is a Centroid along y-axis.

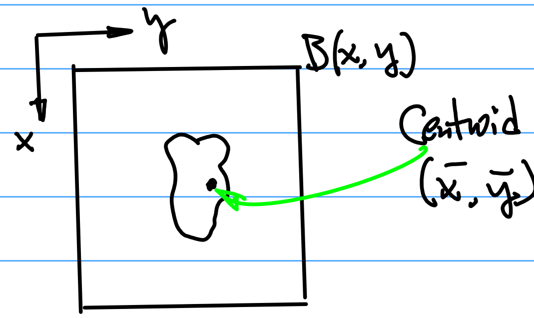
In Addition, extend Eqn (2) & (3) to the following.

$$m_{pq}(x,y) = \frac{\iint_\Omega (x-\bar{x})^p (y-\bar{y})^q B(x,y)\,dx\,dy}{\iint_\Omega B(x,y)\,dx\,dy} \quad \ldots (4)$$

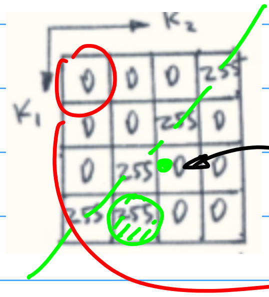
$$m_{pq}(x,y) = \frac{\sum_{y=0}^{M-1}\sum_{x=0}^{N-1} (x-\bar{x})^p (y-\bar{y})^q B(x,y)}{\sum_{y=0}^{M-1}\sum_{x=0}^{N-1} B(x,y)} \quad \ldots (4\text{-}b)$$

where $p,q = 0,1,2,\ldots K$

"Hu-moments" ← Very Good Way to Describe Object Shape



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



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

Sept. 22 (Thu).

Example: Pre-Processing Technique.
$\bar{x}, \bar{y}$ Centroid.

From the previous Note.

$$\bar{x}^k = \frac{\iint_\Omega x^k B(x,y)\,dx\,dy}{\iint_\Omega B(x,y)\,dx\,dy} \quad \ldots (2)$$

Rewrite Eqn (2) as

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

AREA $= 5$

Now,

$$\sum_{y=0}^{3}\sum_{x=0}^{3} x\,B(x,y)$$
$$= \sum_{y=0}^{3}\left[0 \cdot B(0,y) + 1 \cdot B(1,y) + 2 \cdot B(2,y) + 3 \cdot B(3,y)\right]$$

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 $(0,0)$.

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

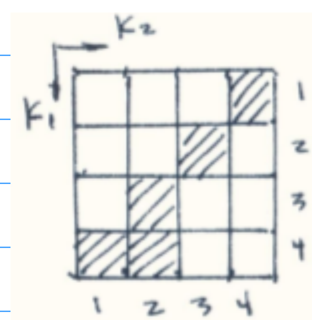
$$\sum_{y=1}^{4}\sum_{x=1}^{4} x\,B(x,y)$$
$$= \sum_{y=1}^{4}\left[1 \cdot B(1,y) + 2B(2,y) + 3 \cdot B(3,y) + 4 \cdot B(4,y)\right]$$
$$= \overset{0}{\cancel{B(1,1)}} + 2 \cdot \overset{0}{\cancel{B(2,1)}} + 3B(3,1) + 4B(4,1) +$$
$$\cdots$$
$$\cdots$$
$$B(1,4) + 2 \cdot B(2,4) + 3B(3,4) + 4B(4,4)$$
$$= 14$$

Therefore

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

For Visualizing the Result

Keep this for Continued Computation in your program.



Similarly,

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

See the handout on the Class github, we have

$\bar{y} = 12/5 = 2.4 \simeq 2$

For Visualization Purpose Only.

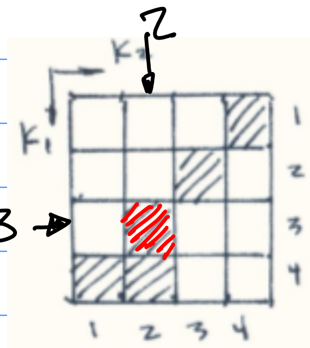
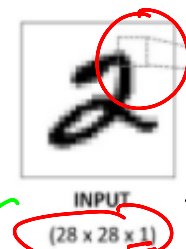


Fig. 1.

Example: Consider 2D Convolution of a given Image.



$M \times N$, $M = N = 28$

A single channel.
"A grayscale Image"

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

$I(x,y)$ $M \times N$ $(M=N=4)$

$g(x,y)$

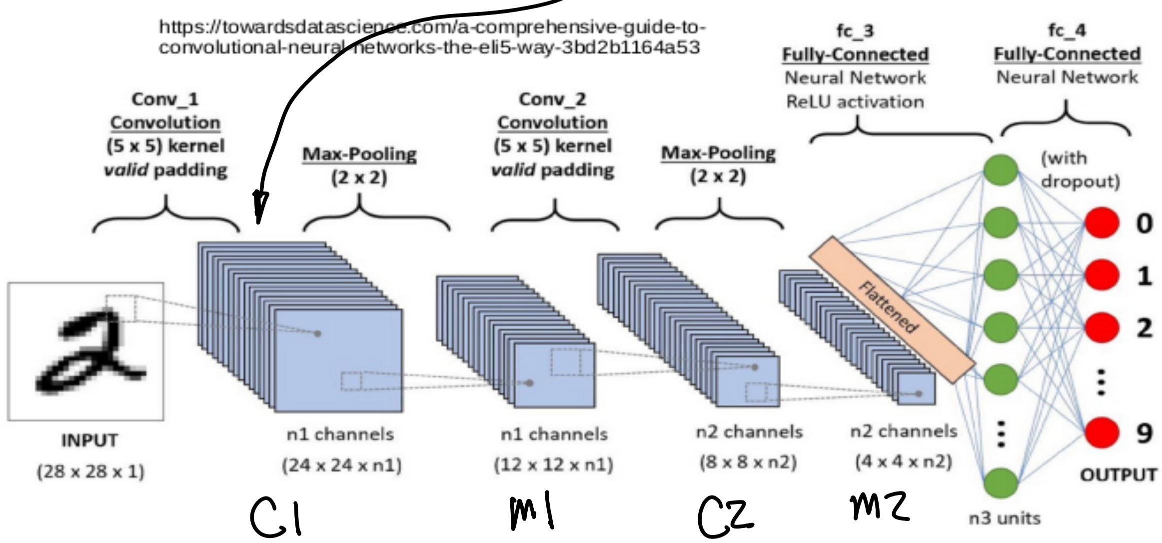
1	0	-1
1	0	-1
1	0	-1

A Kernel
$K \times K$

Fig. 2

Now, consider Convolution Technique.

Illustration of A CNN for Digits Recognition



CMPE258

Sept.22,22

About the Kernel:

K = "odd" Number.

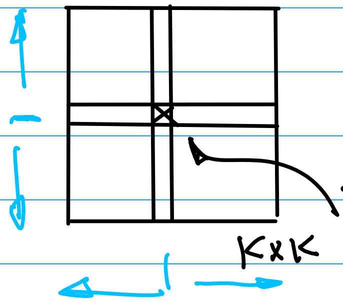


Fig.3

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 2D 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, 2D Convolution is defined as

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

$$= \iint_{\Omega} I(u,v)\, g(x-u,\, y-v)\, du\, dv \quad \cdots (1)$$

For Discrete Cases, we have

$$\sum_{v=0}^{N-1} \sum_{u=0}^{M-1} I(u,v)\, g(x-u,\, y-v) \quad \cdots (2)$$

a. Summation Index u, v,
for $u = 0, 1, 2, \cdots, M-1$,
$v = 0, 1, \cdots, N-1$.
Resolution of The Image $I(x,y)$ is $M \times N$.

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 → pixel of Interest;
"odd" Number makes No. of col. (Right & Left) equal, No. of Rows (Above & Beneath) equal → Counting the Rows, Col. in a "Un-Biased" fashion.

c. $g(x,y)$ ① → Eqn (2) ②
Originally Outside Eqn (2)

③ $g(u,v) \rightarrow g(-u,-v)$
↓ ④
$g(C_1-u,\, C_2-v)$
↓ ⑤
$g(x-u,\, y-v)$

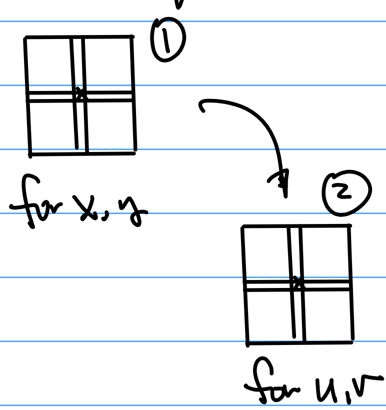
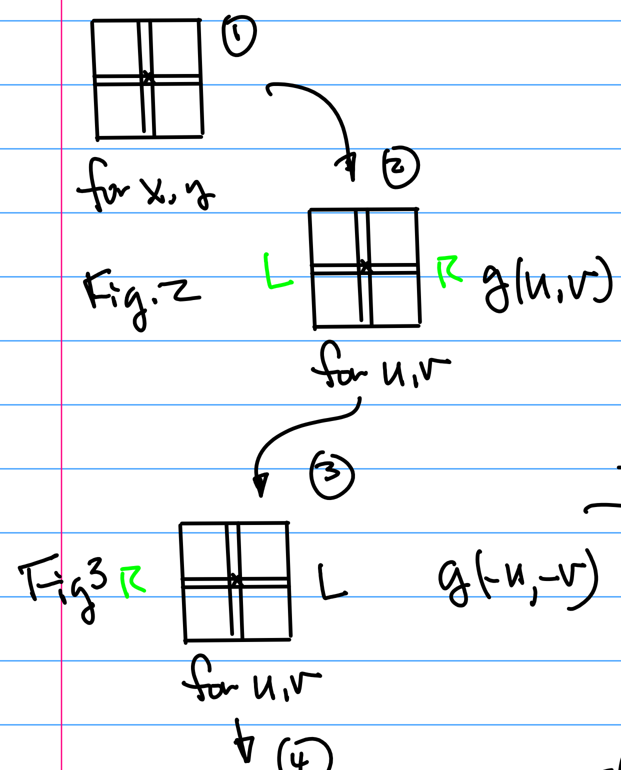
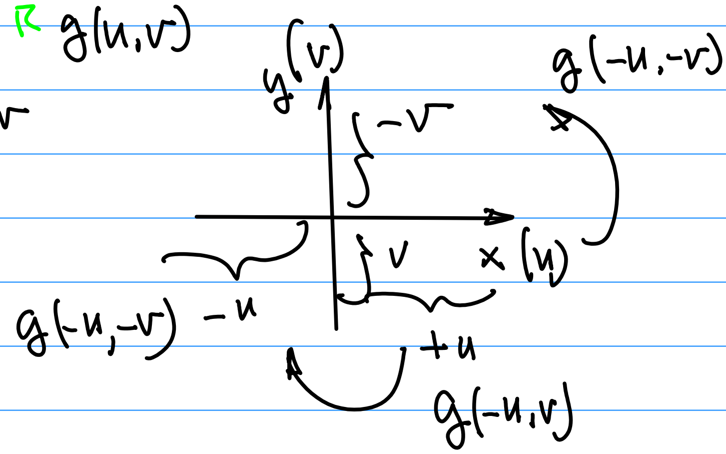
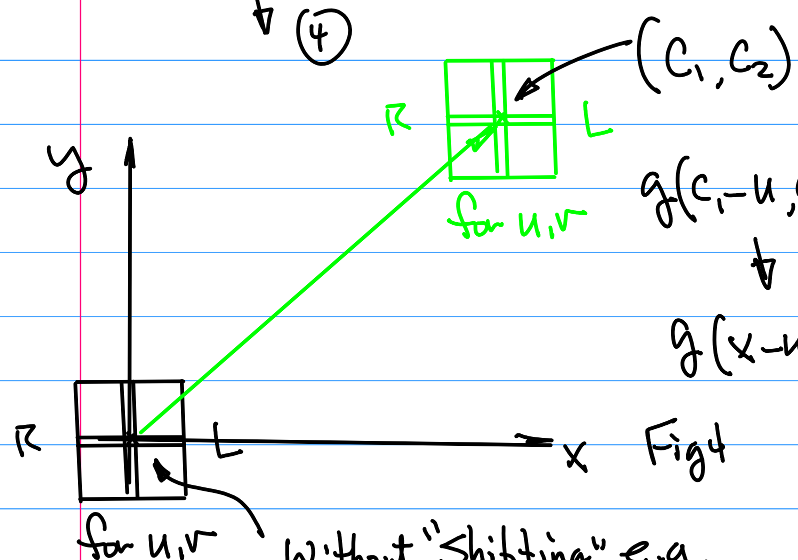


Fig.1







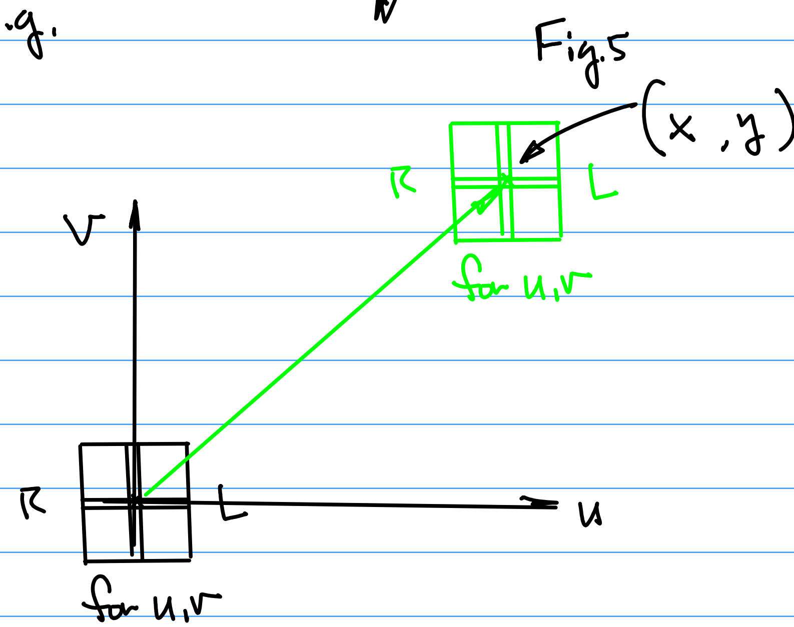
$g(c_1-u, c_2-v)$

Let $c_1 = x$, $c_2 = y$.

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

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

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

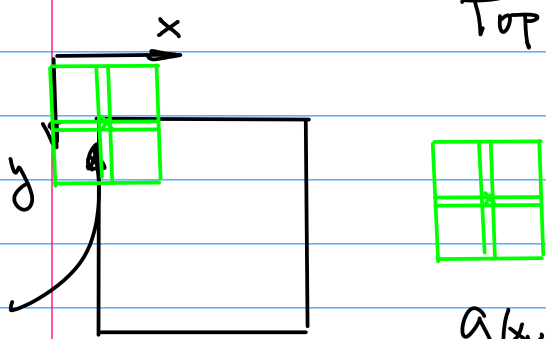


Note: To perform 2D Convolution defined in Eqn(2) with the Kernel $g(x, y)$.

"Slide" the Kernel at (0,0) of place $I(x,y)$ Top Left Corner.

x

y



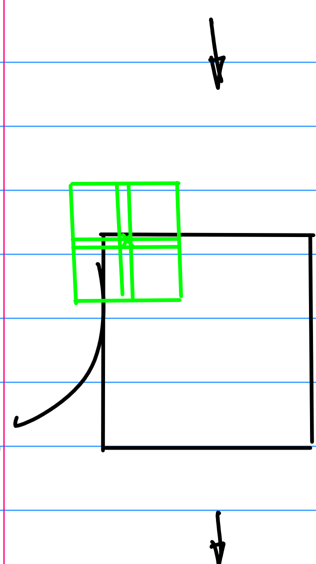
(0,0)

Fig. 6

$g(x,y)$ Kernel of K-by-K

$I(x,y)$

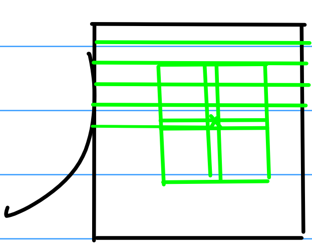
Initial position of 2D Convolution.



(0,0)

Slide the Kernel One pixel at Time from Left to Right

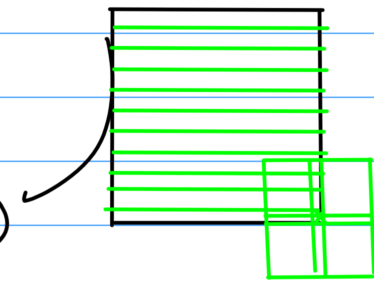
Fig. 7



(0,0)

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

Fig. 8



(0,0)

Note: The Computation of 2D Convolution is defined as Sum of Products. See Eqn(2)

$$\left\{ \begin{array}{l} \underbrace{I(u,v)\, g(x-u, y-v)}_{\text{multiplication.}} \\ \underbrace{\sum_{v=0}^{N-1} \sum_{u=0}^{M-1}}_{\text{Addition}} \end{array} \right.$$

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

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

$I(x,y)$ M x N

$g(x,y)$

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

A Kernel K x K

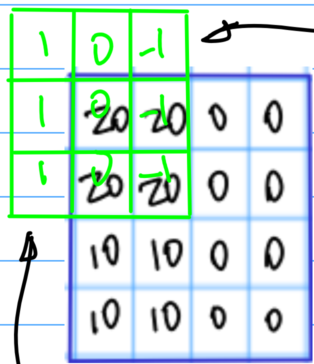
Fig. 9

CmpE258

Sept. 27, 22

Step1. Place g(x,y) at (0,0) on the Image.
Kernel

PAD 0's



$(0,0): 0\times 20$

$(1,0): -1\times 20$

$(0,1): 0\times 20$

$(1,1): -1\times 20$

Add Together

PAD 0's

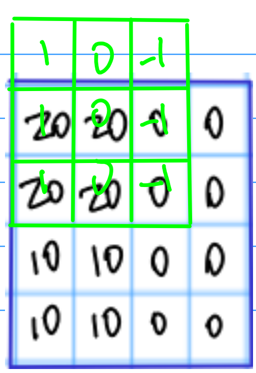
$0\times 20+(-1)\times 20+0\times 20+(-1)\times 20$

$= 0-20+0-20=-40$

Convolution IN Progress

-40			

Step2.



$1\times 0+0\times 0+(-1)\times 0$

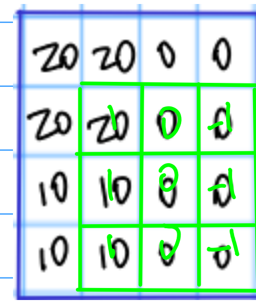
$+\ 1\times 20+0\times 20+(-1)\times 0$

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

Convolution IN Progress

-40	40		

Step3. 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.



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

$+\ 1\times 10+0\times 0+(-1)\times 0$

$+\ 1\times 10+0\times 0+(-1)\times 0 = 40$

Convolution IN Progress

-40	40	"	"
"	"	"	"
"	"	40	

CmpE258
Sept. 28, 22

Sept. 28.

Note: 1° Brief update from Each team, Status, Activities. On Thursday (A week from today)

2° First Project on DCNN (MNIST) for Hand written Student ID recognition System Design is due on Oct. 10th.

Kernel (k x k) is placed inside Image Plane. see Fig. 1

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

Fig. 1.

↓

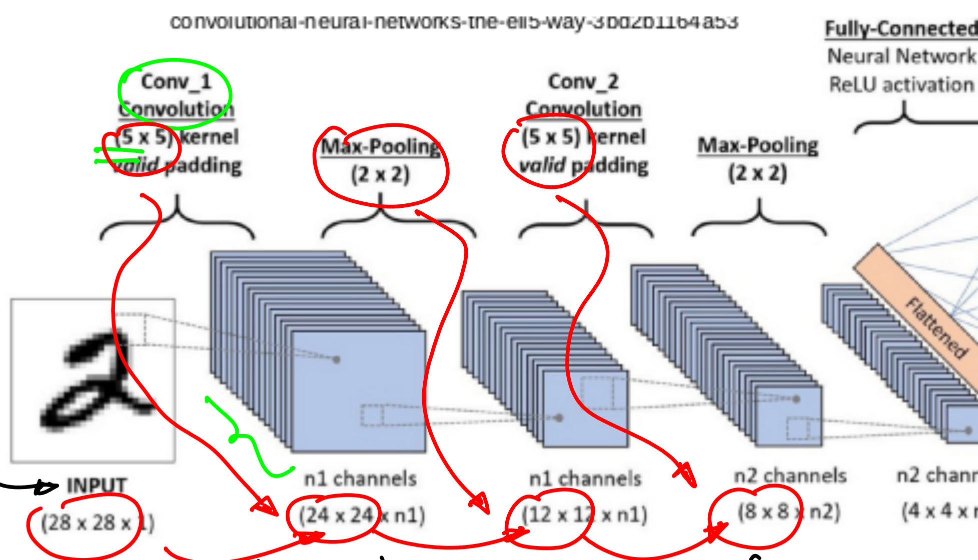
Reduction of the Rows (Top & Bottom), Col. (Left & Right) in such a manor Non of them (Rows, Cols) placed outside the Boundary of the Image.

↓

K=3, Image 28x28 Therefore Output Dimension: 24x24

Example:

a. Input Single Channel Image → Gray Scale



Feature planes Result of 2D Convolutions.

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

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

Note 1. Resolution Change. Input: 28x28
Output (Convolution) 24x24

Image $\underline{M \times N}$, 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)

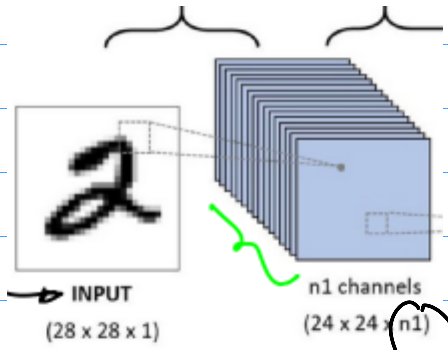
$$[M-(K-1)] \times [N-(K-1)]$$

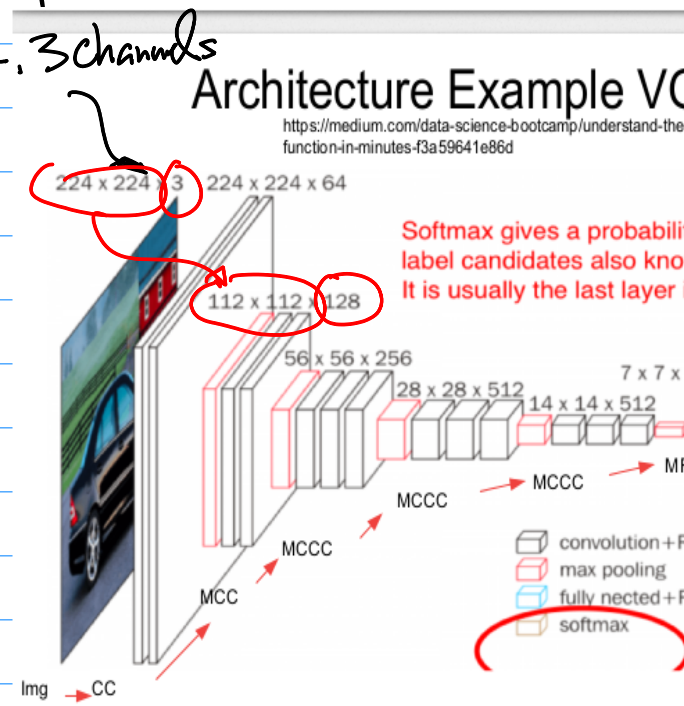
CmpE258

Sept. 28

Example:

a. 3 channels





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

Note 3. Max Pooling

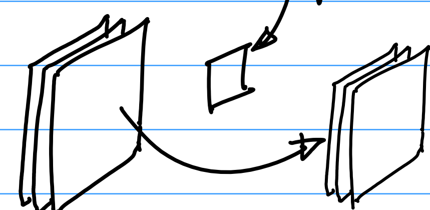
125	220
10	0

Pixels

Max Pooling $\rightarrow$ $\boxed{220}$

Max{125, 220, 10, 0} = 220

Kernel K x K



3 Channels

3 channels with 1 kernel Convolution.

Biological Inspiration from Human Visual Perception System, Early Vision.

Retina. 3 types of photo Receptors.

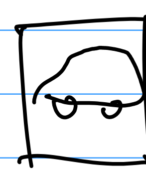
optic ~1 million

Rods, Bipolar Cells, Ganglion Cells.

~110-120 million

Optional (Not Required)

Example:



I(x,y)

Feature Extraction "Edge" map.

Boundary of the object.

Change 2D Convolution "Step Size" Stride to Realize Both Function Convolution & Pooling.

Build A Kernel for the purpose of Extraction of Boundaries.

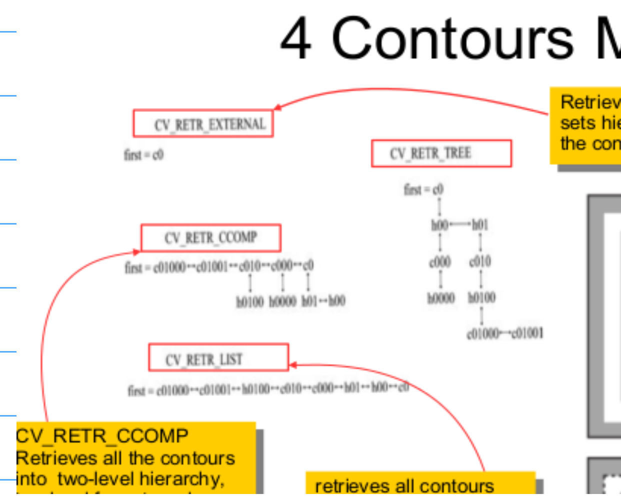
Object Boundaries → Sudden Change of Image Intensity. $I(x,y)$

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

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

Partial Derivative ← How to Detect Abrupt Change of the Intensity.

$$\frac{\partial}{\partial x} I(x,y) = \lim_{\Delta x \to 0} \frac{I(x+\Delta x, y) - I(x,y)}{\Delta x} \quad \cdots (2)$$



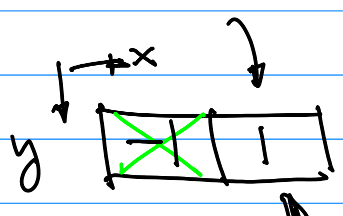
Similarly for y.

$$\frac{\partial}{\partial y} I(x,y) = \lim_{\Delta y \to 0} \frac{I(x, y+\Delta y) - I(x,y)}{\Delta y} \quad \cdots (3)$$

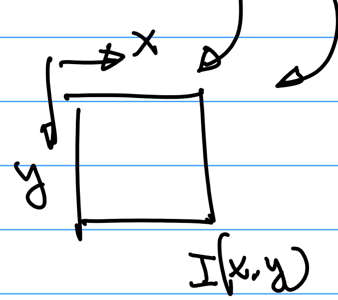
From Eqn(2), on Digital Image $\Delta x = 1$

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

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



kernel



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 Python Program. To Capture Web CAM Video → Step2 Preprocessing Task. Goal: To Build Bounding Boxes for the Digits. (One Box for One digit)

One Blank Printer Paper + Marker

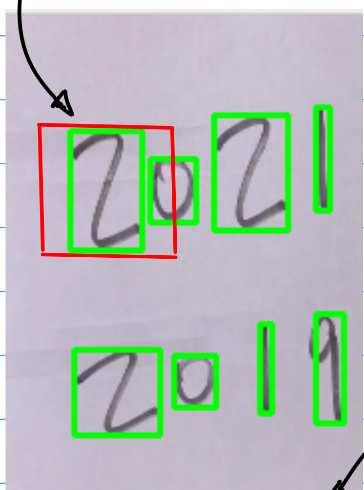
Write 4 Digits of your SID

LoG(x,y) Kernel, Laplace of Gaussian Kernel.

Now, Consider Contour Analysis as Pre-processing Technique

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 28x28

28x28x1

Step4. To Run the MNIST Sample Code. (Sample code from google T.F. Tutorial, or from the Class Recommendated, 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.

Step5. Save the trained DCNN. e.g. all weights & Biases Once trained, are Saved into A file for the future use. as well as its architecture

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.

Dense Feedforward NN
64 Neurons

1st Max-pooling Layer

The MNIST architecture consists of C_1M_1 C_2M_2 C_3 $F_{576}D_{64}D_{10}$, and it is illustrated below.

Flatten Layer

No. of Neurons.

10 Neurons

1st Convolutional Layer

F_{576} Flatten Layer as output Layer from C_3

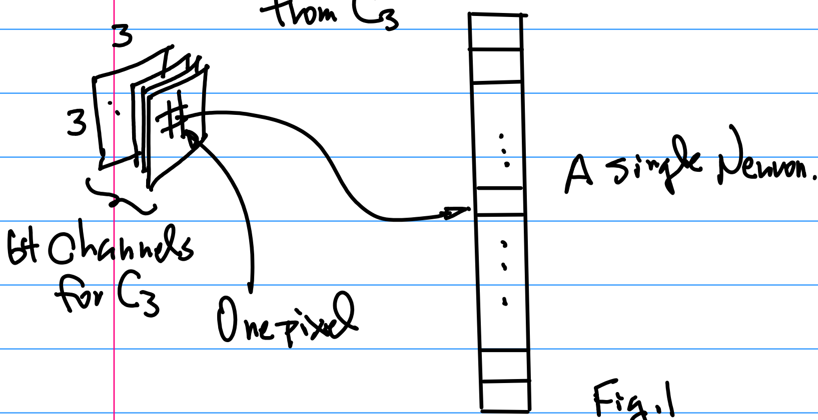


Fig. 1

Video Source from A web CAM.
the 2nd part of the Homework:
Hand Calculation of (1) Binary Images. & (2) 2D convolutions.

Team Project Quick Status checking, will continue on Thursday.

Oct. 6 (Thurs).

Note: 1° Project Announcement for SJSU Hand Written Digits Recognition System, Due Oct. 23rd, official Announcement will be posted on CANVAS Before End of the Day Friday;
2° Midterm after the project.

Example: ON MNIST.

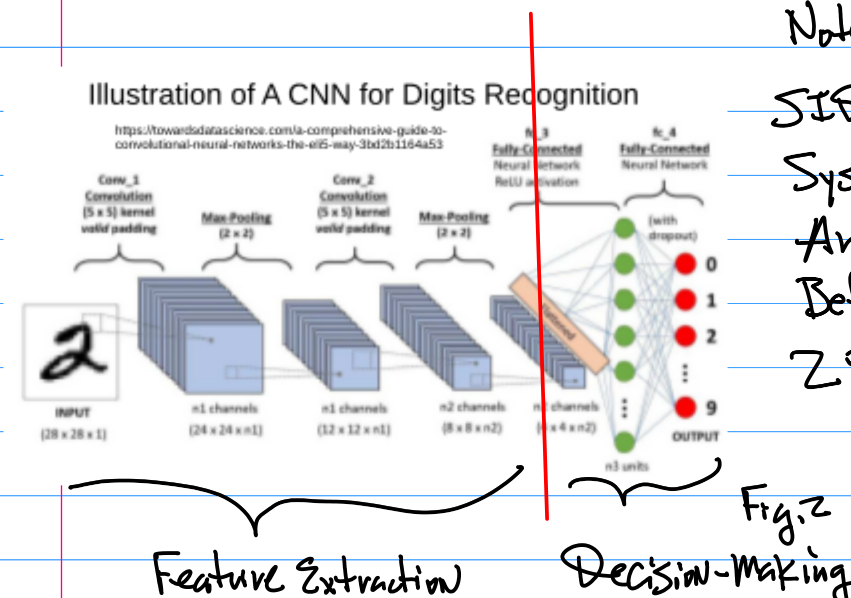


Fig. 2

Note: No. of Neurons @ the Flatten Layer = 576

```
conv2d_3 (Conv2D)          (None, 3, 3, 64)
flatten_1 (Flatten)        (None, 576)
```

No. of the pixels @ C_3 Layer : $\underbrace{3 \times 3}_{\text{Resolution}} \times 64$

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

CmpE258

Oct.6,22

29

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

b.

```
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'))
```

KxK Kernel

No. of Kernels

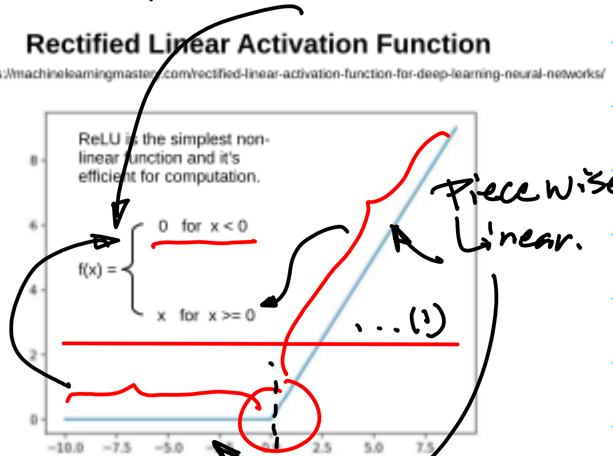
No. of the Neurons

① models.Sequential

② model.add( )

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

③ Activation function. "relu", Math. Formulation



Piecewise Linear.

...(1)

Note: Eqn(1) "RELU" is one of the very Popular Activation Functions.

④ Activation Function Softmax.

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

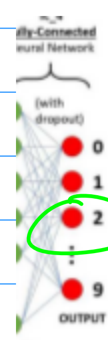
Where $\sum_{j=1}^{K}$ ← No. of Classes/Cats in the Context of Handwritten Digits Recognition: K=0 (0,1,2,...,9)

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

we have

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

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



Want to Produce Output whose value can be formulated/Interpretated as Probability Value.

$$0 \le Prob(x) \le 1 \quad \ldots (3a)$$

$$\int Pro(x)\,dx = 1 \quad \ldots (3b)$$

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

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

$$0 \le \frac{e^{z_i}}{e^{z_1} + \cdots + e^{z_{10}}} \le 1$$

(match up to Eqn 3a)

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

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

$$= \frac{e^{z_1} + \cdots + e^{z_{10}}}{e^{z_1} + \cdots + e^{z_{10}}} = 1,$$ 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: → 105b Saves the Trained Result (e.g. Both Architectural Information, all Weights/Biases)

↓

105C. Deployment.

105a: BaseLine Code For Training

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

Note: Save your Result in the following Naming Convention:

"FirstName_LastName_SID(4 Digits)_P1.h5"

CmpE258
Oct.6,22

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 minimzes 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 Interests) To Crop the Image → Further Processing

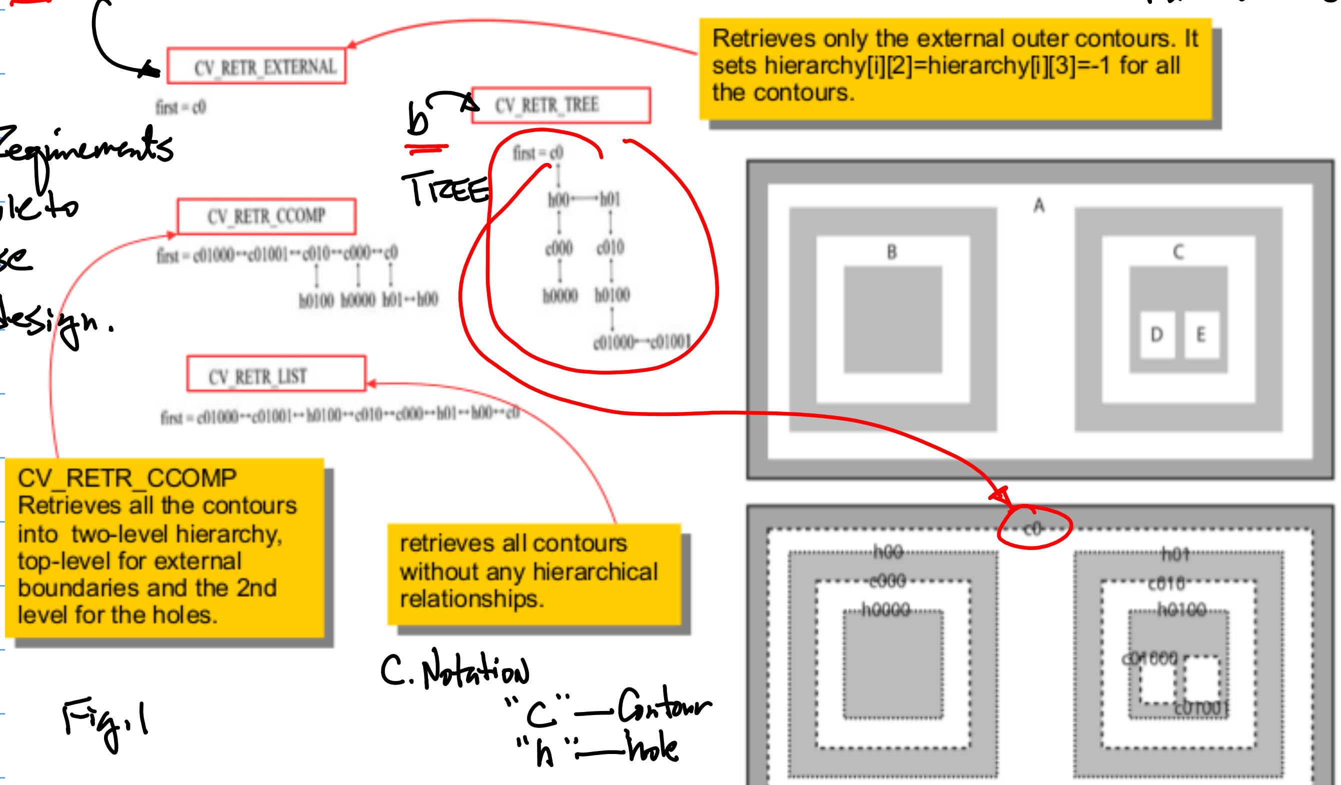
OpenCV Implementation for Contour Analysis.

4 Contours Models

a. EXTERNAL

b. TREE

Note: Requirements to Be Able to Apply these in your design.



C. Notation
"c" — Contour
"h" — hole

Fig.1

CmpE258
Oct. 11, 22

Oct 11, 22.

Note: 1° White paper 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.

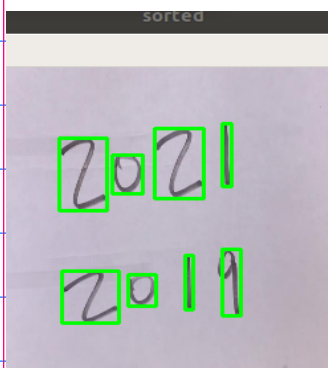


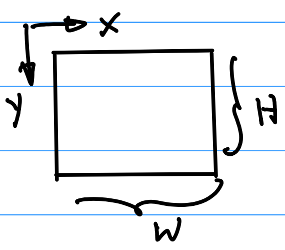
Fig. 1. We will Need Contour Analysis Technique.

From the Sample Code for Contour Analysis.

1. Line 24. Resize function Can Change the Aspect Ratio of An Image.

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

Fig. 2



where Aspect Ratio $= W/H$, $W \neq H$ ... (1)

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

Analysis "EXTERNAL" Contour Definition and its OpenCV Implementation.

Start from A Black Background Image Fig. 1. PP.31

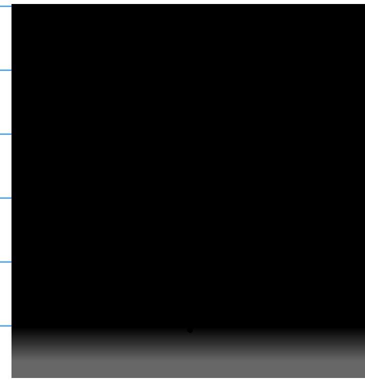


Fig. 3a

Add An Object (White)

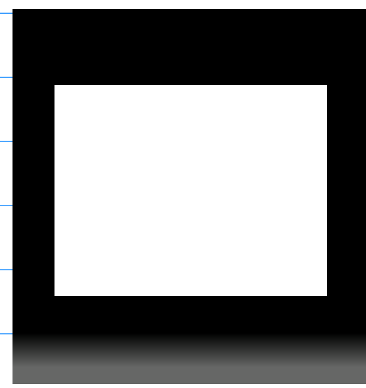


Fig. 3b

White Obj has 2 "Black" Holes

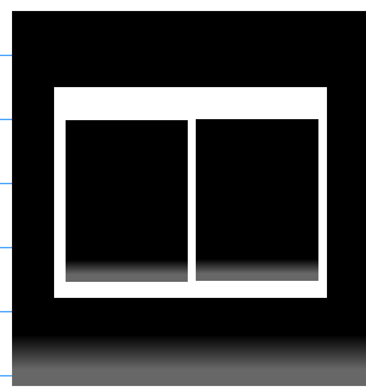


Fig. 3c

Add 2 Foreground White Squares.

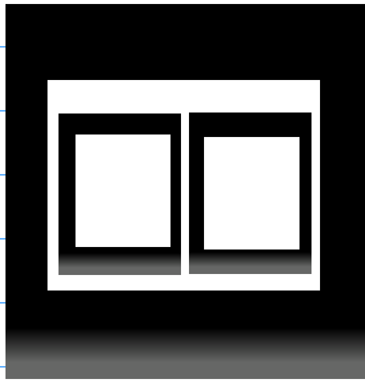


Fig. 3d

Add 2 Black Squares As Background

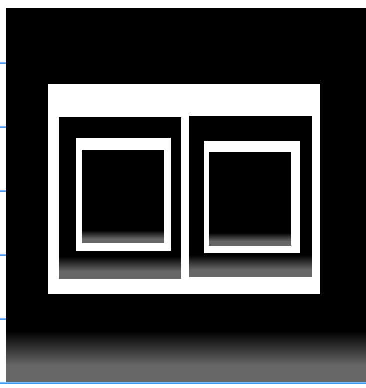


Fig. 3e

Add 2 Smaller WHT Squares to the Right Black Square

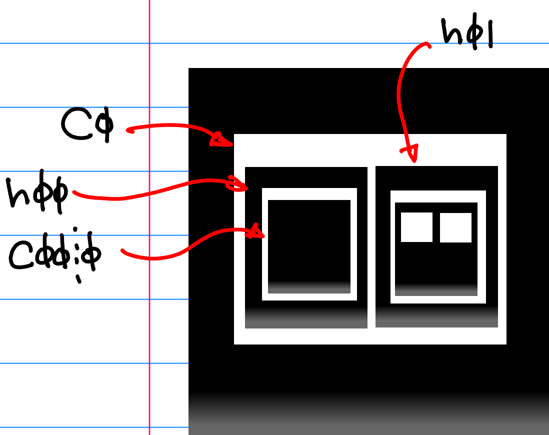


Fig. 3F

Cϕ
|
hϕϕ — hϕ1
| |
Cϕϕ:ϕ Cϕ1:ϕ
| |
hϕϕϕ:ϕ hϕ1ϕ:ϕ
|
Cϕ1ϕϕ:ϕ — Cϕ1ϕϕ:1

Note:

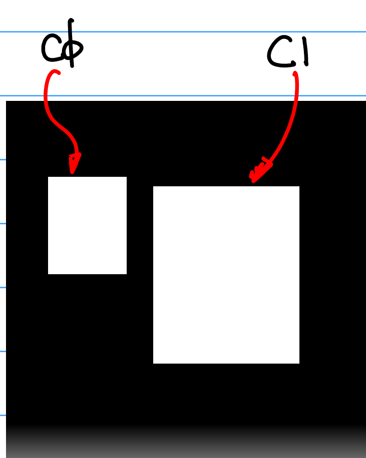


Fig. 3g

CmpE258

Oct.11,22

External Contour

```
thresh = cv2.Canny(imgray, 100, 200)
cv2.imshow('Canny', thresh)

contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
# Draw contour
thresh1 = cv2.cvtColor(thresh, cv2.COLOR_GRAY2BGR)
cv2.drawContours(thresh1, contours, -1, (0, 255, 0), 5) # all contours
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

```
import cv2
import numpy as np

img = cv2.imread('star.jpg',0)
ret,thresh = cv2.threshold(img,127,255,0)
contours,hierarchy = cv2.findContours(thresh, 1, 2)

cnt = contours[0]
M = cv2.moments(cnt)
print M
```

: To find moments of Any Contours, i or j

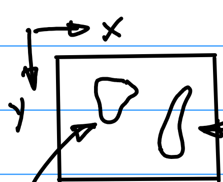
2. Contour Area

```
area = cv2.contourArea(cnt)
```

Set Threshold T=10 (pixels)

if area[i] < T

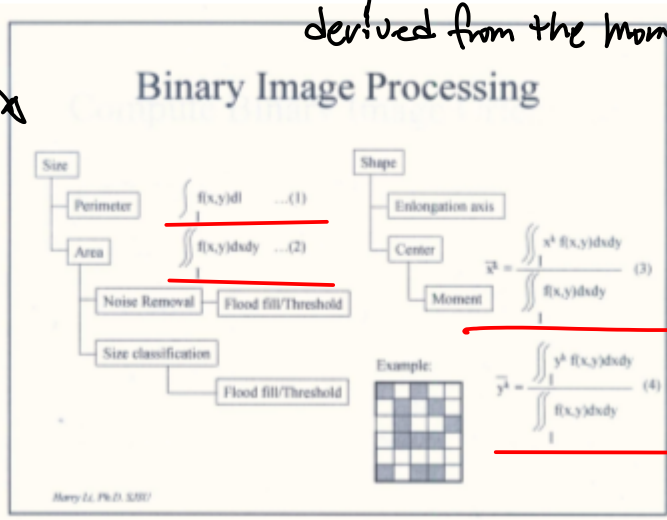
then area[i] is Random Noise



Oct.13 (Th)

Example: About Resize the Image And/OR ROI Without Distortion of Aspect Ratio.

Many Features Can be derived from the moments.



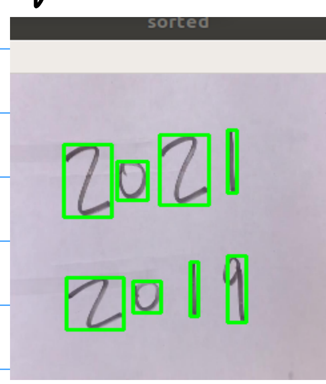


Fig.1a

Step1. Make the Bounding Box

Design ROI as A Square Image without Distortion of A.R

Use OpenCV find Contours function
↓
Bounding Box → w, h (Width, Height)
↓
$\max\{w, h\} = \hbar$

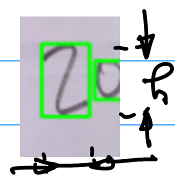


Fig. 1b

Use h Dimension to Build Square Image.

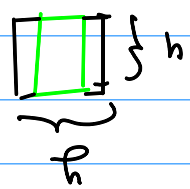


Fig. 1c

bitwiseAND

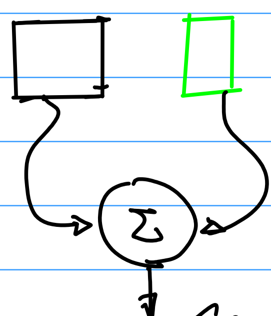


Fig 1d

Square Image.

Ref. pp. 36.

2022S-101-note-part2-cmpe258-2022-04-26.pdf

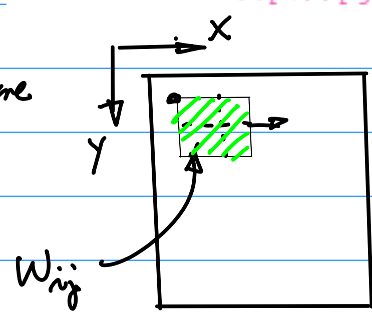
(For Yolo As well)

Background:

Existing (Previously) Techniques. ① Sliding Windows ② Region Proposal Technique

You Only Look Once:
Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/



Object Detection Technique is applied on the W_{ij}.

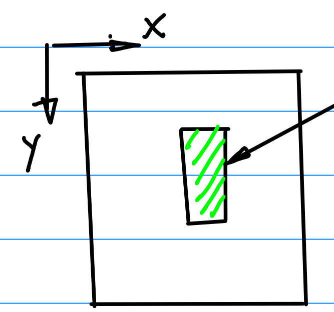
$\phi(W_{ij})$

localized, And No "global Image Information".

$I(x, y)$

Fig. 2a

Region Proposal Technique Built for R-CNN.



B_{ij}, $\theta(B_{ij})$ Object Detection & Recognition

Need | Lack of Global Image Information

Post Processing is Need.

Fig. 2b.

Now, YoLo for Object Detection & Recognition

Ref: 1. YOLO Paper

opencv / deep-learning-2022s / **2022S-112-yolo-paper.pdf**

2. Readme

deep-learning-2022s / **2022S-113-README-yolo4-v2-yy-hl-2021-4-5.txt**

```
10  Refernece URL
11  https://github.com/theAIGuysCode/tensorflow-yolov4-tflite
```

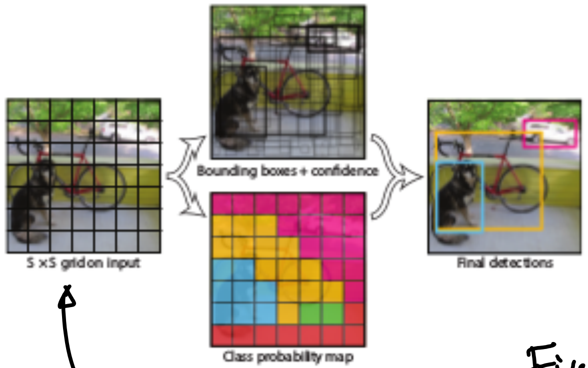


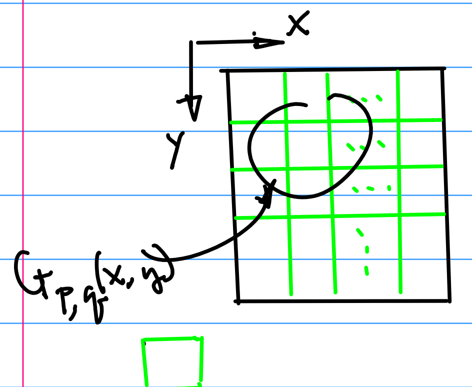
Fig.2C

Note:

1. Image $I(x,y)$ is divided into $S \times S$ Grids.

   Denote it as $G_{p,q}(x,y)$, ...(1)

   where $p,q = 0,1,2,\dots$ indicate the location of the grid. (see Ref. Lecture Note, pp40)



2. Bounding Boxes on Each Grid.

   $B_{i,j}(x,y)$ ...(2a)

   $i=x,\ j=y$

   x,y Coordinate of the top left Corner of the Box

3. For Each Bounding Box, we define 5 parameters.

   x, y (location, Top left corner)

   w, h : width and height of the Box.

   f : Confidence of the Box for its belonging to a certain Object

We can write

$$(x,y,w,h,f) \rightarrow B(x,y,w,h,f) \quad \dots (2b)$$

Note: for Each Grid Cell,

1. It predicts Bounding Boxes and their Confidence Score
2. It Computes Conditional probability

   $\text{Prob}(\text{Class}_i|\text{Object}) \quad \dots (3)$

   The probability of a given object which belongs to Class i, for $i=1,2,\dots,M$;

4. The Entire Image $I(x,y)$ is employed to generate Class Probability Distribution Map (Fig.2C), to be used in Eqn Below

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}} = \Pr(\text{Class}_i) * \text{IOU}^{\text{truth}}_{\text{pred}} \quad (1)$$

5. IOU (Intersection of Union). pp40 Ref

   Example, Illustration of IOU

Fig.3

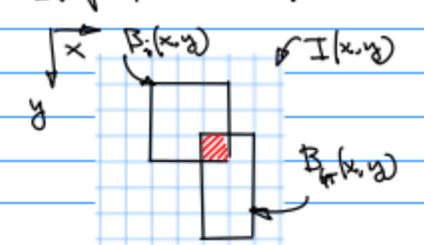


Fig.5

Intersection of Union

$$\text{IOU} = \frac{\text{Intersection}}{\text{Union}} \quad \dots (7)$$

Where Bounding Box : $B_i(x,y)$ is a Prediction.
Bounding Box : $B_{GT}(x,y)$ is the Ground Truth

Red Region: Prediction matches the ground truth; → Intersection

$$B_i(x,y) \cap B_{GT}(x,y) \quad \ldots (4)\text{-a}$$

Entire Area of $B_i(x,y)$ and $B_{GT}(x,y)$

$$B_i(x,y) \cup B_{GT}(x,y) \quad \ldots (4\text{-b})$$

Oct.18th (Tue)

Continued from the Discussion of Notation.
From Fig.2C, pp36.

Bounding Boxes + Confidence.

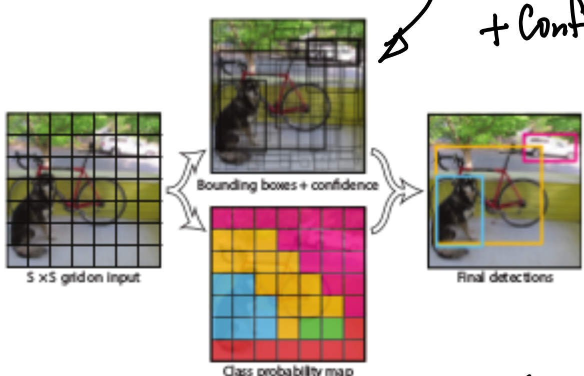
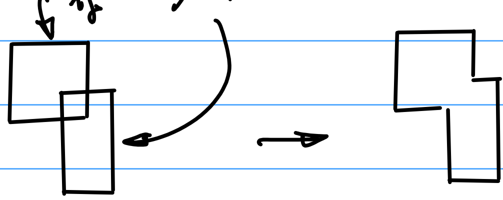
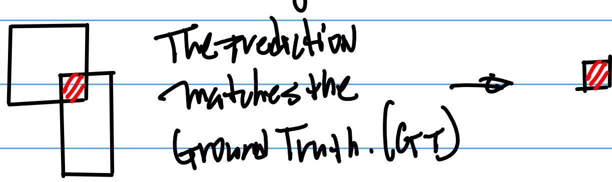


Fig.1

The Union of $B_{ij}(x,y)$, $B_{GT}(x,y)$

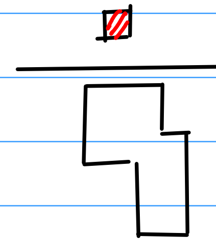


The Intersection of $B_{ij}(x,y)$, $B_{GT}(x,y)$



The prediction matches the Ground Truth. (GT)

Confidence of the Prediction By Bounding Box $B_{ij}(x,y)$ is defined as a ratio :



C = (IOU) …(1)

Confidence, denoted as η Sometimes.

Example : Given $B_{ij}(x,y)$ and $B_{GT}(x,y)$ in Fig 5, pp.36. Calculate Confidence "C" (η)

Sol
First, Find the Union of $B_{ij}(x,y)$ and $B_{GT}(x,y)$.

$$B_{ij}(x,y) \cup B_{GT}(x,y) =$$

$$N[B_{ij}(x,y)] + N[B_{GT}(x,y)] - N[B_{ij}(x,y) \cap B_{GT}(x,y)]$$

$$= 3\times3 + 4\times2 - 1\times1 = 9+8-1 = 16$$

Next,

$$N[B_{ij}(x,y) \cap B_{GT}(x,y)] = 1$$

Hence

$$C = 1/16$$

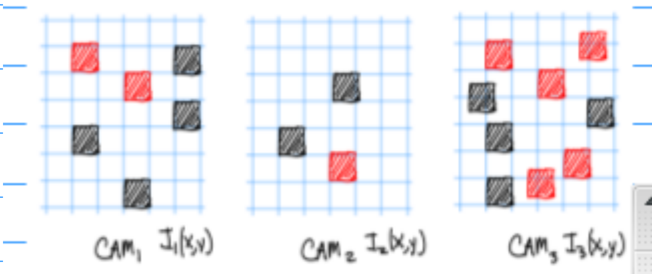
Note: $0 \le C \le 1$, similar to the probability.

Consider ~~Class~~ The Class probability map. (Same size as the Image $I(x,y)$)

2022S-101-note-part2-cmpe258-2022-05-3.pdf

Background on Baysian Theorem.

Example, pp.39



Camera1. $I_1(x,y)$, Camera2, $I_2(x,y)$, ...

Red Squares: Pedestraint; Black Squares: Vehicles

1. Event of having pedestraint Appeared in the Images

$$R = RI_1 + RI_2 + RI_3 \quad \cdots (1^*)$$

2. Probability of the event R.

$$Prob(R) = Prob(RI_1) + Prob(RI_2) + Prob(RI_3) \quad \cdots (2)$$

Where $I_1 \cap I_2 \cap I_3 = \phi$ (Empty set)

3. Consider

$$Prob(RI_1) = Prob(R/I_1)\,Prob(I_1)$$

Pedestraint(s) in $I_1(x,y)$ ... (3a)

$$Prob(RI_2) = Prob(R/I_2)\,Prob(I_2) \quad \cdots (3b)$$

$$Prob(RI_3) = Prob(R/I_3)\,Prob(I_3) \quad \cdots (3c).$$

Eqn (2) and Eqn (3a)–(3c):

$$Prob(R) = \sum_{i=1}^{3} Prob(R/I_i)\,Prob(I_i) \quad \cdots (4)$$

Now, consider the formulation Below: (see Fig.1, pp.37)

$$\underbrace{\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object})} * \text{IOU}^{\text{truth}}_{\text{pred}} = \underbrace{\Pr(\text{Class}_i)} * \text{IOU}^{\text{truth}}_{\text{pred}} \quad (1)$$

Bounding Box Confidence.

$$Prob(Class_i) = Prob(Class_i | Object)\,Prob(Object) \quad \cdots (5)$$

Note: The Discussion and Mathematic Formulation, Equations, Formula are required.

Implementations: { Code (github), Architecture, Loss function

Homework: Due Oct 27th (Th).

Installation & Test Run Yolo4.

1. Down Load Yolo4 (Readme from the github). Run the code.

CmpE258
Oct. 18, 22

2022F-106-YOLO4-README-Tiny-Yolo4-G...

2. Use Your Smart phone to Record ~15 sec. Video Clips for the testing of Yolo4. (Such as Traffic, Vehicles, Pedestraints).

3. Submission:

① Screen Capture of the Success execution of the code; (Identifier, Such as folder name etc.).

② Video Clips Saved After the processing

Example: Architecture of Yolo, Ref:

2022F-108a-Yolo-architecture-loss-function-2022-10-10.pdf

Architecture Discussion. Fig. 1

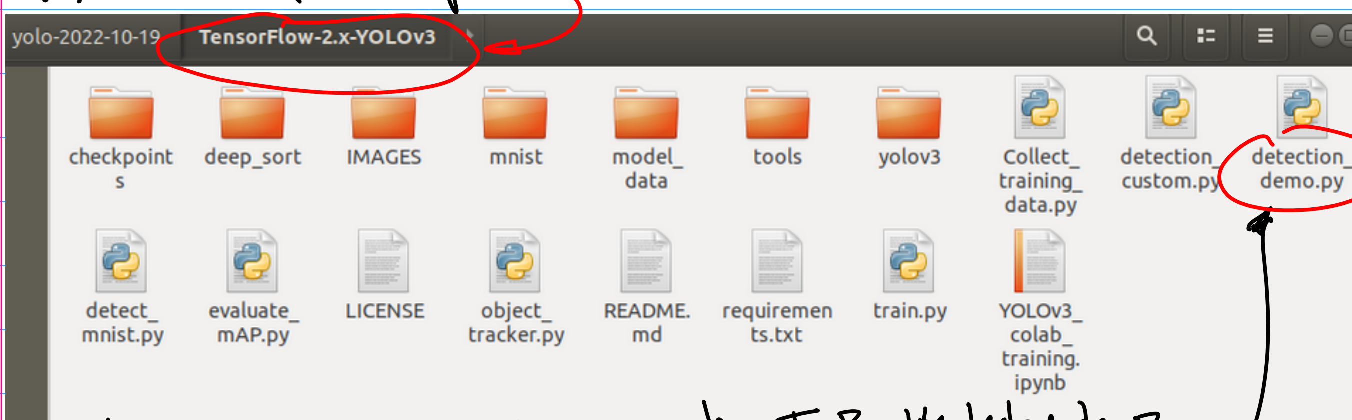
Note: Midterm Exam Schedule Nov. 3rd. Review Session will be provided;

Oct. 20.

Note: Homework Using Yolo. Updated Readme (github link was updated).

2022F-106-README-Tiny-Yolo4-GPU-Ubu...

a. Down Load from the github



marked as V3, But follow the readme Document, modify the Configuration file (.py) to Choose Yolo v4;

b. To Run the test code, Run Python Code

c. Create "yml" file for Conda Environment Setup.

CmpE 258

Oct. 20, 22

40.

a. 448x448x3

Channel

b. Convolution with Stride = 2. "Scan" the input Layer at every two pixels from Left to Right, then at every other row from Top to Bottom.

Base Line Yolo Architecture

Hand Calculation Is Required.

1. Input image size: 448x448x3; 2. Resolution reduction for feature extraction/abstraction Pooling and convolution with stride = 2;

Design guideline: The block of convolutional layers to extract image features, the fully connected layers to predict the output probabilities and the locations (coordinates).

convolutional layers | Fully connected layers

c. 1x1 Convolution.

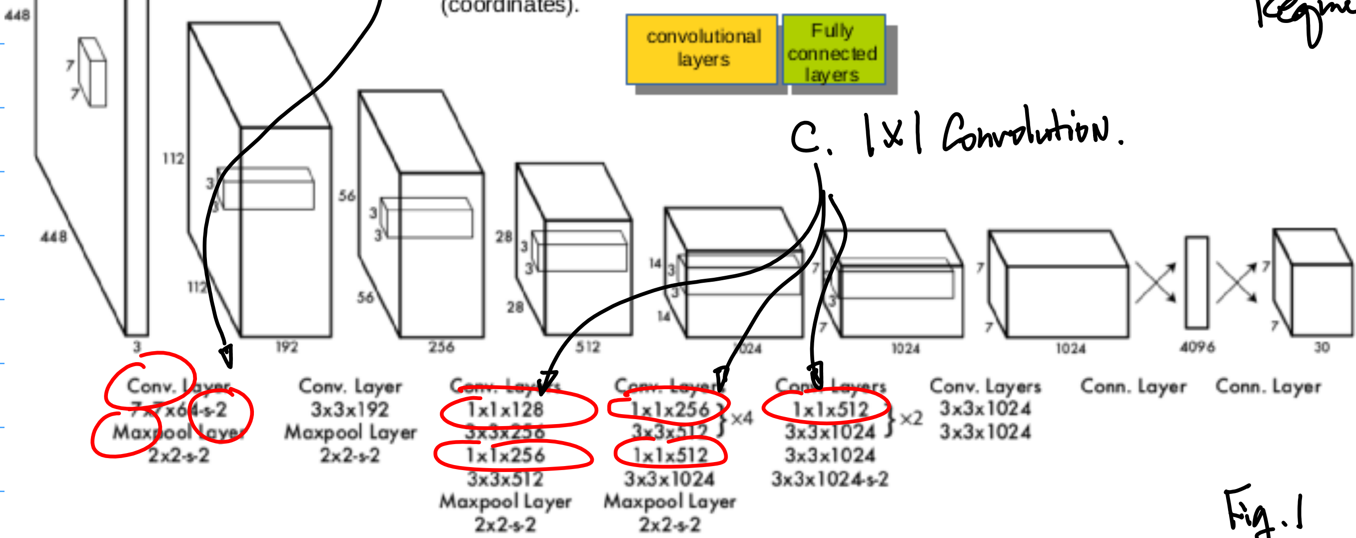


Fig. 1

The Motivation for the 2-Stride, or K-stride convolution is to combine Convolution with Subsampling, e.g. Reduction of the feature layer. Example Below, See Stride pattern in Green.

Consider 3x3 Kernel convolution

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

Fig. 3a

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

3x3x1 Channel

Image

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

Fig. 2

Reduce 3x3 kernel to 1x1 Kernel below, then Convole with the image $I(x,y) \longrightarrow$ Output Resolution is MxN

MxN

$\boxed{25}$

X-Y Two Dimensional Space "Spatial"

V.S. Temporal (Time)

CmpE258

Oct. 20, 22

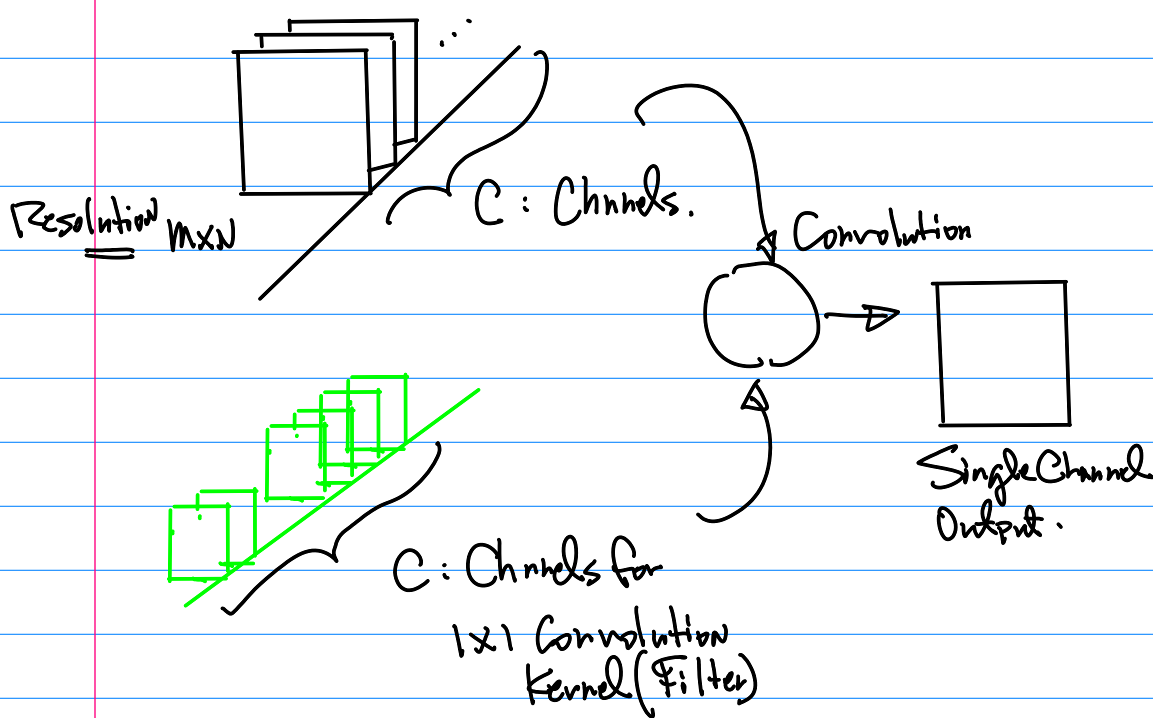
41

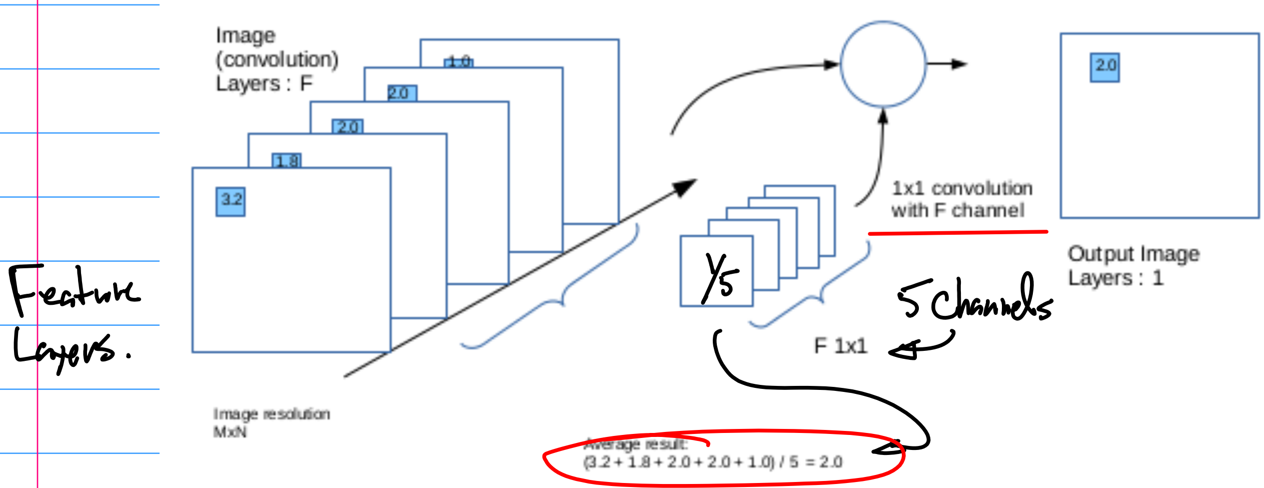
Space (Spatial) in 2D → Space in 3D and Beyond.

for Example for Image

Note: 1° 1x1 Convolution Reduces the Number of Layers of the Input.

2° Pooling.





L.i, Ph.D.

CmpE258

Oct. 20, 22

Example: a) A loss function can be defined by subtracting the Output (function) from the ground truth. → Square it to prevent from possible Cancellations when Summed up together

Loss Function for YOLO

Location Based Loss function. c)

Location $(\hat{x}_i, \hat{y}_i)$ of Bounding Box

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2-1} \sum_{j=0}^{B-1} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\textbf{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

$\mathbb{1}_{ij}$: i for Each Grid S^2; j for Bounding Boxes $B+1$

$$\mathbb{1}_{ij} = \begin{cases} 1 & \text{when } B_{ij} \text{ exists} \\ 0 & \text{o/w (otherwise)} \end{cases}$$

b) One Loss function

$$f_{Loss\,\Sigma} = \sum_{i \in I} f_{Loss,i} \quad \cdots (1)$$

$$f_{Loss\,\Sigma} = \alpha_1 f_{Loss,1} + \alpha_2 f_{Loss\,2} + \cdots + \alpha_K f_{Loss\,K}$$

where $\alpha_1 + \alpha_2 + \cdots + \alpha_K = 1$

Note: If we want emphasize Location Prediction, → f_{Loss1} → $\alpha_1 \uparrow$ Bigger α_1

c. From Eqn Below.

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}} \quad (1)$$

Lower Branch Upper Branch

Fig1.a

d) Geometric Shape (Always in Rectangle Shape, Size) Loss Function

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right]$$

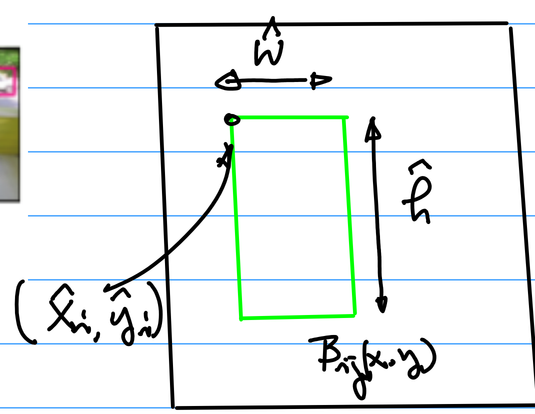


Fig1b

$G_{ij}(x,y)$

CmpE258

Oct.25,22

Note: Midterm Exam Scheduled on the 3rd, Nov.

$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{obj}}\left(C_i-\hat{C}_i\right)^2$$

$$+\lambda_{\text{noobj}}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{noobj}}\left(C_i-\hat{C}_i\right)^2$$

Loss function Based on the Confidence of Bounding Box

Loss function of the class probability.

Ground Truth

Estimated Probability

$$+\sum_{i=0}^{S^2}\mathbb{1}_{i}^{\text{obj}}\sum_{c\in\text{classes}}\left(p_i(c)-\hat{p}_i(c)\right)^2$$

Prob(c)

Class, example, Traffic Signs, Pedestrian, Vehicles etc.

Note: To be able to Interprete & Apply this Technique to Yolo and other Related Design Need.

Example: Ref:

2022S-114c-Kmean-handCalculat…

2022S-114c-KmeanCluster-v3-20…

Background: A Tool to Create Probability Distribution Map. (In Fig 1a, pp 42)

2022S-114c-KmeanCluster-v3-2022-4-19.pdf

$$\underset{\mathbf{S}}{\arg\min}\sum_{i=1}^{k}\sum_{\mathbf{x}\in S_i}\left\|\mathbf{x}-\boldsymbol{\mu}_i\right\|^2=\underset{\mathbf{S}}{\arg\min}\sum_{i=1}^{k}|S_i|\operatorname{Var}S_i \quad \ldots(1)$$

Feature vector $\vec{X}=(x_1, x_2, \cdots x_N)$ with Dimension N. → X

for Example N=2.

a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$),

Experiment 1

$X_1=(x_{11}, x_{12})$

the 1st Sub: For Experiment 1.

So, for k-th Experiment, we have $X_k=(x_{k1}, x_{k2})$. etc.

$\mu_i=(\mu_{i1}, \mu_{i2}, \cdots, \mu_{iN})$, mean of the given Class i

A Vector, Same dimension as the feature Vectors

for N=2, $\mu_i=(\mu_{i1}, \mu_{i2})$

i index for Class i.

CmpE258

Oct. 25, 22

44

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \underbrace{\|\mathbf{x} - \boldsymbol{\mu}_i\|^2}_{} = \underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i \quad \cdots (1)$$

Comparison, for the purpose of Classifying / Grouping a feature Vector to Class i.

Mean (AVG)

μ_i : feature Vector Distribution

then, $\sum_{x \in S_i}$ ← Take all feature Vectors X, as Long as they are from the Class S_i, Class i

Example: $X_j = (x_{j1}, x_{j2})$

$\mu_i = (\mu_{i1}, \mu_{i2})$

$X_i = (x_{i1}, x_{i2})$

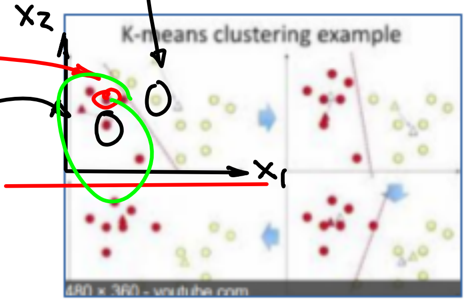


Fig. 2

Cluster Seeking

Graphically μ_i a Clusster

features: Edge Comp. from Edge Detection; Colour Distribution; Contours

Then, To make sure the minimization Argmin$_S$ to cover all classes :

$\sum_{i=1}^{K}$ ← Total K Classes.

$X - \mu_i$

↓

Minimize the Classification Error (Distance) By Grouping Feature Vector to its Right (Optimal) Cluster (Class)

"argmin" Or "Min" Minimization

↓

$\underset{S}{\text{argmin}}$ ← for All the possible Classes

Oct. 27.

Midterm Exam is scheduled on Next Thursday (Nov. 3rd). Brief Review Session will be conducted on next Lecture.

Example: K-mean Calculation

2022S-114c-KmeanCluster-v3-2022-4-19.pdf

b. t Iteration

$$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 \le \|x_p - m_j^{(t)}\|^2 \ \forall j, 1 \le j \le k\} \quad \ldots (2)$$

a. $\{S_i \mid i = 1, 2, \cdots, K\}$

Classes / Group.

if $\|\vec{X}_p - \vec{m}_i^{\,t}\|^2 < \|\vec{X}_p - \vec{m}_j^{\,t}\|^2$

$\vec{X}_p$ is better fitted into the Group i, O/W, $\vec{X}_p$ is better fitted to Group j

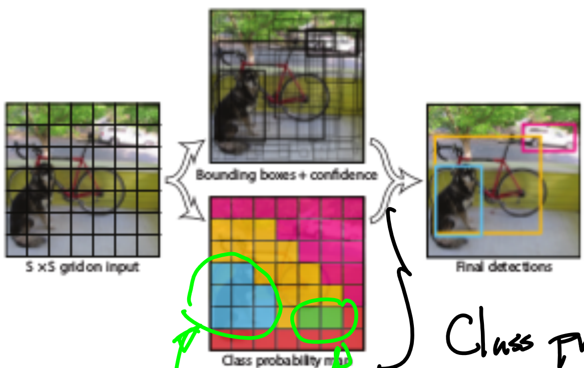
&

Condition(s) in Eqn (2).

c. $\vec{X}$ feature Vector

$\vec{m}_i^{\,t}$ ← Iteration, Updated @ Step t

mean from Class i



Class probability map.

Probability of Class i

Probability of Class j

$\forall j$, such that $1 \le j \le K$

For Any j j is in this Range, And it Covers all the groups. ($\because \sum_{i=1}^{K}$ from Eqn (1)).

$$\vec{m}_i^{\,t} = \begin{pmatrix} m_{i1}^{t} \\ m_{i2}^{t} \end{pmatrix} \quad \ldots (3)$$

Take Eqn (2) to Actual Computation.

d. $\|\cdot\|^2$ Euclidean Distance

$$\|\vec{X}_p - \vec{m}_i^{\,t}\|^2$$

$$= \left\| \begin{pmatrix} x_{p1} \\ x_{p2} \end{pmatrix} - \begin{pmatrix} m_{i1}^{t} \\ m_{i2}^{t} \end{pmatrix} \right\|^2 = \left\| \begin{matrix} x_{p1} - m_{i1}^{t} \\ x_{p2} - m_{i2}^{t} \end{matrix} \right\|^2$$

$$= \sqrt{(x_{p1} - m_{i1}^{t})^2 + (x_{p2} - m_{i2}^{t})^2}$$

Update step: Calculate the new means to be the centroids

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \quad \ldots (3)$$

Algor
1).

Vector for Group i, mean, e.g. Average;

"$t+1$" indicates it is updated from the previous Stage/Step "t"

$$\frac{\vec{X}_1 + \vec{X}_2 + \cdots + \vec{X}_Q}{Q}$$

Q Total Number of the feature Vectors from ith Group

$|S_i^t|$ Total Number

Compare 2 Groupings, e.g. 2 Ways of Classify feature Vector $\vec{X}_p$ to Group i and Group j

Example: Hand Calculation

2022S-114c-Kmean-handCalculation1-converted.pdf.pdf

$X_1=\begin{bmatrix}0\\0\end{bmatrix}$ $X_2=\begin{bmatrix}1\\0\end{bmatrix}$ $X_3=\begin{bmatrix}0\\1\end{bmatrix}$ $X_4=\begin{bmatrix}1\\1\end{bmatrix}$

$X_5=\begin{bmatrix}2\\1\end{bmatrix}$ $X_6=\begin{bmatrix}1\\2\end{bmatrix}$ $X_7=\begin{bmatrix}2\\2\end{bmatrix}$ $X_8=\begin{bmatrix}3\\2\end{bmatrix}$

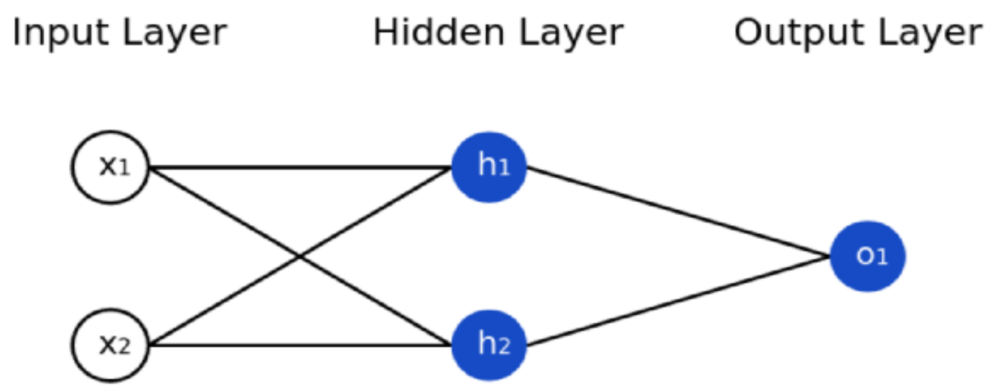
$X_9=\begin{bmatrix}6\\6\end{bmatrix}$ $X_{10}=\begin{bmatrix}7\\6\end{bmatrix}$ $X_{11}=\begin{bmatrix}8\\6\end{bmatrix}$ $X_{12}=\begin{bmatrix}6\\7\end{bmatrix}$

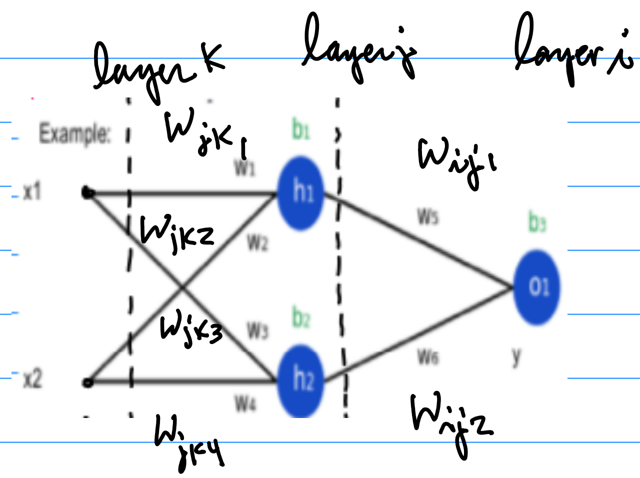
$X_{13}=\begin{bmatrix}7\\7\end{bmatrix}$ $X_{14}=\begin{bmatrix}8\\7\end{bmatrix}$ $X_{15}=\begin{bmatrix}9\\7\end{bmatrix}$ $X_{16}=\begin{bmatrix}7\\8\end{bmatrix}$

$X_{17}=\begin{bmatrix}8\\8\end{bmatrix}$ $X_{18}=\begin{bmatrix}9\\8\end{bmatrix}$ $X_{19}=\begin{bmatrix}8\\9\end{bmatrix}$ $X_{20}=\begin{bmatrix}9\\9\end{bmatrix}$

Feed Forward NN

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





Range them from Output Layer to the input layer.

$W_{ij} \to W_{jk}$, Scalable

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

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \cdots \\ \frac{\partial f}{\partial x_i} \\ \cdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

$$(w^{k+1}_{ij1}, w^{k+1}_{ij2}) = (w^{k}_{ij1}, w^{k}_{ij2}) + (-\eta(\nabla f)^t)$$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial w_{ij1}} \\ \frac{\partial f}{\partial w_{ij2}} \end{pmatrix}$$

Where $f = \frac{1}{2}\sum_{j=0}^{M-1}\sum_{i=0}^{N-1}\left(y_i^j - \hat{y}_i^j\right)^2$

loss function

And y_i^j is the Output

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

y_i^j ith output @ the last layer.

@ Experiment j

Drop j. Rewrite Subscript.

$$y_{ij1} = f\left(\sum_k w_{ijk} x_{ijk} + b_{ij}\right)$$

$$= f\left(h\left(w_{ijk}, x_{ijk}, b_{ij}\right)\right)$$

$$\frac{\partial}{\partial w_{ijk}} L = \frac{\partial}{\partial w_{ijk}}\left(\frac{1}{2}\sum_M \sum_K \left(y^M_{ijk} - \hat{y}^M_{ijk}\right)^2\right)$$

$$\rightarrow \underbrace{2 \cdot \frac{1}{2} \cdot \left(\sum_M \sum_K \left(y^M_{ijk} - \hat{y}^M_{ijk}\right)\right)}_{S} \frac{\partial}{\partial w_{ijk}}\left(-y^M_{ijk}\right) = -S \frac{\partial}{\partial w_{ijk}} f\left(\sum_K w_{ijk} x_{ijk} + b_{ij}\right)$$

$$= -S f'(\cdot) \frac{\partial}{\partial w_{ijk}}\left(\sum_K w_{ijk} x_{ijk} + b_{ij}\right)$$

$$= -S f'(\cdot) \frac{\partial}{\partial w_{ijk}} \sum_K w_{ijk} \quad = -S f'.$$