Jan 28, 2021

Welcome To CMPE258 !

First Day of the Class

Harry LI, github/hnaliki/opencv/deep-Learning-20205

20-2015 Email: hnali@sjsu.edu

Office Hours M.W. 4:30-5:30 PM.

Zoom Based

(650) 400-1116 Text Only

On-Line Material { github/hnaliki
CANVAS { Homework Assignment
Collect/Submission of Homework

HANDS-ON:

Programming: Python

(1) OpenCV, T.F. (Tensor Flow) Keras API

google Colab

Action 1: Installation of OpenCV. Version 4.2

Note: Use/Adopt Linux Ubantu 18.04

Virtual Box

2nd O.S.

Note: Python 3. Python Virtual Enviroment

3 major Areas { (1) Handwritten Nemals Recognition. MNIST
(2) Time Series Prediction LSTM
(3) FaceNet, ResNet
(4) Deep Reinforcement Learning DRL

Subjects

C.V. R.O.I. Deep NN

Action-Plan

Work to be done:

1. Programming / Code Development from Repo.
2. Write/Submit Pseudo Code (Brief Summary) Report 1 page
Note: Post a Sample on github Latex
3. Homework Submission & Submission (Including Semester Long team Project).

Action 2: Form 4-person Team By Feb 14 week; work has to Individual/Encourage team Discussion.

Grading Policy: { mid: 30%
Homework: 30%
Final: 40%

* Introduction

Nemal Nets

Biological System

Human Brain

Neurons (Cells)

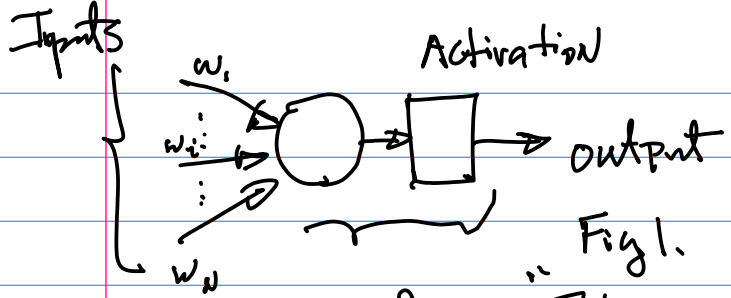


Fig 1.

Prof. Carvard Mead — "Silicon Brain"
Intel Processor ~1992-94, ↓ Father of VLSI"
~1994-95
↓
Autonomous System
↓
2005-2006
Standford Group + Google
Self Driving Market Hype
↓
2013 Alex Net (Deep convolutional Neural Network)
NVDA GPU Architecture

Recently: FaceNet, ResNet
Time Series Analysis (LSTM)
Deep Reinforcement

OpenCV
1997

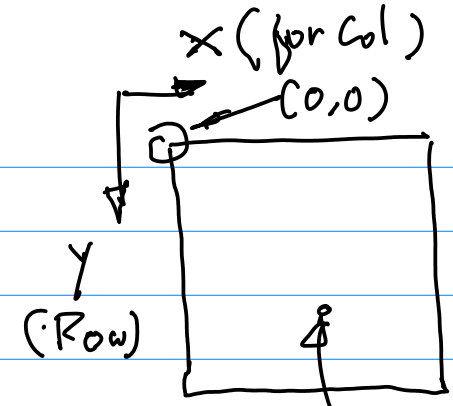
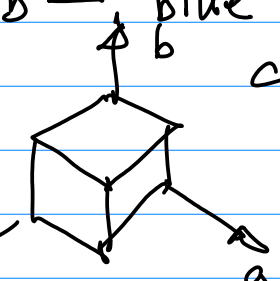


Fig.3

Note:
1° "Scanning" ~ From L to R, top to B
"L2R, T2B"
2° Resolution 1024x768
M x N
Resolution
No. of column
No. of Row
M x N → 1024x768
M N
Colum Row
X y

3° Color Image Vector
Color Space (r, g, b)
Primitive Color
r — red;
g — green;
b — blue



color Cube

Fig.4

(0,0,0) Black (1,1,1) White
grey scale as Traveling from (0,0,0) towards (1,1,1)

Computer Vision

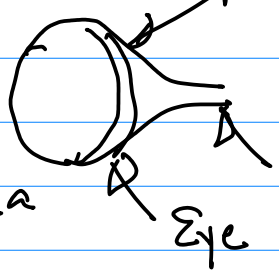


Fig.2a

multiple Layers of Photo Receptors
~10 layers.
110 ~ 120 million P.R.
Optic N. (Brain cells)
~ 1 million

Image Pyramid

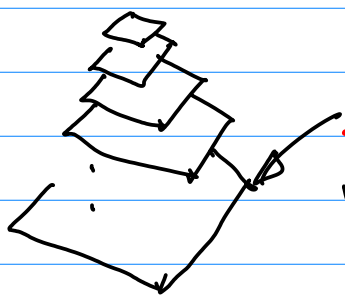


Fig.2b

Highest Red $(r,g,b) = (1,g,b)$

" Green $(r,g,b) = (r,1,b)$

" Blue $(r,g,b) = (r,g,1)$

$I(x,y)$ → Intensity in terms of (r,g,b)

① Location On the Image Plane

$I(100, 201)$

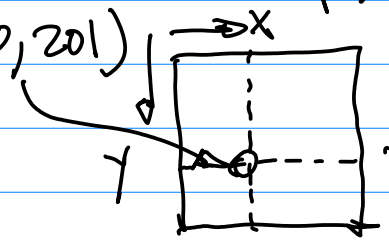


Fig 5

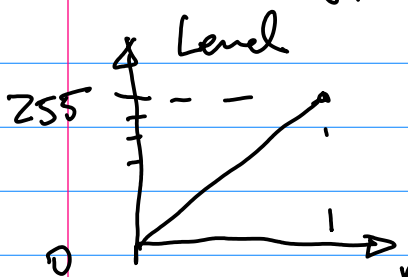
r: 8 bit [0, 255] $(2^8 = 256)$

g, b; 8 bit, " "

Quantization Level

Pixel Depth (BPP: Bit Per Pixel)

For 8 bit Grayscale Image



$$I(x,y)_g = \frac{1}{3}[r(x,y) + g(x,y) + b(x,y)] \quad \cdots (1)$$

r,g,b. Convert Color Image to Grayscale Image

Note: 1° Installation of Python, OpenCV, and T.F.

Python { 2.7
3.5 or higher

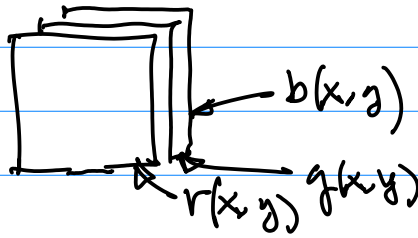
Anaconda — Python Distribution → Python Virtual Environment

OpenCV 4.2 Version 3.0 or higher

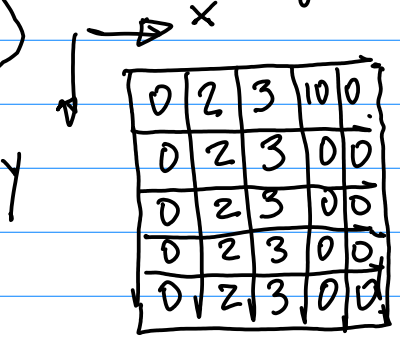
T.F. Keras (API)

Given a digital image

$I(x,y) \rightarrow r(x,y), g(x,y), b(x,y)$



Example: Suppose an $I(x,y)$ is given below,



Action 3. Enable OpenCV { colab / Locally (Install)

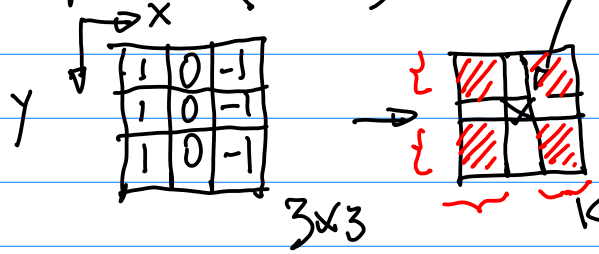
.... Display A "Jpg" Color Image → From Your Smart

Feb 4th, CMPE 258

Today's Topics: 1° Convolution, Two Dimensional Convolution; 2° Intro to Neural Network

Performan Convolution on $I(x,y)$

Convolution kernel (mask)



Ref: [_2020S_ ... 2DConvolution

From 1D Case, given a function $f(x)$ and a kernel $g(x)$

$$\int_{\Omega} f(u)\, g(x-u)\, du \quad \dots (1)$$

$$\sum_{i \in \Omega} f(i)\, g(x-i) \quad \dots (2)$$

Now, 2D Case

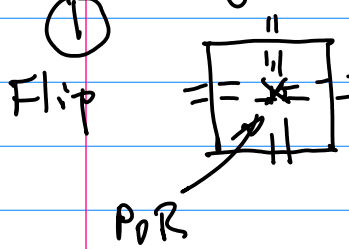
$$\iint_{\phi} f(u,v)\, g(x-u, y-v)\, du\, dv \quad \dots (3)$$

$$\sum_{j} \sum_{i} f(i,j)\, g(x-i, y-j) \quad \dots (4)$$

Image, Kernel, Index of pixel location

Kernel $g(x,y) \rightarrow g(-x,-y)$

① Flip



a) Flips the Kernel

b) $g(x,y) = g(-x,-y)$ if Symmetric

② $f(i,j)\, g(x-i, y-j)$ product

③ shift

$g(x-i, y-j)$

$g(1-i, 1-j)$

$g(2-i, 1-j)$

$g(2-i, 2-j)$

and so on ...

Summary: 2D convolution consists of shift-product-summation

Compute 2D convolution:

Step 1. Place the kernel @ the initial condition, e.g. the top left Hand corner of the image, in such a way its boundary rows and columns are aligned with the image boundary row and column.

$f(0,0)*1 + f(1,0)*0 +$
$f(2,0)*(-1) +$

$f(0,1)*1 + f(1,1)*0 +$
$f(2,1)*(-1) +$

$f(0,2)*1 + f(1,2)*0 + f(2,2)*(-1)$

$= 0*1 + 2*0 + 3*(-1) +$
$0*1 + 2*0 + 3*(-1) +$
$0*1 + 2*0 + 3*(-1)$

$= -3 - 3 - 3 = -9$

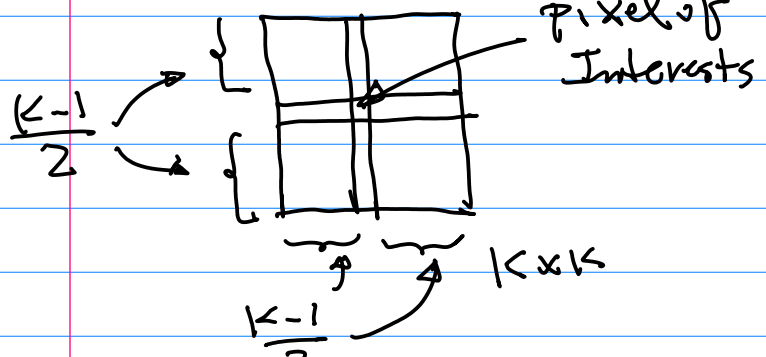
Note: This convolution resulted in a new processed image plane $I_{New}(x,y)$ whose Rows is less than the original image,

No. of Rows = Original No. of Rows − 2

for 3x3 kernel, for KxK kernel (K is odd Number), in this case

No. of Rows Reduced by $2*\left(\frac{K-1}{2}\right)$

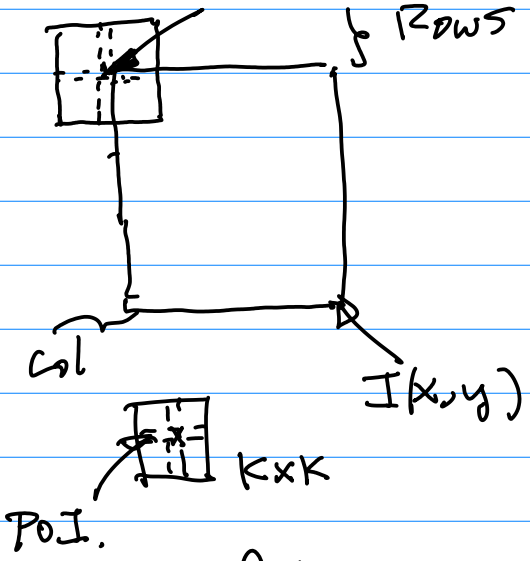
$= K-1$.



Homework (Exercise — No Submission)

Based on the given I(x,y) Image from PPT @ github, perform hand Calculation of 2D convolution

Note follow the example in Class.



Consider Neural Networks.

Supervised Learning

Reference: ZR-20215-Z

Example: $\bar{x}$

Input $(x_1, x_2, \cdots, x_n)$ $\bar{w}$

Weights $(w_1, w_2, \cdots, w_n)$

$\bar{x} \cdot \bar{w} = (x_1, x_2, \cdots, x_n) \cdot (w_1, w_2, \cdots, w_n)$

$= x_1w_1 + x_2w_2 + \cdots + x_iw_i + \cdots + x_nw_2$

$= \sum_{i=1}^{n} x_i w_i \quad \cdots (1)$

Define Transfer function $f(\cdot)$ as follows

$$f(\bar{x}) = \begin{cases} 1 & \sum_{i=1}^{n} x_i w_i + b > 0 \\ 0 & \text{O/W} \end{cases} \quad \cdots (2)$$

Feb 11,

Today's Topics: 1° Introduction to NN; 2° Coding (Python) Examples.

Action 1: Form 4-person Team. Submit your team member information By Friday 5:00 pm.

Subject: CmpE258 First Last Name of the group coordinator

First, Last Name, Last 4 Digits of your SID; E-mail Contact information;

Homework 1: Due A week from Today; Submission to CANVAS.

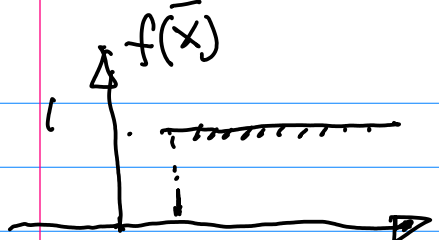
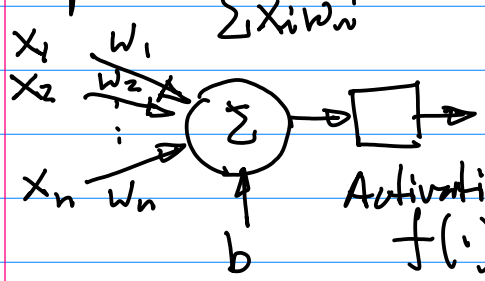


Fig 1.

Consider a Simple Feedforward NN Example:



$\sum x_i w_i$

Fig. 2

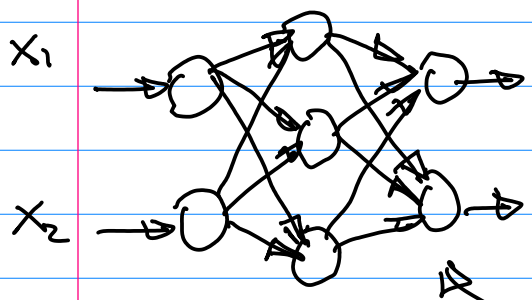


Fig. 3

Input Layer 2 Neurons; Hidden Layer 3 Neuron; Output Layer 2 Neurons

Consider Dense Feedforward NN From the PPT.

1. A Single Neuron { Block Diagram; Notation

$\bar{x} = (x_1, \cdots, x_2)$

$y = f(\bar{x}) = f(x_1, x_2, \cdots, x_n)$

Activation function

FeedForward

Have Training Dataset ϕ or Ω

Two Classes C_1, C_2 Representing 2 Patterns

Feature Vector $X = (x_1, x_2)$, up to N_1 of them for C_1 upto N_2 for C_2

N_1, N_2 do not have to be equal;

Training (Learning)

$$w_i^+ = w_i^- + \Delta w_i = w_i^- + \eta(d - y)x_i \quad \cdots (3)$$

w_i^+: updated; w_i^-: Current; η: Rate; d: desired Output; y: Actual Output

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

3. Python Implementation
Create A template for your program header,

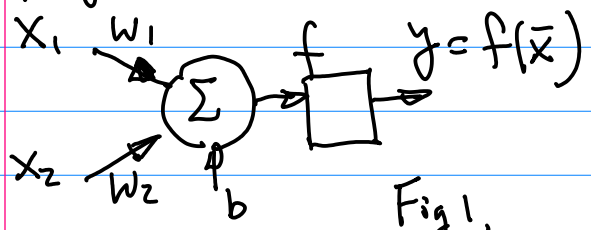


Fig 1.

$X_1 = 2, X_2 = 3, W_1 = 0, W_2 = 1.$
$b = 4$, $f(x) = \text{Sigmoid}(x)$

4. Feedforward NN

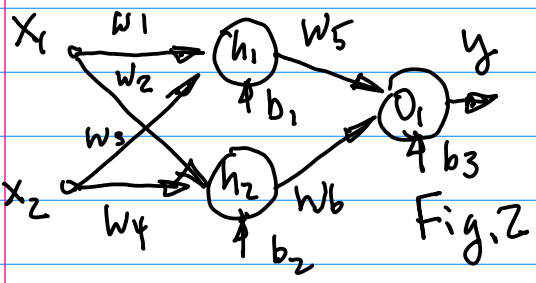
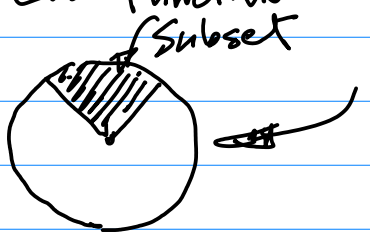


Fig. 2

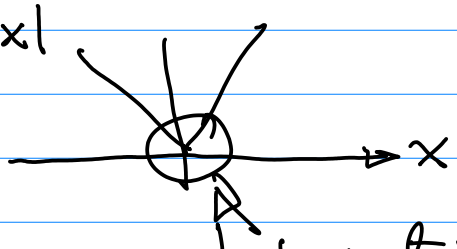
5. Data set → Pre-processing to allow Activation function to Better handle the input values

Supervised Learning
Data → Its Categories Known
Labeling Images/Videos → Annotation.
ImageNet ~1.4 million Test Images

6. Loss Function → Objective Function



y_{true}, y_{pred} — Notation & meaning
gives the information for which Category the data belongs to
→ use squared difference to handle potential error (Loss)
Cancellations due to opposite Signs
Note: Be Careful Not Absolute value!



No derivatives at this point.

8. Define Loss function
↓
Behavior of Loss Function → minize the Loss/error function in the negative direction of the gradient
Gradient gives the fastest increase of the error

9, Denote the Loss as

$$L(W,b) = \sum (y_{True} - y_{pred})^2 \quad \cdots (2)$$

$W = (w_1, w_2, \cdots, w_N)$ $\quad b = (b_1, b_2, \cdots, b_M)$

$M \neq N$

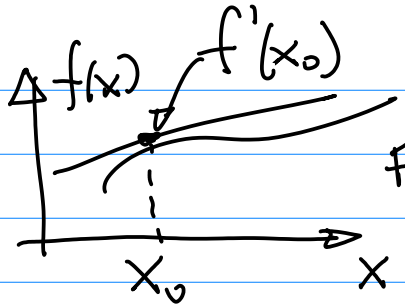


Fig. 1

$f'(x_0) > 0 \rightarrow f(x_0 + \Delta x) - f(x_0) > 0 \quad \cdots (2)$

Increase

$f'(x_0) < 0 \rightarrow f(x_0 + \Delta x) - f(x_0) < 0 \quad \cdots (2^*)$

$f(x) = \text{BasicBuildingBlock}_0 + BBB_1 + BBB_2 + BBB_3 + \cdots$ Higher order terms $\cdots (3)$

BBB_0 1st One Constant

BBB_1 2nd one Linear

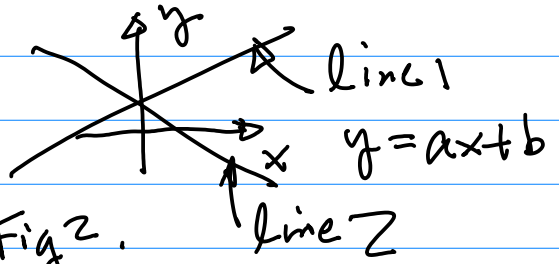


Fig 2.

BBB_2 3rd 2nd order quadratic $y = ax^2 + bx + c$

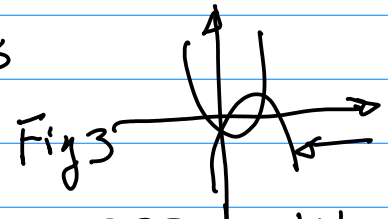


Fig 3

BBB_3 4th 3rd order Cubic $y = ax^3 + bx^2 + cx + d$

θ CMPE258 Feb18

Reference: 1° Training Algorithm (PART I) 2° OpenCV — Preprocessing

Objectives: Deep Convolutional NN, Deep Reinforcement Learning

(1) Handwritten Digits Recognition MNIST (2) Facial Recognition (3) Deep Reinforcement Learning (4) LSTM Time Series Data

Note: 1° Minimization, Optimization, Maximization.

f_{min} ? $\rightarrow 1/f_{max}$

$g_{max} \rightarrow 1/g_{min}$

2° Taylor Expansion

Basic Building Blocks to Build A Given function $f(x)$ provided it has derivatives up to order K.

$$\lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x} = f'(x) = \frac{d}{dx} f(x) \quad \cdots (1)$$

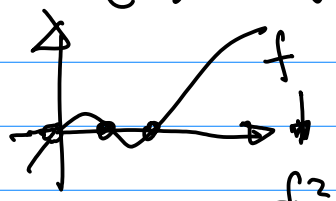
$\Delta x \rightarrow \Delta f \rightarrow \Delta f / \Delta x \rightarrow \frac{\Delta f}{\Delta x}\Big|_{\Delta x \to 0}$

$$f(x) = f(x_0) + \frac{f'(x)}{1!}(x - x_0) + \frac{f''(x)}{2!}(x - x_0)^2 + \cdots \quad (3^*)$$ Fig4

$$f(x_i^{n+1}) - f(x_i^n) < 0 \quad \cdots (5)$$

$\Delta x_i^n = ?$

Root finding
Minimization



3° For N-Dimensional Case

$\underbrace{w_1, w_2, \cdots, w_{N_1}; b_1, b_2, \cdots b_m}_{N = N_1 + M}$

Rewrite $b_1, b_2, \cdots, b_m$ as w_i

$i = N_1+1, N_1+2, \cdots, N_1+M$

$L(w, b)$

$\downarrow$

$L(\underline{w})$

Partial Derivatives

One-Dimensional Case $f'(x) = \frac{d}{dx} f(x)$

N-Dimensional Case

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

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

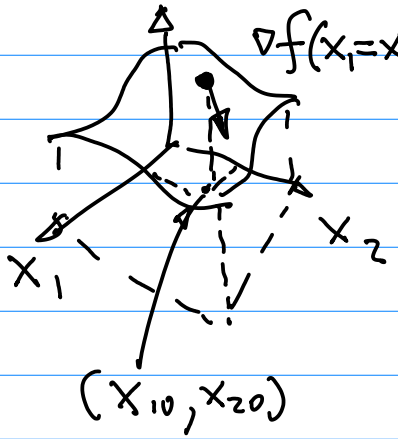
$$\cdots$$

$$\frac{\partial}{\partial x_N} f(x_1, x_2, \cdots, x_N) \quad \cdots (4)$$

Find $x_i^{n+1} = x_i^n + \Delta x_i^n$

4° Gradient

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



5° Update Rule

$(x_1^{n+1}, x_2^{n+1}, \cdots x_n^{n+1})$ New updated Values

Based on Gradient with "—"

$$-\eta \nabla f = -\eta \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \quad \cdots (7)$$

$$f(x_1,x_2,\cdots,x_n)=f(a_1,a_2,\cdots,a_n)+\frac{\partial f}{\partial x_1}(x_1-a_1)+\frac{\partial f}{\partial x_2}(x_2-a_2)+\cdots+\frac{\partial f}{\partial x_n}(x_n-a_n)+\text{higher Order Term} \quad \cdots(8)$$

Remove higher Order terms,

$$f(x_1,x_2,\cdots,x_n)\cong f(a_1,a_2,\cdots,a_n)+\frac{\partial f}{\partial x_1}(x_1-a_1)+\frac{\partial f}{\partial x_2}(x_2-a_2)+\cdots+\frac{\partial f}{\partial x_n}(x_n-a_n) \quad \cdots(8x)$$

$$f(x_1,x_2,\cdots,x_n)-f(a_1,a_2,\cdots,a_n)=\cdots$$

"<0"

$$f(x_1,x_2,\cdots,x_n)<f(a_1,a_2,\cdots,a_n) \quad \cdots(9)$$ We want

RHS $=$

$$\frac{\partial f}{\partial x_1}(x_1-a_1)+\frac{\partial f}{\partial x_2}(x_2-a_2)+\cdots+\frac{\partial f}{\partial x_n}(x_n-a_n)$$

Let $\Delta x_1=-\frac{\partial f}{\partial x_1}$, $\cdots$

$\Delta x_n=-\frac{\partial f}{\partial x_n}$ $\quad \cdots(11)$

Substitute Eqn(11) Back to Eqn(10)

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

$$=\left(-\frac{\partial f}{\partial x_1},-\frac{\partial f}{\partial x_2},\cdots,-\frac{\partial f}{\partial x_n}\right)\begin{pmatrix}\frac{\partial f}{\partial x_1}\\ \vdots \\ \frac{\partial f}{\partial x_n}\end{pmatrix}$$

$$=-\left[\left(\frac{\partial f}{\partial x_1}\right)^2+\cdots+\left(\frac{\partial f}{\partial x_n}\right)^2\right] \quad \cdots(12)$$

"≤ 0"

$$(x_1-a_1,\cdots,x_n-a_n)\cdot\begin{bmatrix}\frac{\partial f}{\partial x_1}\\ \frac{\partial f}{\partial x_2}\\ \vdots \\ \frac{\partial f}{\partial x_n}\end{bmatrix} \quad \cdots(10)$$

$\Delta x_1 \cdots \Delta x_n$

Gradient

from Eqn(11) (Handout, Note1, pp.1)

Feb25 (Thu)

Topics:

1° MNIST Convolution Neural Network — Sample Code, github

2° Pre-processing Techniques (OpenCV) to prepare Images/Videos for MNIST

Ref:

1° 20-20215-4-gradient-

Typo: "−" sign should be "+"

$$\frac{\partial}{\partial x_1}(x_1^2 + x_1 x_2) = \frac{\partial}{\partial x_1}(x_1^2) + \frac{\partial}{\partial x_1}(x_1 x_2)$$

$$= 2x_1 + x_2 \;;$$

then

$$\frac{\partial}{\partial x_2} f = \frac{\partial}{\partial x_2}(x_1^2 + x_1 x_2) = \frac{\partial}{\partial x_2} x_1^2 + \frac{\partial}{\partial x_2} x_1 x_2$$

$$= 0 + x_1 = x_1$$

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

2° 3 Sample Code for Tensors/Vectors github

3° Base Reference Code for MNIST CNN (Sample code)

TK Keras Implementation of MNIST CNN.

1° Architectural of the CNN

≙ Basic Building Block: Convolution layer

model.Summary in Keras to print the CNN Architecture

Conv2D - 1

"C" ↔ No. of the layers

max pulling

A technique to Reduce Image Resolution By Selecting the Largest Number from 2x2 Image Patch

Example: Max-pulling Hand-Calculation

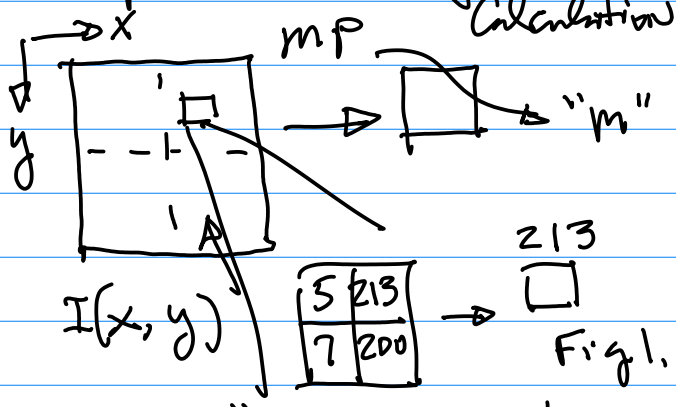


Fig 1.

"Better way" → Max → Higher Frequency Comp. ← Edge or Boundary of Object

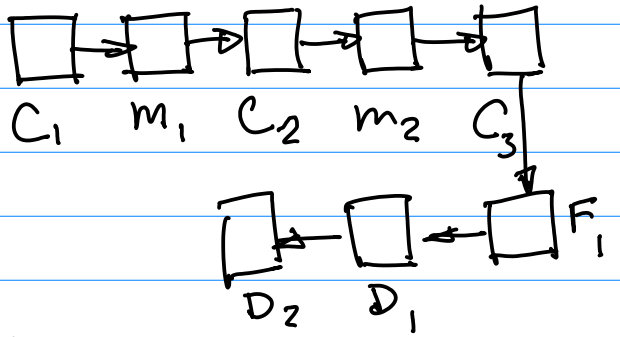
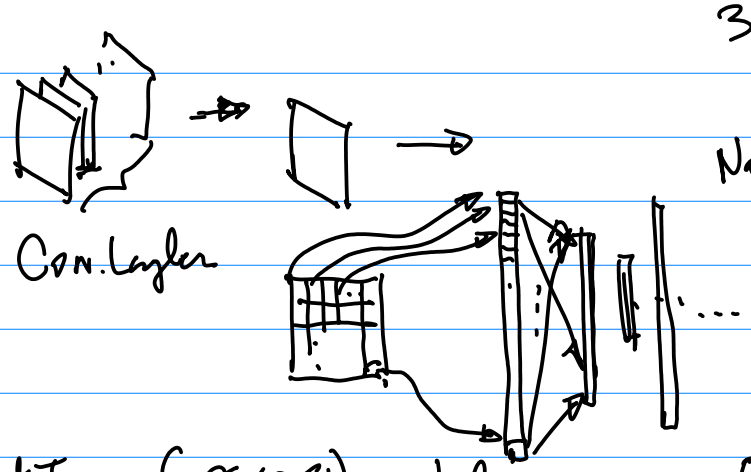


Fig 2. $C_1 M_1 C_2 M_2 C_3 F_1 \theta_1 \theta_2$

Note

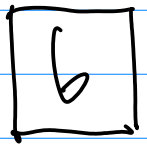


3. Sample code
models.sequential( )

Note: Dataset { from Repos.
from Internet
from google
Create the data by yourself

OpenCV → { I_i } is partition into { Training, Labeling / Testing, Labeling

Annotation

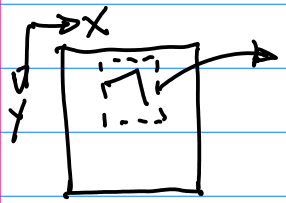


→ Label I_i "b"

Input Image (28×28) → 1 layer plane

Gray Scal Image

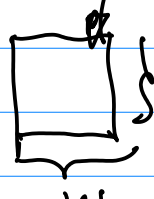
Example: From Python Program.



1° Localize the ROI (Region of Interest)

2° ROI: $I_{ROI}(x,y)$ W, H

1080P
1920×1080



W.X H → M × N

Aspect Ratio

$W/H = \alpha$

Preprocessing Using OpenCV for Handwritten Digits Recognition

Step 1. Input Image/Video

Localization of ROI.

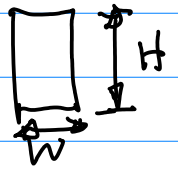
1. Binarization of Image

$I(x,y;t) =$
$(r(x,y;t), g(x,y;t), b(x,y;t)) \rightarrow$ Convert it to gray scale Image

$I_{gray}(x,y;t) \rightarrow I(x,y)$

Preserve the Aspect Ratio

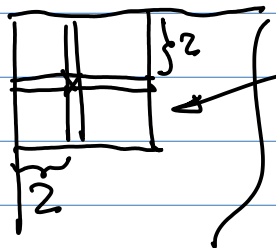
$I_{ROI}(x,y) \rightarrow$ Square Image



↓

Resize the Image to 28×28

Note: Each Convolution → Produce One output (One Image Plane)



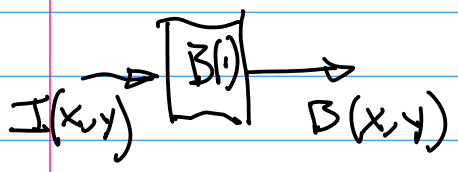
Kernel K×K 5×5

Top Row − 2
Left Col. − 2
↓
B. Row − 2
B. Col − 2
} 4
} 4

$I(x,y)$ Gray Scale

Binarization

$$B(x,y) = \begin{cases} 255 & I(x,y) \geq T_1 \\ 0 & O/W \end{cases} \quad \cdots (1)$$



Example:

$I(x,y)$ $T=50$ $B(x,y)$

111	200	200
5	55	180
7	20	30

→

0	255	255
0	255	255
0	0	0

Note Use OpenCV for Binarization,

Consider using mean, Std

mean ± Std * η

as a Threshold.

Step 2. Ability to Detect the Shape of the Object(s)

Primitive Features of Given Image

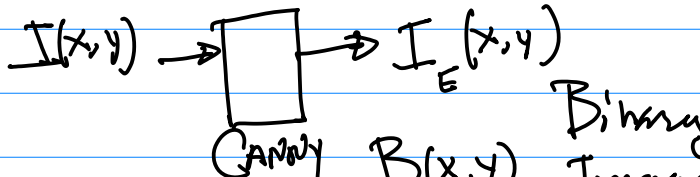
= Edge, Contours elements, Aspect Ratio, Area, moments, Shape Descriptor(s)

etc. $(\bar{x}, \bar{y})$ Centroid;

→ Size of the Object

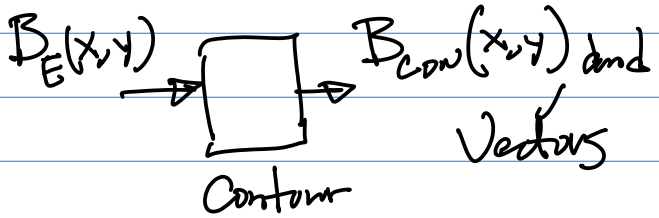
$I(x,y)$ → Canny Edge Detection → Edge Map

gray



$I(x,y) \rightarrow B_E(x,y) \rightarrow$ Contour Analysis

$B(x,y)$



Contours can be Computed By OpenCV function. (github. Sample)

4 - types of contours.

↓

External Contour.

↓

Moments

↓

$(\bar{x}, \bar{y})$, Area, W/H

x_1, x_2, x_3, x_4

$X = \{x_1, x_2, x_3, x_4\}$

Unity AI (D.R.L)

↓

(S.I.A.R)

Project Topics: Unity & ROS

1° DRL for 6 DoF Robot Pick & place Application

2° Yolo4/3 Pedestrian & Behavior Analysis (Pose Recognition)

3° Unity AI and Self Driving Based on Unity ml-Toolkit

4° LSTM Time Series Analysis for Abnormality Detection

2 weeks from Today (March 4)
Homework/Project 1: 10 pts
Handwritten Digits Recognition

March 4 (Thursday)
Topics: 1° Handout on Source Code Walk-Through
2° Project 1 & Pre-processing Technique

To Be Posted on CANVAS in 1~2 days

Note: (1) A Semester Long Project is A Team Project, e.g. 4-person Tem work together on your Project. And Submission is ALSO Team Submission. Due 1 week before the End of Semester

(2) Topics (4 Topics)

Ref: 20-2015-6- Handout 1. Architecture

Note: $f(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ ...(1)

$f(z_1) + f(z_2) + \cdots + f(z_{10})$

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

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

Therefore, they are equal to 1.

Example Input $[2.0, 1.0, 0.1]$

Find $Prob(z_1 = 2.0)$, $Prob(z_2 = 1.0)$, $Prob(z_3 = 0.1)$

Sol From Eqn (1)

$Prob(z_1) = \frac{e^{z_1}}{\sum_{j=1}^{3} e^{z_j}}$

$= \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$

$= \frac{e^{2.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = 0.7$ //

Note model.add { layers.Conv2D ... .Flatten

CMPE258

Note: Activation Functions { Softmax, RELU }

a. Math Formulation

b. Hand Calculation, c. T.F. Code

Discussion On Project 1. (10 pts)

1. Objectives

To Develop/Integrate CNN with Pre-processing capability to Realize the function of Hand Written Digits Recognition

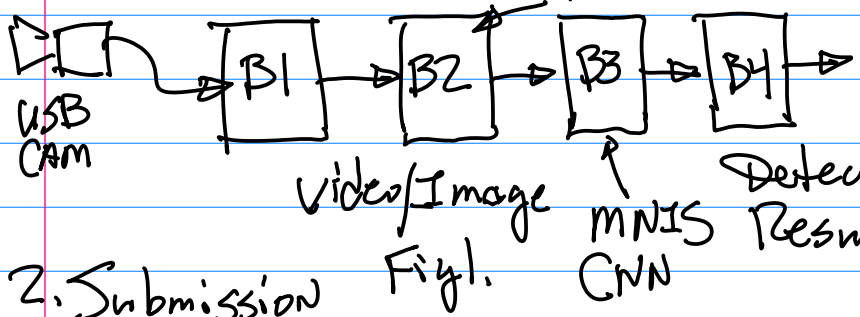


Fig1.

2. Submission

2.1. Readme.txt ; 2.2 Python Code ;

2.3 Video Clips (3~5 seconds) of the Result ;

3. Ref 20-2025-3-load_deployment_~

4. Preprocessing Techniques

Step1 VideoInput, → Step2 grayscale

Image Frame $I(x,y;t) \rightarrow I(x,y)$

Image Binarization $I_{gray}(x,y)$

$$B(x,y) = \begin{cases} 255 & \text{If } I(x,y) \geq T \\ 0 & \text{o/w} \end{cases} \quad \cdots (2)$$

Note: Perform CANNY Edge Detection Find

$$I_{CANNY}(x,y) = B_{CANNY}(x,y) = B(x,y)$$

Step3. Contour Computation

Contour Map → Draw your Result Back on $I_{gray}(x,y)$

Step4. $\bar{x}, \bar{y}$

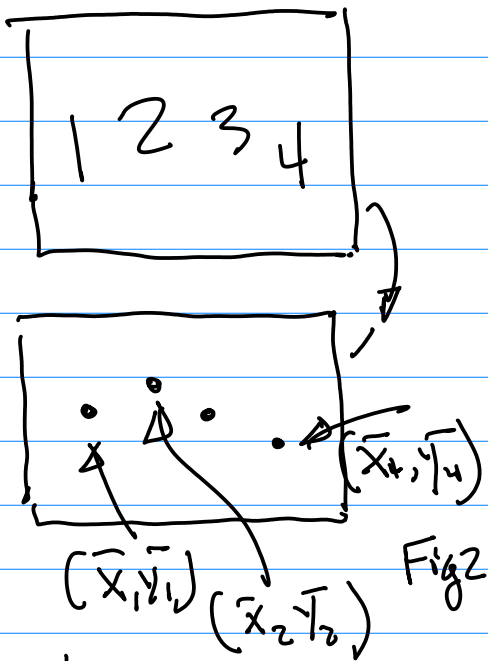


Fig2

Step5 Build Bounding Boxes (ROI)

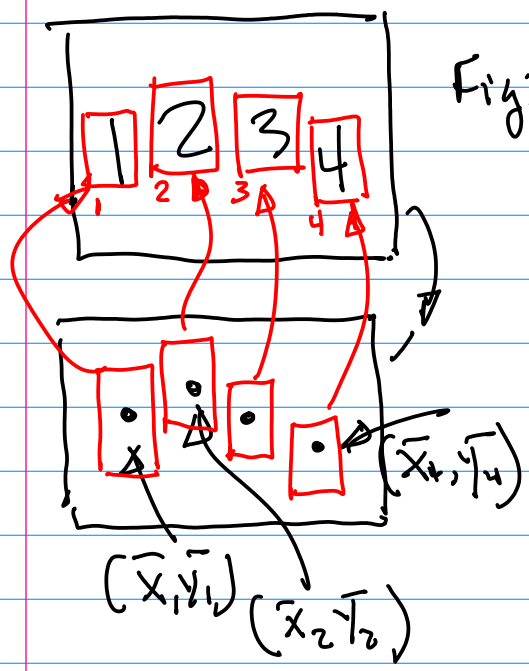


Fig3.

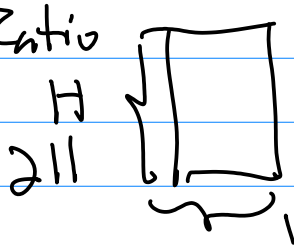
Use pre-Build function to find $\bar{x}, \bar{y}$, then use either pre-Build function from Contours or your self developed function (If self Developed, then use B(x,y)).

Find H, W for the Bounding Box

Step6. Resizing while keeping the Original Aspect Ratio

1° Aspect Ratio

H/W OR W/H



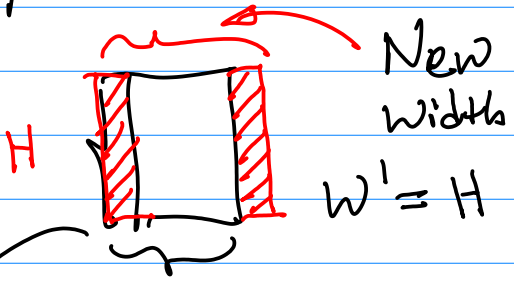
R.O.I

First make it a Squared Image,

$\max\{W, H\} \rightarrow D$

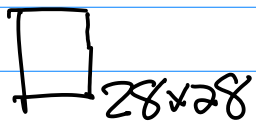
$\max\{125, 211\} = 211$

make your ROI 211x211.



Bitwise function from OpenCV to make it Square ROI

Then, Resize the Squared ROI.



Step7. From the deployment Code, Integrate your pre-processing stage Function, To complete the Project.