影像處理、電腦視覺及深度學習概論
(Introduction to Image Processing, Computer Vision and Deep Learning)

Homework 1

TA: 張議隆

Gmail: nckubot65904@gmail.com

Office Hour: 14:00~16:00, Mon.

10:00~12:00, Fri.

At CSIE 9F Robotics Lab.

Notice (1/2)

- Copying homework is strictly prohibited!! Penalty: Both individuals will receive a score of 0!!
- Due date => 09:00:00, 2023/11/07 (Tue.)

  Do not submit late, or the following points will be deducted:
  - Submit within seven days after the deadline, and your score will be reduced by half.
  - If you submit after this period, you will receive a score of 0.
- You must attend the demonstration, otherwise your score will be 0. The demonstration schedule will be announced on NCKU Moodle.
- You must create GUI, otherwise your point will be deducted.
- Upload to => 140.116.154.28 -> Upload/Homework/Hw1
  - User ID: opencvdl2023 Password: RL2023opencvdl
- Format
  - Filename: Hw1_StudentID_Name_Version.rar
    - Ex: Hw1_F71234567_林小明_V1.rar
    - If you want to update your file, you should update your version to be V2,
    - Ex: Hw1_F71234567_林小明_V2.rar
  - Content: Project folder *(Excluding the pictures)

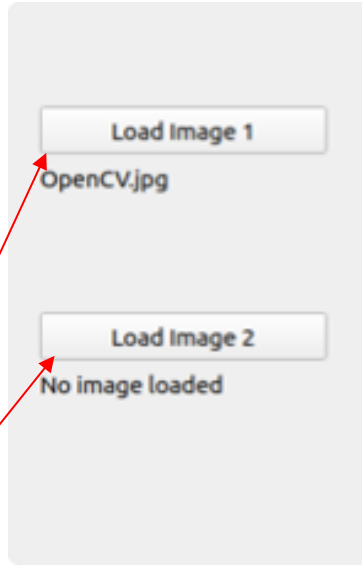
    *Note: Remove your “Debug” folder to reduce file size.

Notice (2/2)

- Python (recommended):
  - Python 3.8 (https://www.python.org/downloads/)
  - **Opencv-contrib-python (3.4.2.17)**
  - Matplotlib 3.7.3
  - UI framework: pyqt5 (5.15.10)
  - Pytorch 2.1.0
  - Torchvision 0.16.0
  - Torchsummary 1.5.1
  - Tensorboard 2.14.0
  - Pillow 10.1.0

Assignment scoring (Total: 100%)

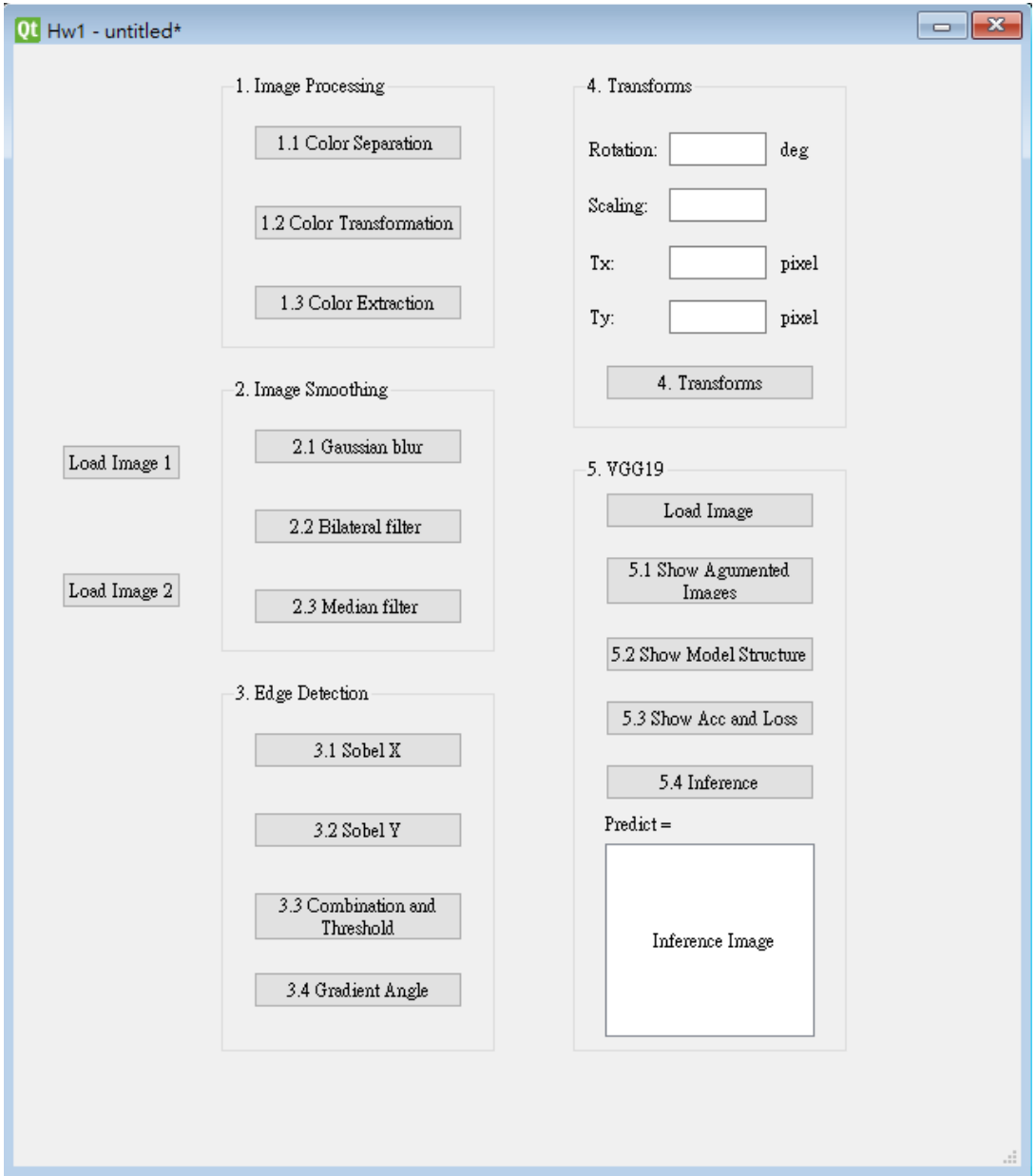
1. (20%) Image Processing (出題： Chen)
   - 1.1 (5%) Color Separation
   - 1.2 (5%) Color Transformation
   - 1.3 (5%) Color Extraction
2. (20%) Image Smoothing (出題：Shang)
   - 2.1 (6%) Gaussian blur
   - 2.2 (7%) Bilateral filter
   - 2.3 (7%) Median filter
3. (20%) Edge Detection (出題：Zhong)
   - 3.1 (5%) Sobel X
   - 3.2 (5%) Sobel Y
   - 3.3 (5%) Combination and Threshold
   - 3.4 (5%) Gradient Angle
4. (20%) Transforms (出題：Jimmy)
   - 4.1 (7%) Rotation
   - 4.2 (7%) Scaling
   - 4.3 (6%) Translate
5. (20%) Training a CIFAR10 Classifier Using VGG19 with BN (出題：Hsiang)
   - 5.1 (4%) Load CIFAR10 and Show 9 Augmented Images with labels.
   - 5.2 (4%) Load Model and Show Model Structure
   - 5.3 (6%) Show Training/Validating Accuracy and Loss
   - 5.4 (6%) Use the Model with Highest Validation Accuracy to Run Inference, Show the Predicted Distribution and Class Label.



* Don’t fix your image path
(There is another dataset for demonstration)
Load image 請用下面Function 來讀取路徑
QFileDialog.getOpenFileName
獲取打開的檔路徑

Assignment scoring (Total: 100%)

- Use one UI to present 5 questions.



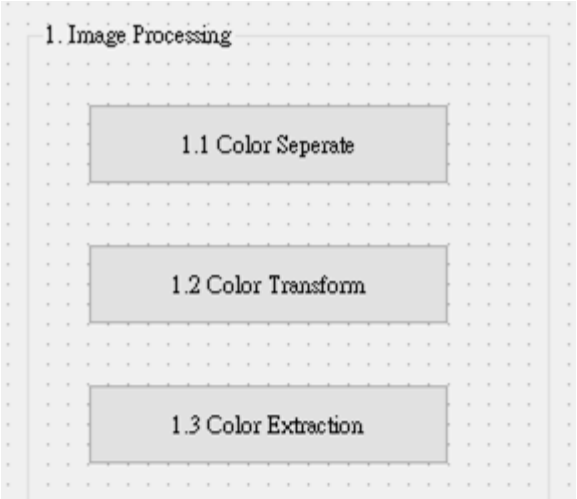
1. Image Processing (20%)

(出題：Chen)

1.1 (6%) Color Separation

1.2 (6%) Color Transformation

1.3 (8%) Color Extraction



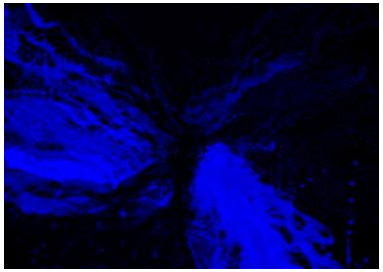
1.1 Color Separation (6%)

(出題：Chen)

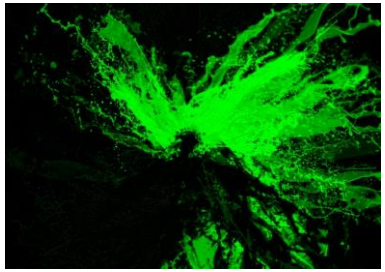
- Given: a color image, “rgb.jpg”
- Q: Extract 3 channels of the image BGR and show the result images.
  1) Use cv2.split() to get R G B gray scale images.
  2) Use cv2.merge() to turn each gray scale image back to bgr image.

Please show each R ,G ,B Image.

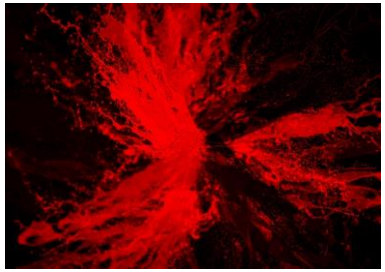
rgb.jpg



B channel



G channel



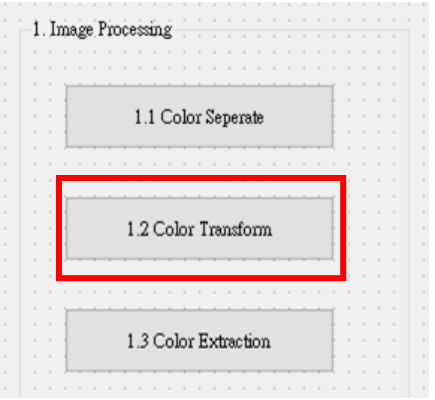
R channel

1.2 Color Transformation (6%)

(出題：Chen)

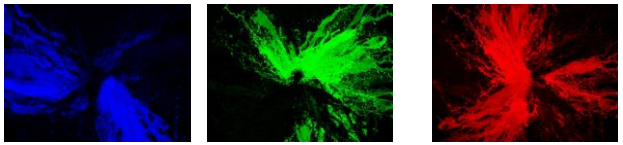
- Given: 1 color image: “rgb.jpg”
- Q: Transform “rgb.jpg” into grayscale by
  - Q1): Calling OpenCV function cv2.cvtColor(…, cv2.COLOR_BGR2GRAY) on rgb.jpg to generate Image I_1
  - Q2): Merge BGR separated channel images from problem 1.1 to generate $I_2 = (R+G+B)/3$.

Please show above 2 images results.



rgb.jpg

Result from problem 1.1

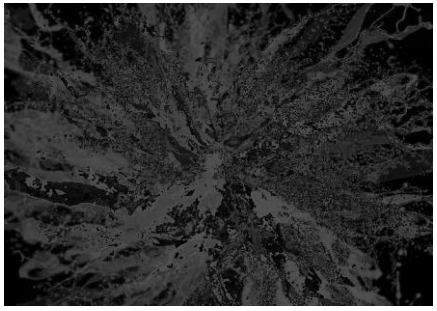


B channel　G channel　R channel

1) I_1 : OpenCV function

Perceptually weighted formula:
$I_1 = 0.299\ R + 0.587\ G + 0.114$

2) I_2 : Average



Average weighted formula:
$I_2 = (R+G+B)/3$

1.3 Color Extraction (8%)

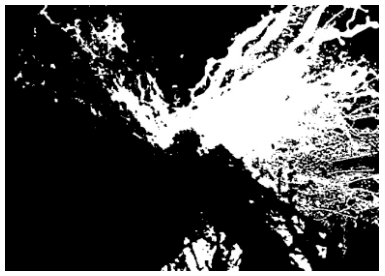
(出題：Chen)

➢Given: 1 color image: “rgb.jpg”

1) Transform “rgb.jpg” from BGR format to HSV format: cv2.cvtColor(bgr img, cv2.COLOR_BGR2HSV)
2) Extract **Yellow-Green** I_1 **mask** by calling : **cv2.inRange(hsv img , lower bound , upper bound**)
3) Turn **Yellow-Green** mask into BGR format by calling : **cv2.cvtColor(…, cv2.COLOR_GRAY2BGR)**
4) Remove **Yellow** and **Green** color in the image to generate I_2 by calling : **cv2.bitwise_not(mask bgr ,bgr img ,mask)**

Please show above 2 image results.

rgb.jpg



mask I_1

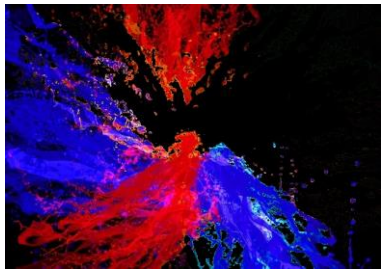
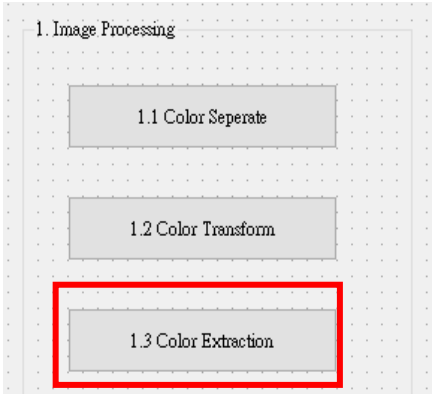


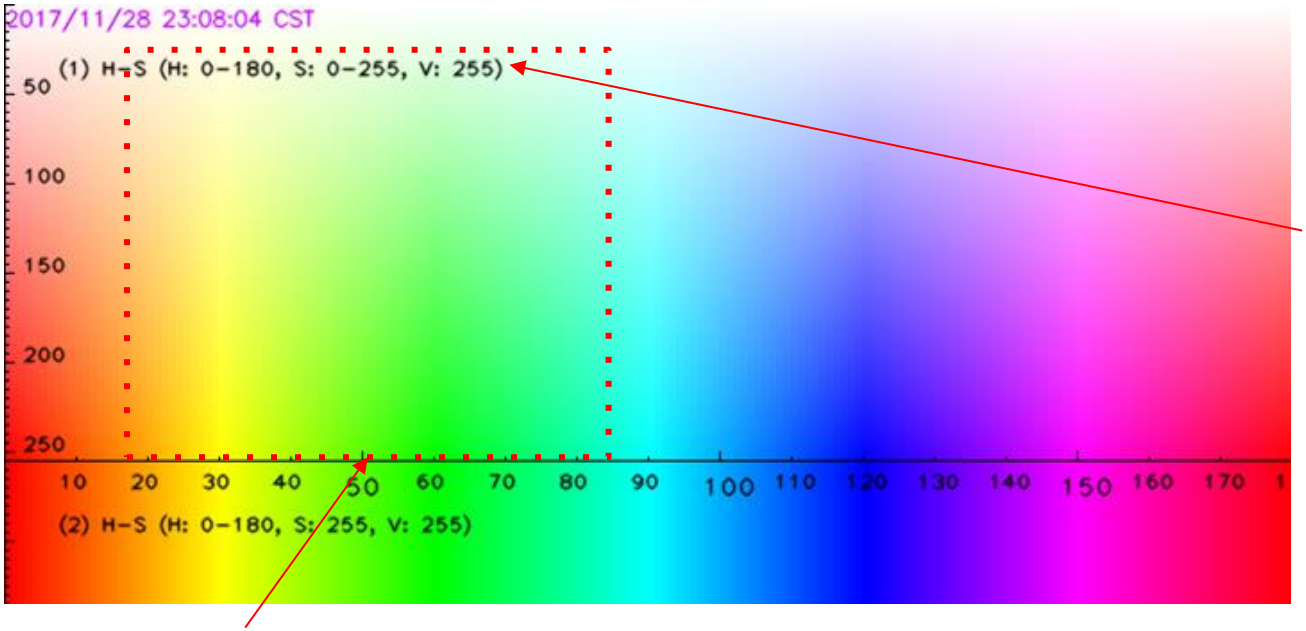
Image without yellow and green I_2



1.3 Color Extraction (8%)

(出題：Chen)

- Hint:
  - cv2.inRange(hsv_img , lower_bound(h,s,v=25) , upper_bound (h,s,v=255))
    - Hue and Saturation pleas check below chart area ,Value is between (25,255)

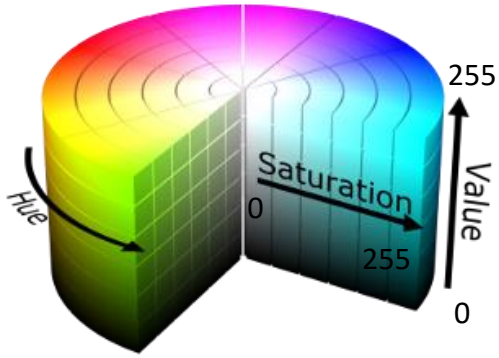


Yellow-Green mask range

Slight difference on range should be fine

HSV values ranges between (h:0–180, s:0–255, v:0–255)

H(Hue) : x axis
S(Saturation) : y axis
V(Value) : 255



REF: https://medium.com/@gowtham180502/how-to-detect-colors-using-opencv-python-98aa0241e713

2. Image Smoothing (20%)

(出題：Shang)

2.1 (6%) Gaussian blur

2.2 (7%) Bilateral filter

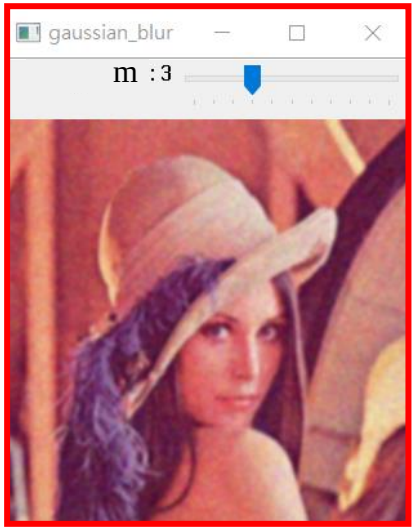
2.3 (7%) Median filter

- Hint
  1) Textbook Chapter 3, p. 50 ~ 52, p.109~115

2.1 Gaussian Blur

(出題：Shang)

1. Given: “image1.jpg”
2. Requirement:
   1) Using “Load image 1” button to load image.
   2) Click “2.1 Gaussian Blur” to show the popup window.
   3) Using cv2.createTrackbar() to create a trackbar on popup window.
   4) Using trackbar to change the window radius (m).
   5) The range of radius size is [1, 5].
   6) Apply gaussian filter (cv2.GaussianBlur()) which kernel size is $(2m + 1) \times (2m + 1)$



Gaussian Blur

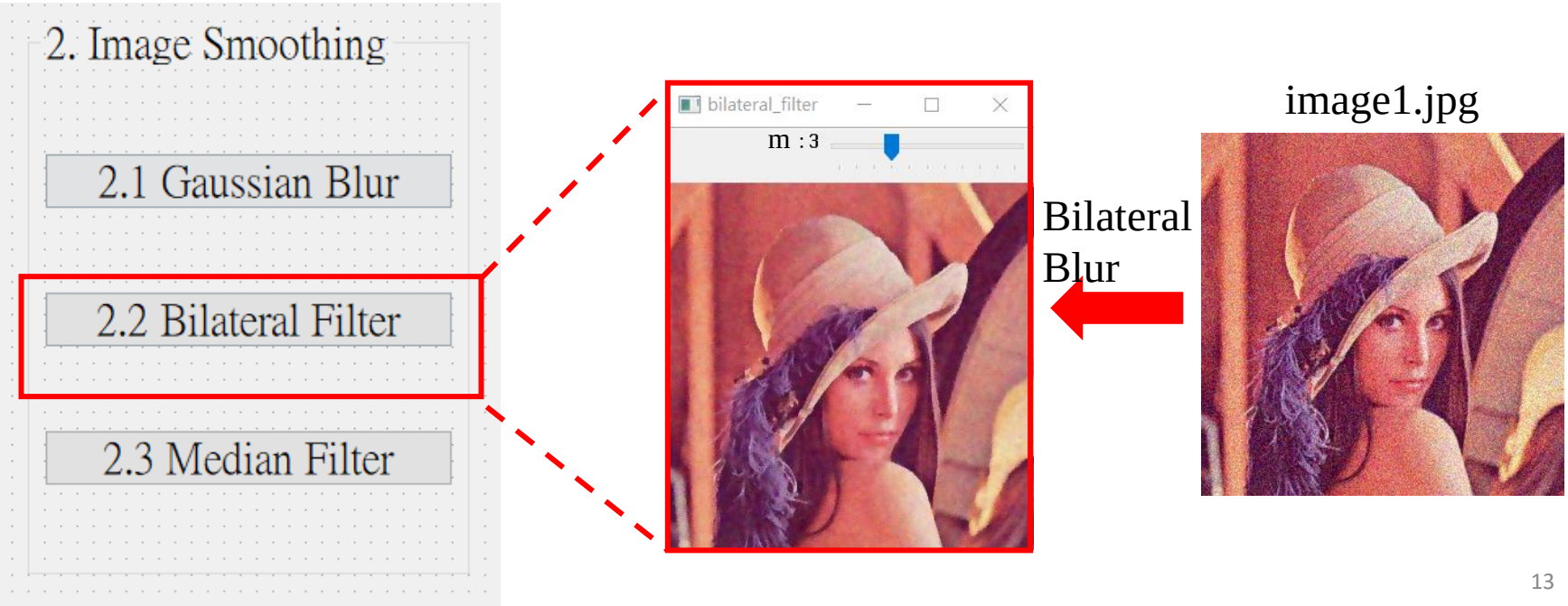
image1.jpg

2.2 Bilateral Filter

(出題：Shang)

1. Given: “image1.jpg”
2. Requirement:
   1) Using “Load image 1” button to load image.
   2) Click “2.2 Bilateral Filter” to show the popup window.
   3) Using cv2.createTrackbar() to create a trackbar on popup window.
   4) Using trackbar to change the window radius (m).
   5) The range of radius size is [1, 5].
   6) Apply bilateral filter (cv2. bilateralFilter()) which kernel size is $(2m + 1) \times (2m + 1)$ to

- Hint
  1) simgaColor = 90, sigmaSpace = 90



2.3 Median Filter

(出題：Shang)

1. Given: “image1.jpg”
2. Requirement:
   1) Using “Load image 1” button to load image.
   2) Click “2.3 Median Filter” to show the popup window.
   3) Using cv2.createTrackbar() to create a trackbar on popup window.
   4) Using trackbar to change the window radius (m).
   5) The range of radius size is [1, 5].
   6) Apply median filter (cv2. medianBlur()) which kernel size is $(2m + 1) \times (2m + 1)$ to

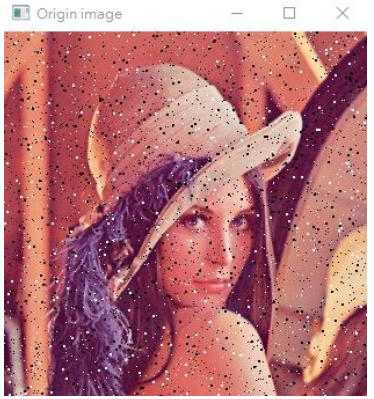
223	186	114
204	161	106
219	194	138

106 114 138 161 **186** 194 204 219 223

Median filter example

Bilateral Blur

image1.jpg



3. Edge Detection (20%)

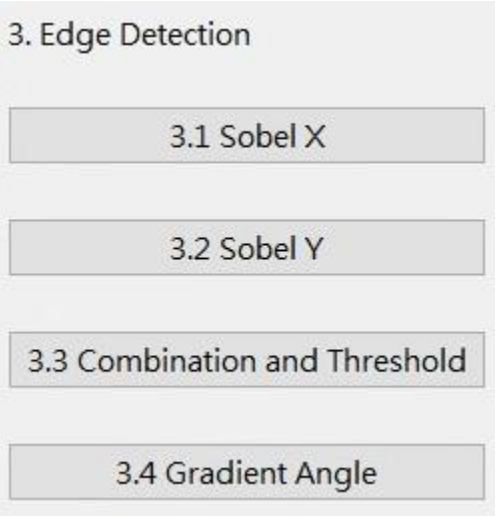
(出題：Zhong)

3.1 (5%) Sobel X

3.2 (5%) Sobel Y

3.3 (5%) Combination and Threshold

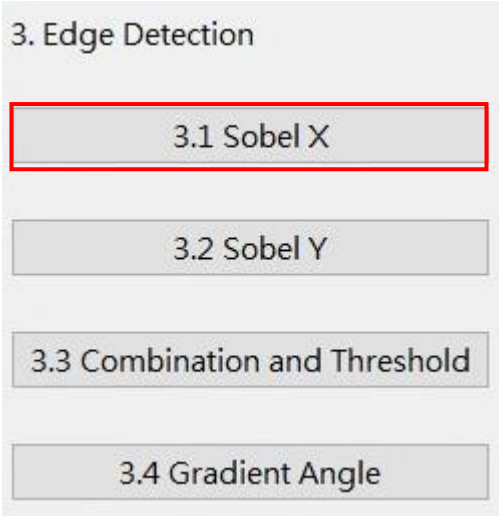
3.4 (5%) Gradient Angle



3.1 Sobel x (5%)

(出題：Zhong)

1. Given: A RGB image, “building.jpg”
2. Q: Generate Sobel x image for “building.jpg”
   1) Convert the RGB image into a grayscale image
   2) Smooth grayscale image with Gaussian smoothing filter.
   3) Use Sobel edge detection to detect vertical edge by your own 3x3 Sobel x operator. (Can not use OpenCV Function cv2.Sobel and cv2.filter2D.)
   4) Please show the result with cv2.imshow function.
- Hint: Textbook Chapter 6, p.144 ~ 149



building.jpg

Grayscale

Gaussian Smoothing

-1	0	1
-2	0	2
-1	0	1

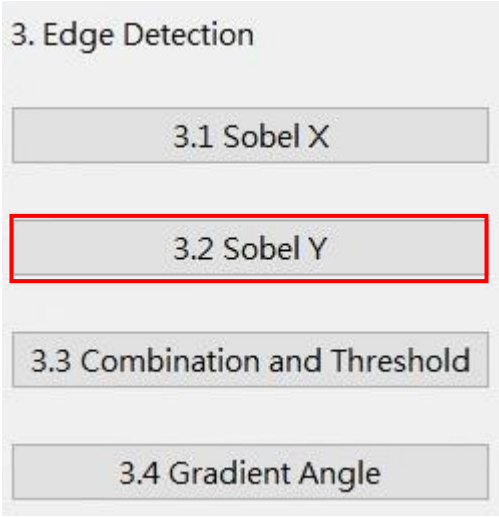
Sobel x Filter

Sobel x

3.2 Sobel y (5%)

(出題：Zhong)

1. Given: A RGB image, “building.jpg”
2. Q: Generate Sobel y image for “building.jpg”
   1) Convert the RGB image into a grayscale image
   2) Smooth grayscale image with Gaussian smoothing filter.
   3) Use Sobel edge detection to detect horizontal edge by your own 3x3 Sobel x operator. (Can not use OpenCV Function cv2.Sobel and cv2.filter2D.)
   4) Please show the result with cv2.imshow function.
- Hint: Textbook Chapter 6, p.144 ~ 149



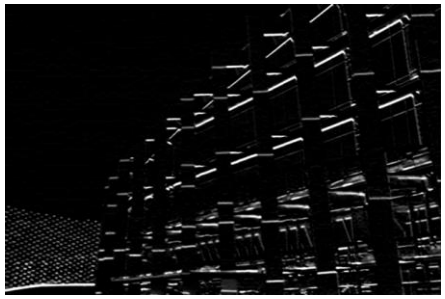
building.jpg

Grayscale

Gaussian Smoothing

-1	-2	-1
0	0	0
1	2	1

Sobel y Filter



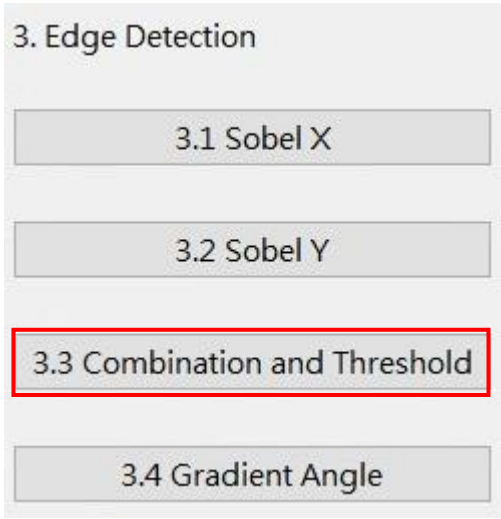
Sobel y

3.3 Combination and Threshold (5%)

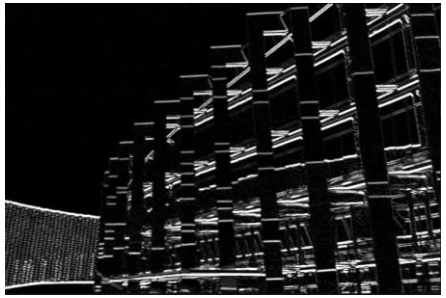
(出題：Zhong)

1. Given: The result of 3.1) Sobel x and 3.2) Sobel y
2. Q: Combine Sobel x and Sobel y, then set threshold for result
   1) New value of pixel = $\sqrt{Sobel_X{}^2 + Sobel_Y{}^2}$
   2) Normalize combination result to 0~255
   3) Given threshold 128. Set to 0 if pixel value is lower than threshold, otherwise, set to 255.
   4) Show both **combination** and **threshold result** with cv2.imshow function. Two results should be shown together.

☐ Hint: Textbook Chapter 6, p.148 ~ 149

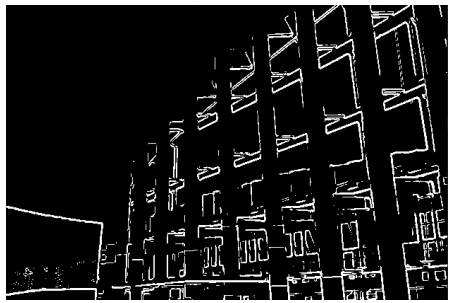


Sobel x



Sobel y

Combination of
Sobel x and Sobel y

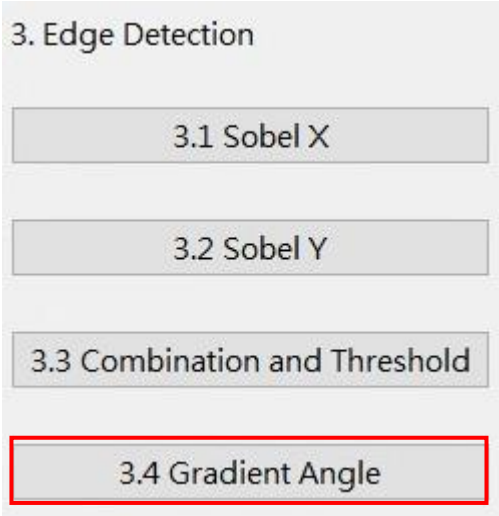


Threshold result

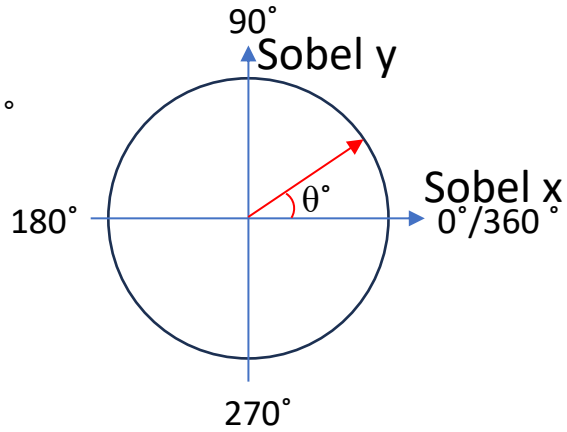
3.4 Gradient Angle (5%)

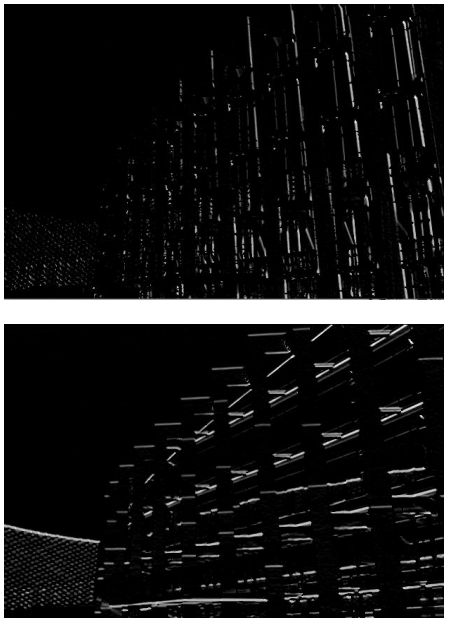
(出題：Zhong)

1. Given: The result of 3.1) Sobel x and 3.2) Sobel y
2. Q: Calculate the gradient angle and show specific range of angle.
   1) Calculate the gradient angle by result of Sobel x and Sobel y
   2) Generate **two different masks** by given two different range of angle (1) 120˚~180˚ (2) 210˚~330˚. Set to 255 if pixel value is in range, otherwise set to 0.
   3) Generate results by calling cv2.bitwise_and(**combination**, **mask**) which **combination** is the result form 3.3
   4) Show both results with cv2.imshow function. Two results should be shown together.



- Hint: Gradient angle θ˚





Output image

4. Transforms (20%)

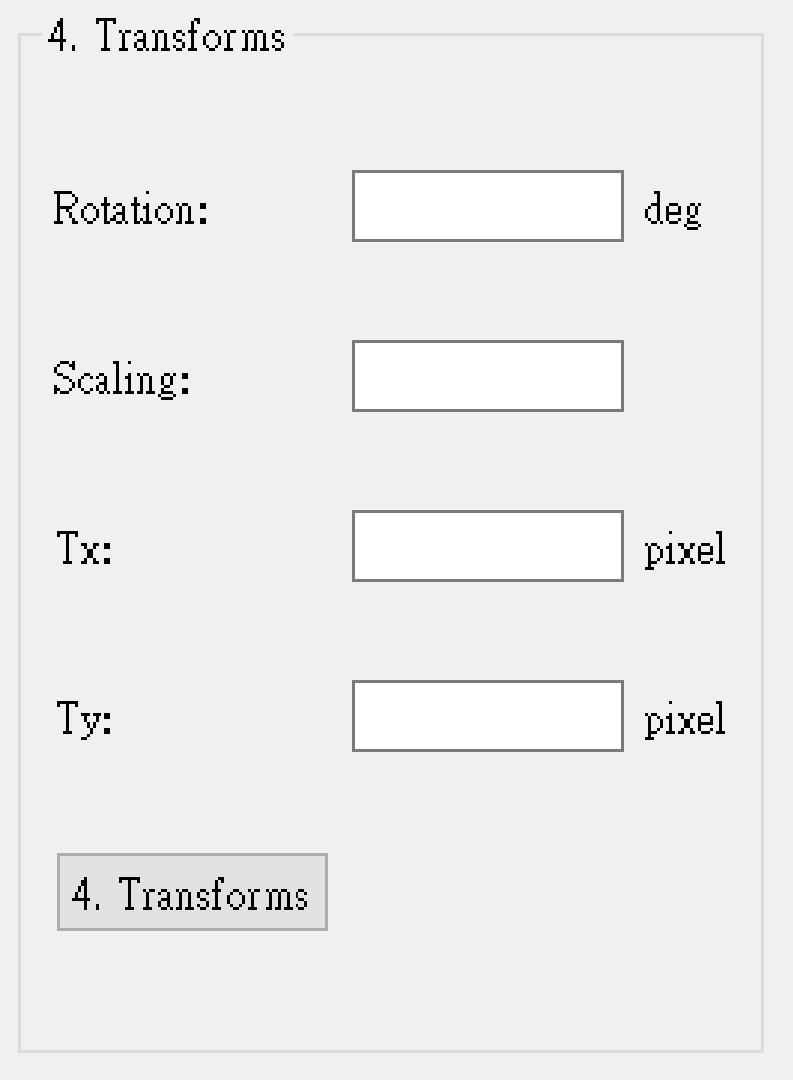
(出題：Jimmy)

4.1 (7%) Rotation

4.2 (7%) Scaling

4.3 (6%) Translate

UI Demo:



4. Transforms (20%)

(出題：Jimmy)

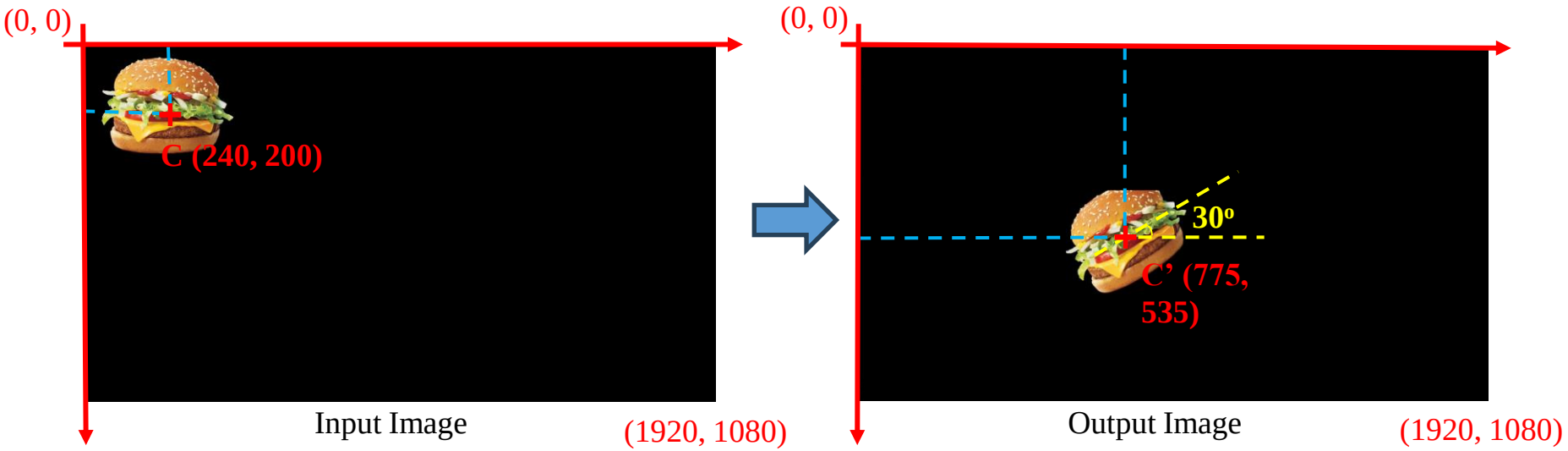
1. Given: “burger.png”
2. Q: 1) Click button “4. Transforms”, burger.png should be showed.
   2) Please rotate, scale and translate the burger (as image below) using cv.warpAffine() function with following parameters (set default values 0, should be manually adjusted in the GUI)
   (1) Angle = 30° (positive degree counter-clockwise)
   (2) Scale = 0.9,
   (3) Translation with:
   - $X_{new} = X_{old}$ + 535 pixels = 240 + 535 = 775
   - $Y_{new} = Y_{old}$ + 335 pixels = 200 + 335 = 535

   - Point C (240, 200) is center point of burger in original image
   - Point C’(775, 535) is center point of burger in result image

➢ Hint: Textbook Chapter 12, (p.407 ~ 412)
python: cv.warpAffine()

- Rotation, Scale: Object center not move
- Translation: Object center move

Result Demo:



5. Training a CIFAR10 Classifier Using VGG19 with BN (20%)

(出題：Hsiang)

5.1 Load CIFAR10 and show 9 Augmented Images with Labels. (4%)

5.2 Load Model and Show Model Structure. (4%)

5.3 Show Training/Validating Accuracy and Loss. (6%)

5.4 Use the Model with Highest Validation Accuracy to Run Inference, Show the Predicted Distribution and Class Label. (6%)

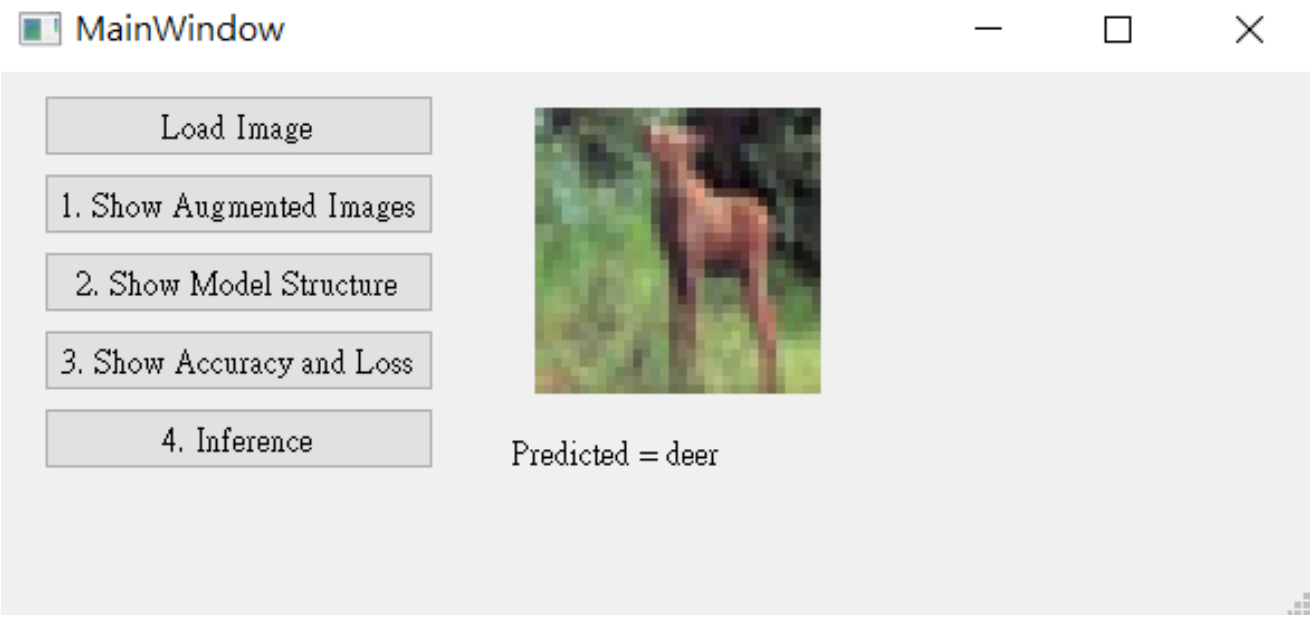


Figure: GUI example

5. Training a CIFAR10 Classifier Using VGG19 with BN (20%)

(出題：Hsiang)

1. Objective

1) Learn how to train a VGG19 with BN (Batch Normalization) model to classify 10 different classes images of CIFAR10.

2. VGG19 with BN

1) VGG19: A convolutional neural network that is 19 layers deep.
2) BN (Batch Normalization): used to make training of artificial neural networks faster and more stable.

3. CIFAR10

1) A collection of 60,000 32x32 color images in 10 different classes that is commonly used to train machine learning and computer vision algorithms.
2) 10 classes:
   airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck
3) Datasets
   (1) Training dataset: 50000 images in total.
   (2) Validation dataset: 10000 images in total.
   (3) Testing dataset: 10 images in total. (Generating from validation dataset.)

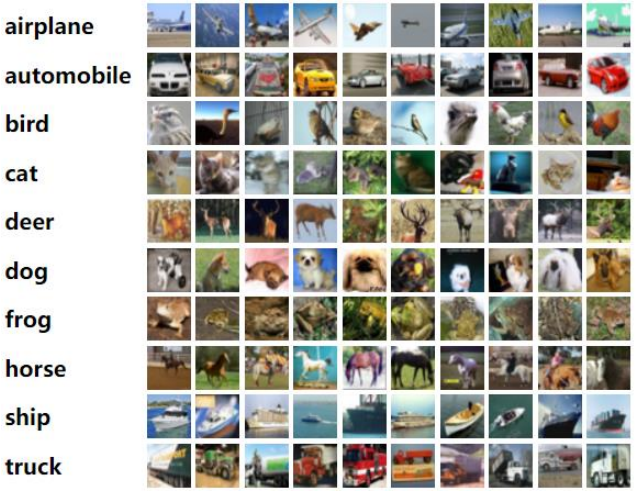


*Figure*1: CIFAR10

R. Reference

1) VGG19
2) Batch Normalization

5. Training a CIFAR10 Classifier Using VGG19 with BN (20%)

(出題：Hsiang)

4. Requirements

1) Train VGG model with batch normalization (BN) using PyTorch.
2) In the submitted file, you need to include
   A. Weight file for VGG19 with BN in .pth format. (File size is approximately 540MB)
   B. Figure of training/validating loss and accuracy in .jpg or .png format.
   C. Code for your GUI program
   D. Code for model training.
3) Please do not include image data in the submitted file.

5. Homework Images

1) There are 2 different folders in ‘Q5_image’.
2) In the subfolder ‘Q5_image/Q5_1,' there are 9 different images used in Q5-1. When demoing, use the same images.
3) In the subfolder 'Q5_image/Q5_4,' there are 9 different images used in Q5-4. These images are used for testing your program. When demoing, we will use different images for the demonstration.

5.1 Show 9 Augmented Images with Labels (3%)

(出題：Hsiang)

Q5.1

1) At home:

(1) Use PIL.Image.open() to load 9 images in /Q5_image/Q5_1/ folder.

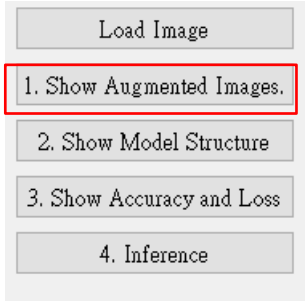
(2) Apply at least 3 different type of data augmentation (tutorial).

A. transforms.RandomHorizontalFlip()

B. transforms.RandomVerticalFlip()

C. transforms.RandomRotation(30)

Notice: This is an example; you can use different data augmentation techniques



2) When the demo:

(1) Click the button "1. Show Augmentation Images"

(2) Load 9 images in /Q5_image/Q5_1/ folder

(3) Apply data augmentation on 9 images.

(4) Show 9 augmented images with label in a new window

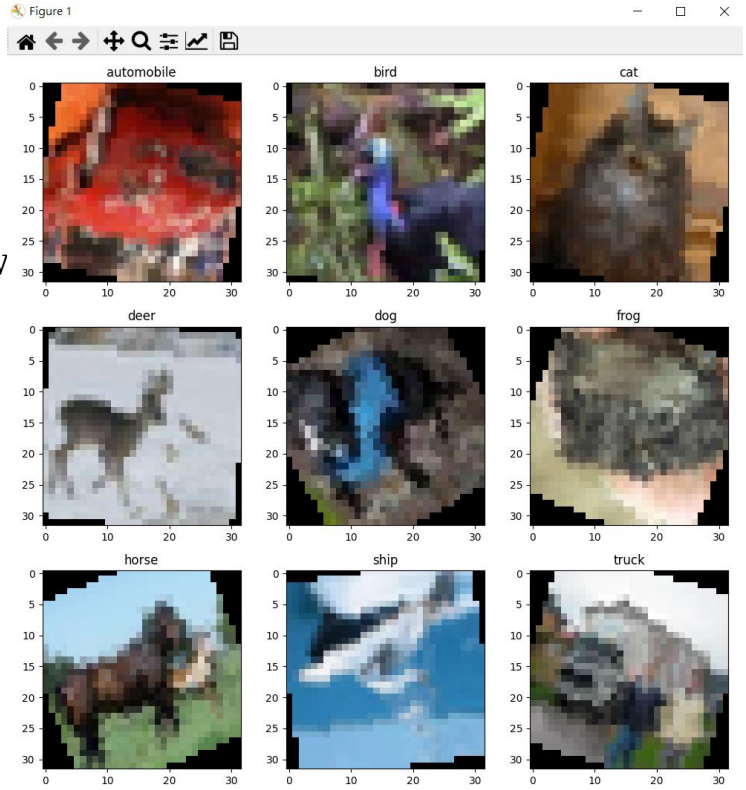


Figure1: 9 Augmented images

Notice: this is an example, the images might differ

5.2 Show the Structure of VGG19 with BN (3%)

(出題：Hsiang)

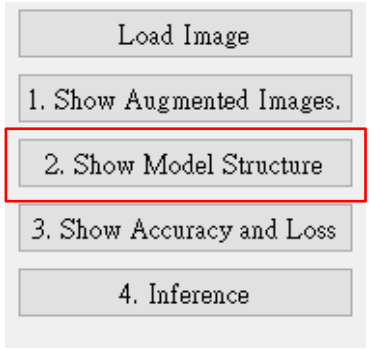
Q5.2

1) **At home:**
   (1) Use torchvision.models.vgg19_bn(num_classes=10) to build a VGG19 with batch normalization (BN) model.
   (2) Use torchsummary.summary to show the structure in the terminal.
2) **When the demo:**
   (1) Click the button "2. Show Model Structure"
   (2) Run the function to show the structure in the terminal.

The -1 indicates that the actual size of batch size can vary.

Feature map shape (Batch, Channels, Height, Width)

Layer (type) Num. of param.




```
      BatchNorm2d-38          [-1, 512, 4, 4]           1,024
             ReLU-39          [-1, 512, 4, 4]               0
          MaxPool2d-40          [-1, 512, 2, 2]               0
             Conv2d-41          [-1, 512, 2, 2]       2,359,808
        BatchNorm2d-42          [-1, 512, 2, 2]           1,024
             ReLU-43          [-1, 512, 2, 2]               0
             Conv2d-44          [-1, 512, 2, 2]       2,359,808
        BatchNorm2d-45          [-1, 512, 2, 2]           1,024
             ReLU-46          [-1, 512, 2, 2]               0
             Conv2d-47          [-1, 512, 2, 2]       2,359,808
        BatchNorm2d-48          [-1, 512, 2, 2]           1,024
             ReLU-49          [-1, 512, 2, 2]               0
             Conv2d-50          [-1, 512, 2, 2]       2,359,808
        BatchNorm2d-51          [-1, 512, 2, 2]           1,024
             ReLU-52          [-1, 512, 2, 2]               0
          MaxPool2d-53          [-1, 512, 1, 1]               0
AdaptiveAvgPool2d-54          [-1, 512, 7, 7]               0
             Linear-55                [-1, 4096]     102,764,544
             ReLU-56                [-1, 4096]               0
          Dropout-57                [-1, 4096]               0
             Linear-58                [-1, 4096]      16,781,312
             ReLU-59                [-1, 4096]               0
          Dropout-60                [-1, 4096]               0
             Linear-61                  [-1, 10]          40,970
================================================================
Total params: 139,622,218
Trainable params: 139,622,218
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 7.55
Params size (MB): 532.62
Estimated Total Size (MB): 540.18
----------------------------------------------------------------
```

Figure: the Structure of VGG19 with BN

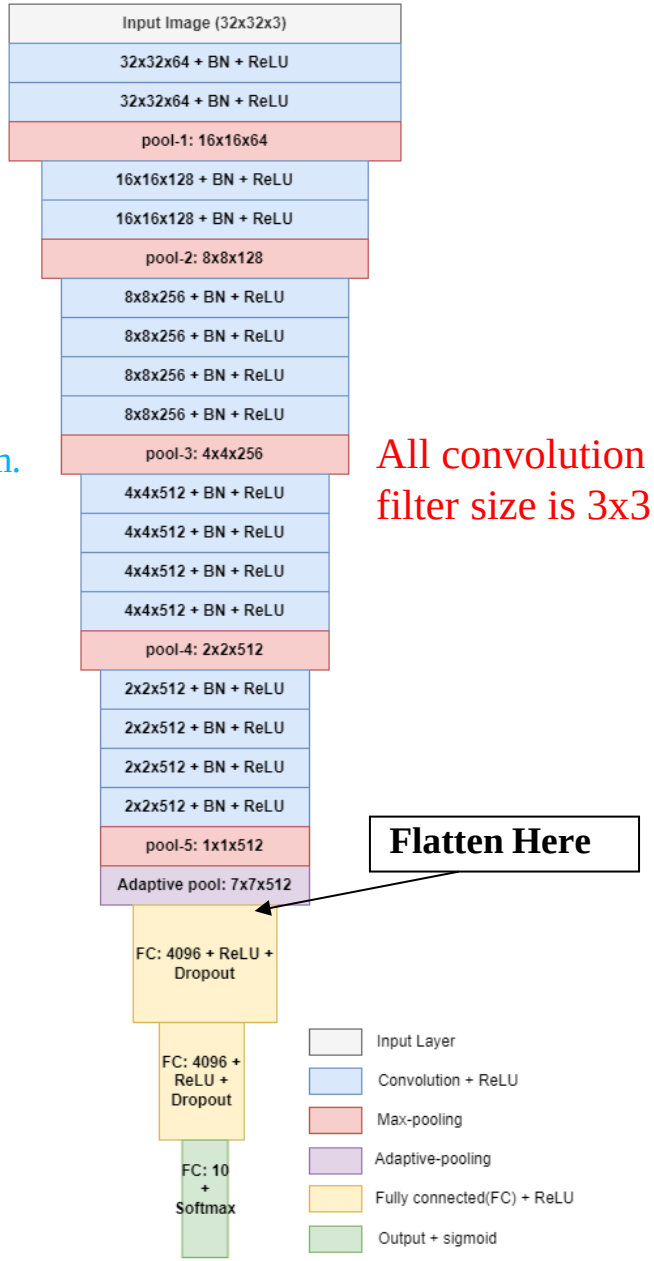


Figure: VGG19 with BN model structure

5.3 Show Training/Validating Accuracy and Loss (6%)

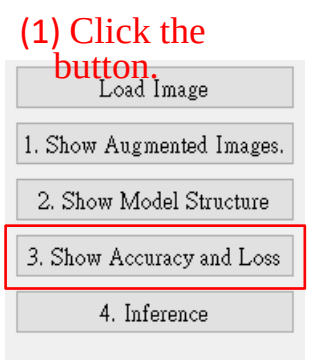
(出題：Hsiang)

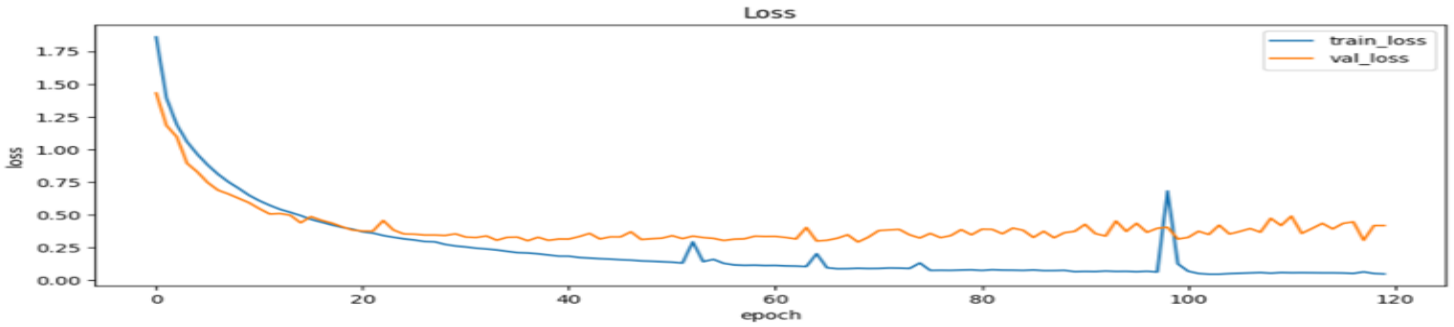
Q5.3

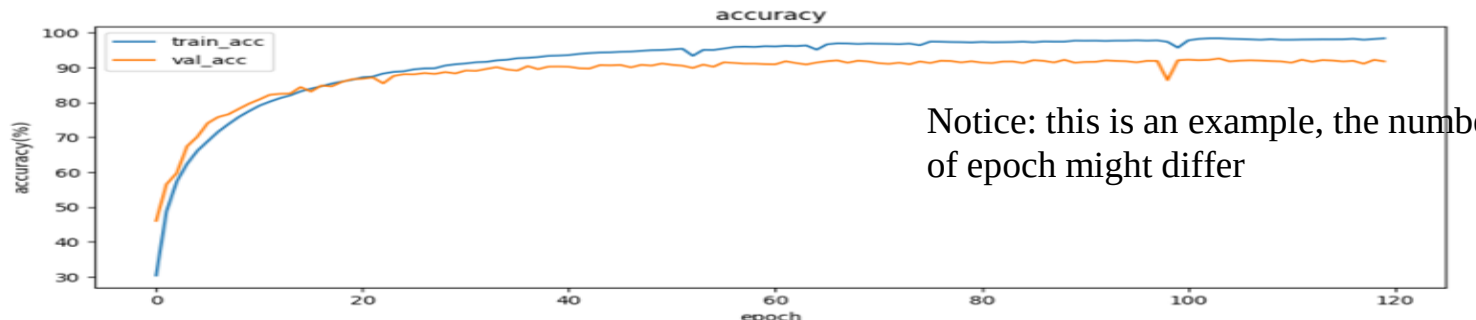
1) **At home:**
   (1) Use torchvision.datasets.CIFAR10 to load the training and validation datasets. (tutorial)
   (2) Training and validating VGG19 with BN at least 40 epochs at home (tutorial) and record the training/validating accuracy and loss in each epoch (tutorial).
   (3) Notice: If your validation accuracy is low, you can try
      A. Adjust the learning rate of the optimizer.
      B. Change the data augmentation techniques used.
   (4) Save weight file with highest validation accuracy .
   (5) Use matplotlib.pyplot.plot() to create a line chart for the training and validating loss and accuracy values.
   (6) Save the figure
2) **When the demo:**
   (1) Click the button "3. Show Accuracy and Loss"
   (2) Show the saved figure of Training/Validating loss and accuracy in a new window

(2) Show the figure in a new window

(1) Click the button.





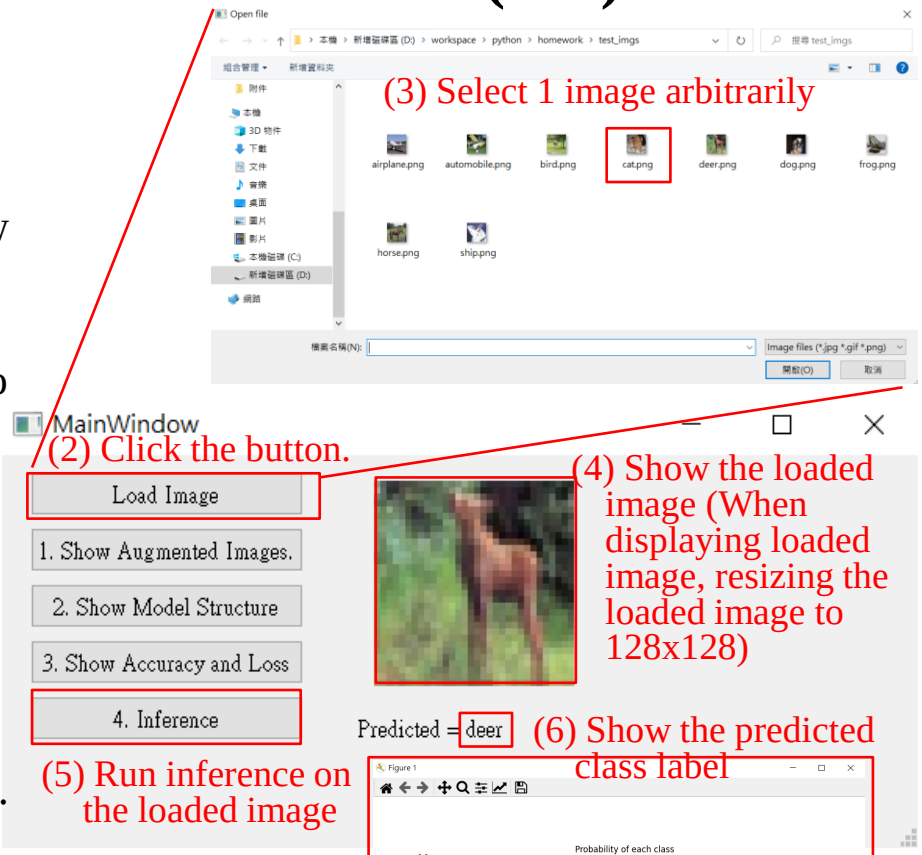


Notice: this is an example, the number of epoch might differ

5.4 Use the Model with Highest Validation Accuracy to Run Inference Show the Predicted Distribution and Class Label. (6%) (出題：Hsiang)

Q5.4

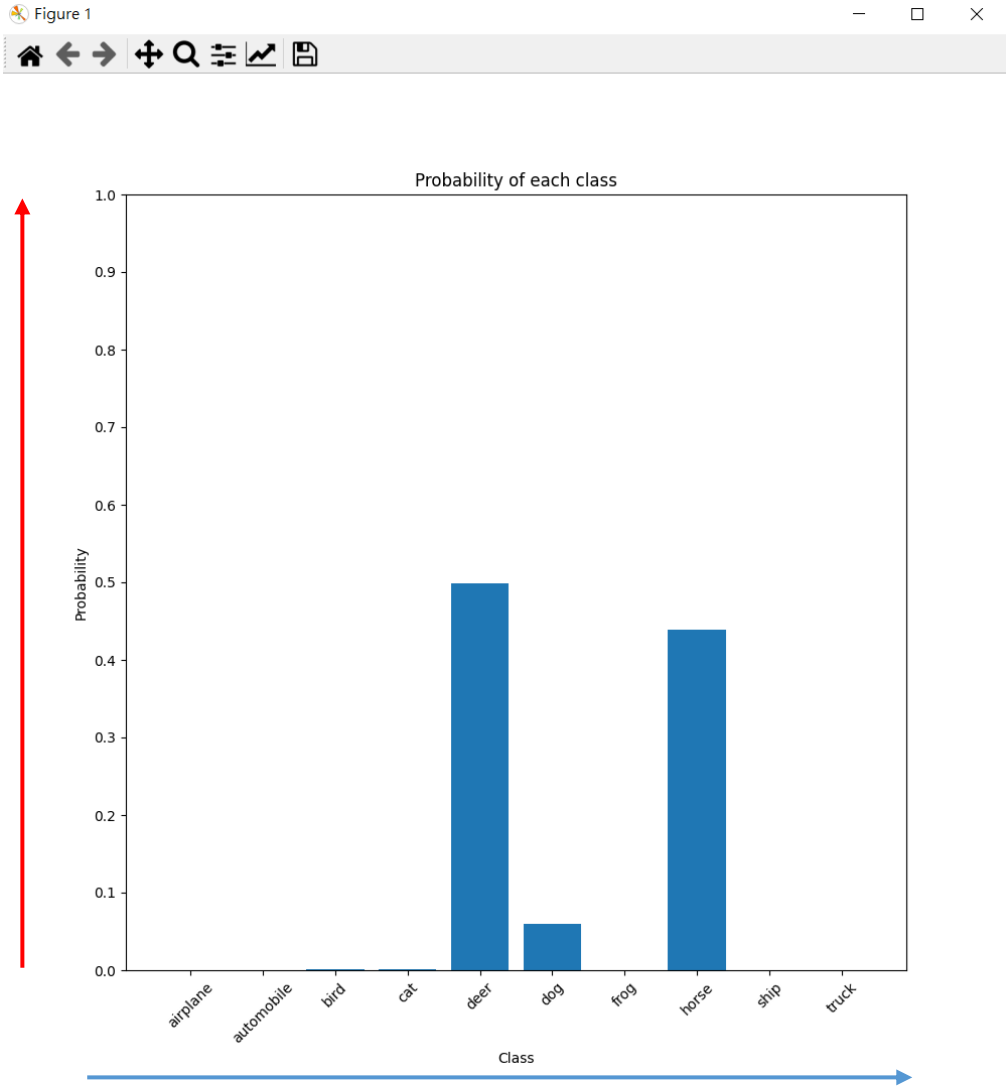
1) **At home:**
   (1) Load the model which trained at home
   (2) Click the button "Load Image" to display a new file selection dialog
   (3) Select 1 image arbitrarily.
   (4) Show the loaded image on the GUI. (In order to make it visually clear on the UI, use QtGui.Qpixmap.scaled to scale the image to 128x128 when displaying it.)
   (5) Click the button "4. Inference" to run inference on the image. (tutorial)
   (6) Show the predicted class label on the GUI.
   (7) Show the probability distribution of model predictions using a histogram in a new window.
2) **When the demo:** repeat the process



(7) Show the probability distribution of model prediction in new window.

5.4 Use the Model with Highest Validation Accuracy to Run Inference Show the Predicted Distribution and Class Label. (6%) (出題：Hsiang)

- The probability distribution of model prediction using a histogram.



Y-axis: probability

X-axis: class name

5. Training a CIFAR10 Classifier Using VGG19 – Example Video

(出題：Hsiang)

- This is an example illustrating the objectives from 5.1 ~ 5.4.