電腦視覺與深度學習
(Computer Vision and Deep Learning)
Homework 1

TA:胡滋紜

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, 2023/11/09 (Thu.)

  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: cvdl2023 Password: RL2023cvdl
- 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

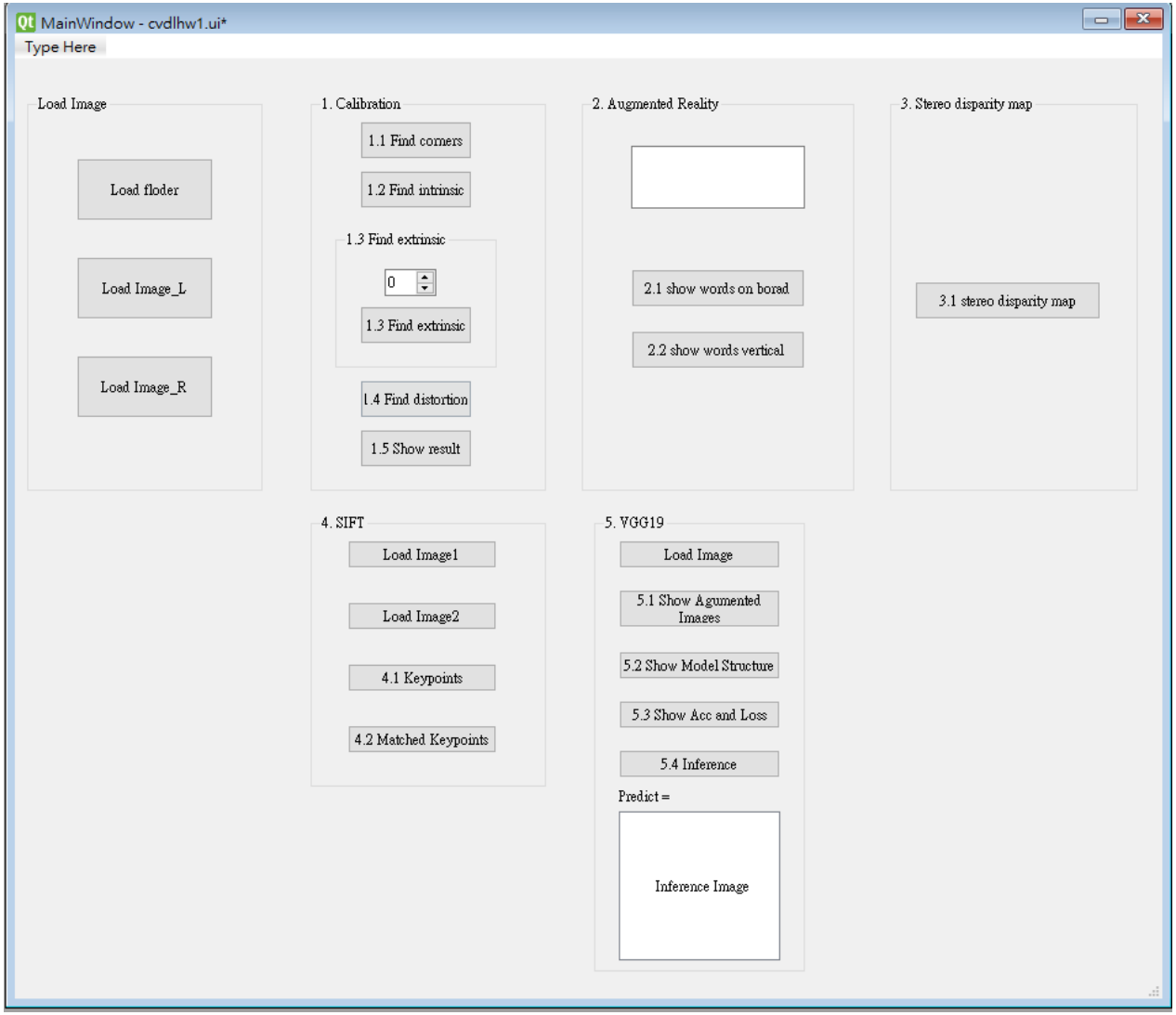
Assignment scoring (Total: 100%)

1. (20%) Camera Calibration (出題： Shan)
   - 1.1 (4%) Corner detection
   - 1.2 (4%) Find the intrinsic matrix
   - 1.3 (4%) Find the extrinsic matrix
   - 1.4 (4%) Find the distortion matrix
   - 1.5 (4%) Show the undistorted result
2. (20%) Augmented Reality (出題：Eric)
   - 2.1 (10%) Show words on board
   - 2.2 (10%) Show words vertically
3. (20%) Stereo Disparity Map (出題：Eric)
   - 3.1 (10%) Stereo Disparity Map
   - 3.2 (10%) Checking the Disparity Value
4. (20%) SIFT (出題：Chen)
   - 4.1 (10%) Keypoints
   - 4.2 (10%) Matched Keypoints
5. (20%) Training a CIFAR10 Classifier Using VGG19 with BN (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%)

* Don't fix your image path
(There is another dataset for demonstration)
Load image please use the following function to read the path.
QFileDialog.getOpenFileName

Assignment scoring (Total: 100%)

- Use one UI to present 5 questions.



1. (20%) Camera Calibration

(出題：shan)

1.1 (4%) Corner detection

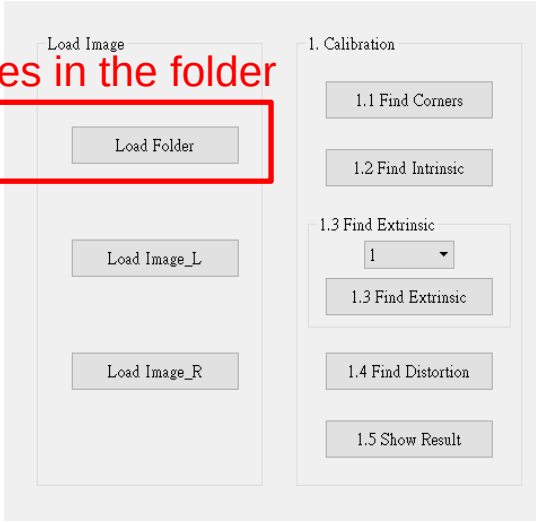
1.2 (4%) Find the intrinsic matrix

1.3 (4%) Find the extrinsic matrix

1.4 (4%) Find the distortion matrix

1.5 (4%) Show the undistorted result

Load all images in the folder



1.1 Corner Detection (4%)

- Given: 15 images, 1.bmp ~ 15.bmp
- Q1:

  1) Find and draw the corners on the chessboard for each image.

  cv2.findChessboardCorners(grayimg, (width, high), **None**)

  cv2.cornerSubPix(image, corners, **winSize**, **zeroZone**, **criteria**)

  winSize = (5, 5) **zeroZone** = (-1, -1), this parameter means to ignore.

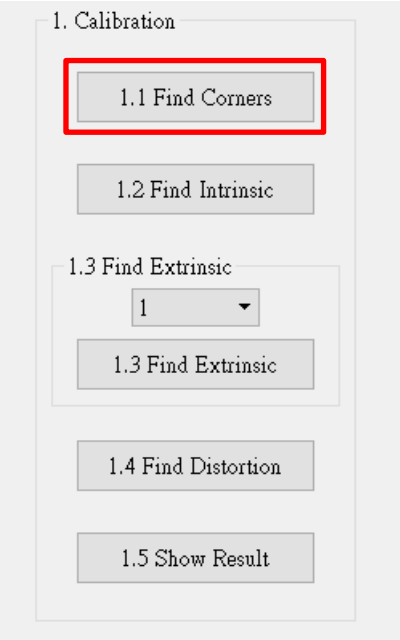
  criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

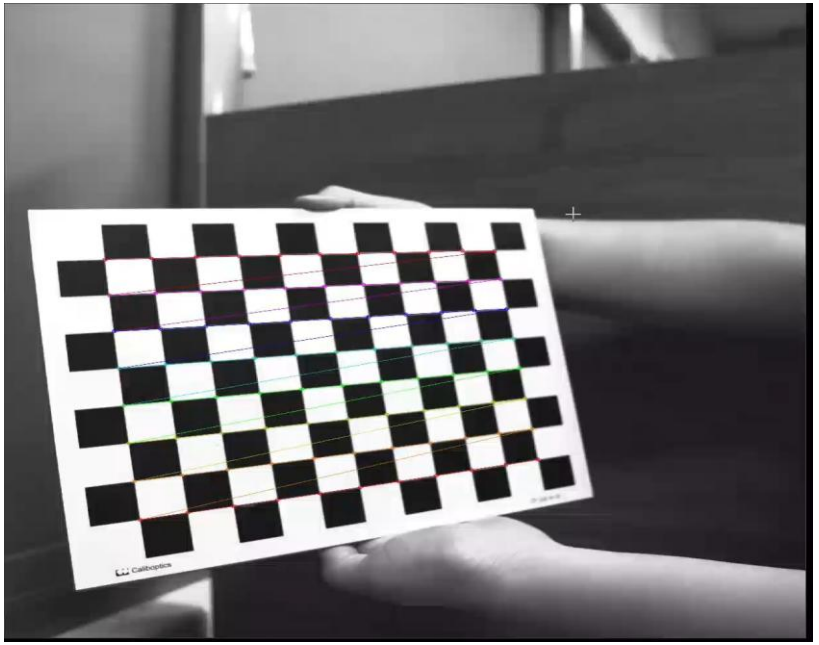
  2) Click button “1.1 Find Corners” to show each picture.

- Hint:

  OpenCV Textbook Chapter 11 (p. 398 ~ p. 399)

- Ex:





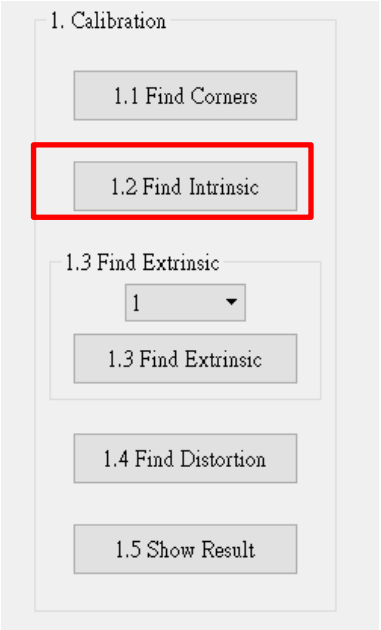
1.2 Find the Intrinsic Matrix (4%)

(出題： shan)

- Given: 15 images, 1.bmp ~ 15.bmp
- Q2:
  1) Find the intrinsic matrix: $\begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$

     cv2.calibrateCamera (objectPoints, imagePoints,(width, high) ,None, None)

  2) Click button “1.2 Find Intrinsic” and then show the result on the console window.



- Output format:

```
Intrinsic:
[[2.22370244e+03 0.00000000e+00 1.03021663e+03]
 [0.00000000e+00 2.22296836e+03 1.03752624e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```

(Just an example)

- Hint: OpenCV Textbook Chapter 11 (p.398 ~ p.400)

1.3 Find the Extrinsic Matrix (4%)

(出題： shan)

- Given: Intrinsic parameters, distortion coefficients, and the list of 15 images
- Q3:
  1) Find the extrinsic matrix of the chessboard for each of the 15 images, respectively:

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{bmatrix}$$

  2) Click button “1.3 Find Extrinsic” and then show the result on the console window.

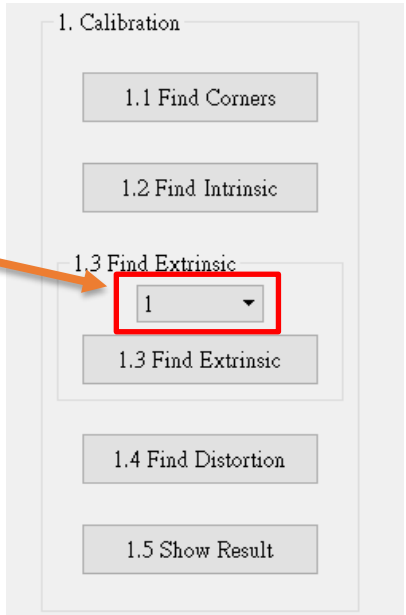
- Output format:

```
Extrinsic:
[[-0.8767247  -0.23001438  0.4224301   4.39838495]
 [ 0.19727469 -0.97293475 -0.12033563  0.68022105]
 [ 0.43867585 -0.02216645  0.89837194 16.22126   ]]
```

(Just an example)

- Hint: OpenCV Textbook Chapter 11, (p.370 ~ p.402)

(1) List of numbers: 1~15
(2) Select 1, then 1.bmp will be applied, and so on



1.4 Find the Distortion Matrix (4%)

(出題： shan)

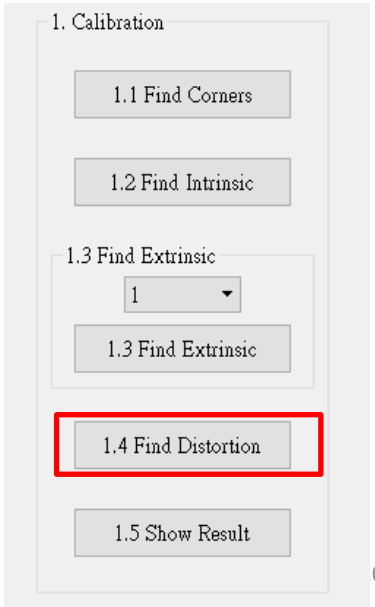
- Given: 15 images
- Q4:
  1) Find the distortion matrix: $[k_1, k_2, p_1, p_2, k_3]$
  2) Click button “1.4 Find Distortion” to show the result on the console window.

- Output format:

```
Distortion:
[[-0.11868112  0.02776881 -0.00092036  0.00047227  0.11793646]]
```

(Just an example)

- Hint:
  - Distortion coefficients can be obtained simultaneously with intrinsic.
  - OpenCV Textbook Chapter 11 (p.398 ~ p.400)

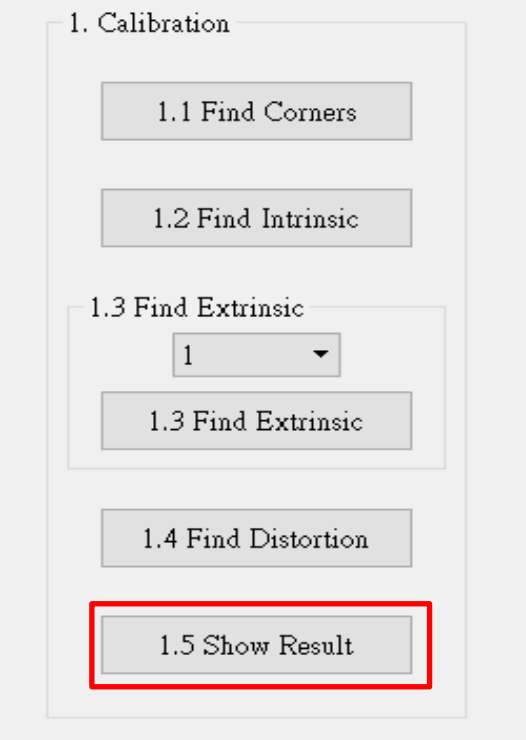


1.5 Show the Undistorted Result (4%)

(出題： shan)

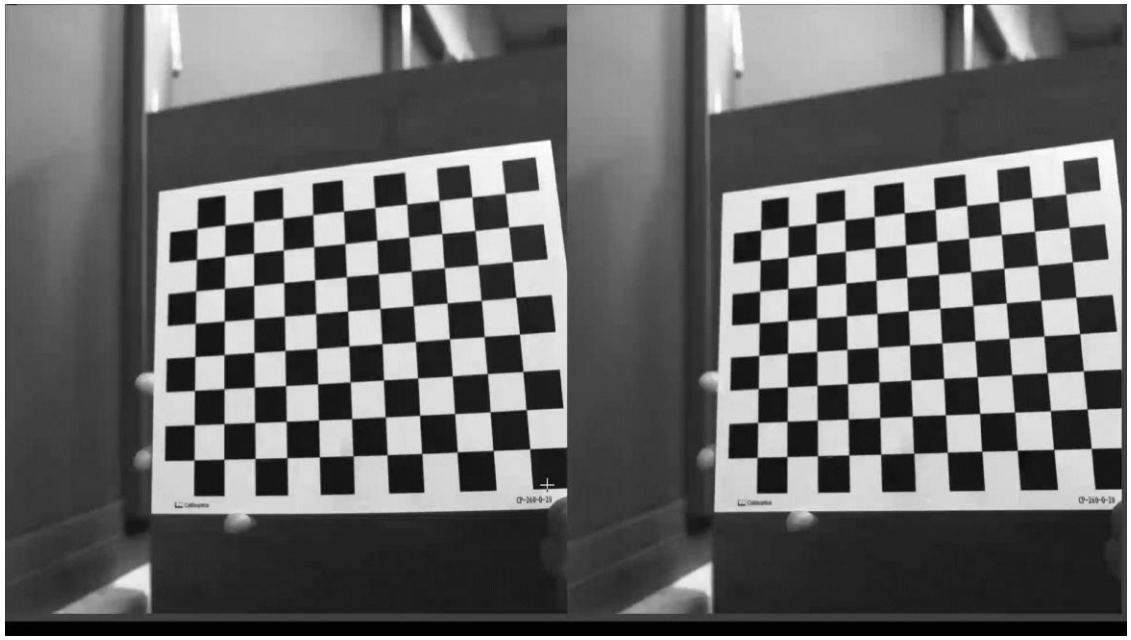
- Given: 15 images
- Q5:
  1) Undistort the chessboard images.
  2) Click button “1.5 Show Result” to show distorted and undistorted images

- Hint:
  - cv2.undistort(…) or cv2.initUndistortRectifyMap(…)
  - OpenCV Textbook Chapter 11 (p.398 ~ p.400)



Distorted image

Undistorted image

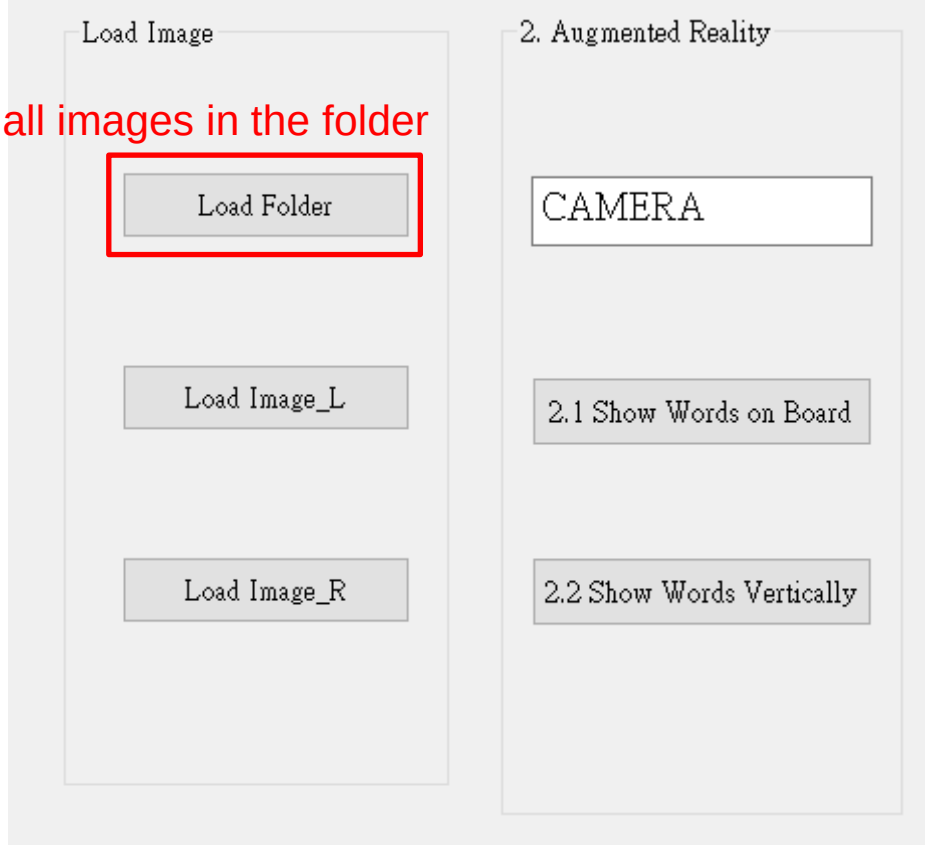


2. (20%) Augmented Reality

(出題：Eric)

2.1 (10%) Show words on board

2.2 (10%) Show words vertically



2. (20%) Augmented Reality

(出題： Eric)

- Given: 5 images (1~5.bmp), alphabet_lib_onboard.txt and alphabet_lib_vertical.txt.

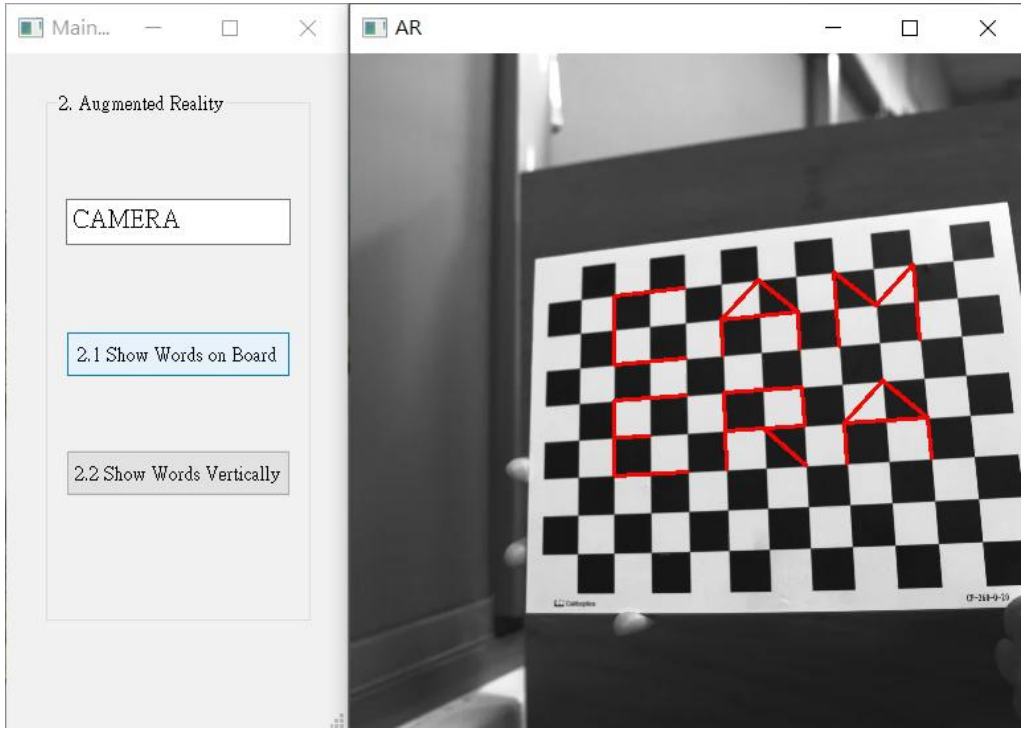
  Q1: **Show a Word (e.g. CAMERA) on chessboard**

  1) Calibrate 5 images to get intrinsic, distortion and extrinsic parameters.
  2) Input a word less than 6 char in English in textEdit box.
  3) Derive the shape of the word by using the provided library (alphabet_lib_onboard.txt).
  4) Click button "2.1 Show Words on Board" to show result, each picture for 1 second (total 5 images).

  Q2: **Show a Word (e.g. CAMERA) vertically on chessboard**

  1) Calibrate 5 images to get intrinsic, distortion and extrinsic parameters.
  2) Input a word less than 6 char in English in textEdit box (Accept the same word entered in as Q1).
  3) Derive the shape of the word by using the provided library (alphabet_lib_vertical.txt).
  4) Click button "2.2 Show Words Vertically" to show result, each picture for 1 second (total 5 images).

- Result Video:



- Hint: OpenCV Textbook Chapter 11 (P.387~395)
  cv2.calibrateCamera()
  OpenCV Textbook Chapter 11 (P.405~412)
  cv2.projectPoints()

2. (20%) Augmented Reality

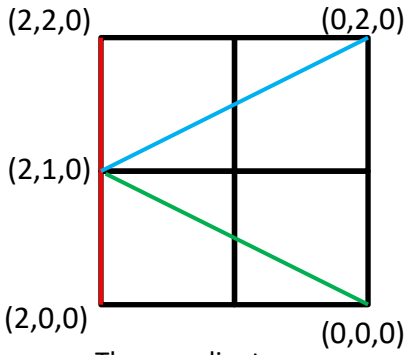
(出題： Eric)

- Guides and Requirements:

1) How to use the library: (alphabet_lib_onboard.txt, alphabet_lib_vertical.txt)
   - Use OpenCV function to read and derive the array or matrix of the char
     Here take 'K' in 'alphabet_lib_onboard.txt' for example

     e.g. (Python):
     fs = cv2.FileStorage('alphabet_lib_onboard.txt', cv2.FILE_STORAGE_READ)
     ch = fs.getNode('K').mat() ➔ get the lines of 'K'

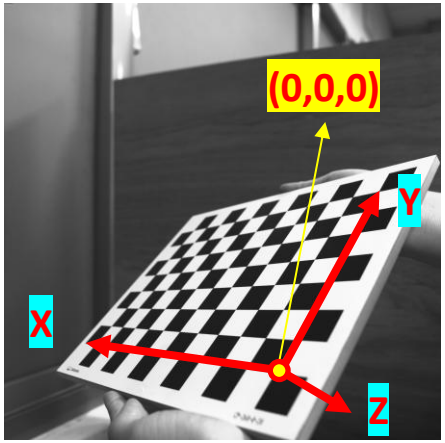
     ch = [[[2, 2, 0], [2, 0, 0]],
     [[0, 2, 0], [2, 1, 0]],
     [[2, 1, 0], [0, 0, 0]]]



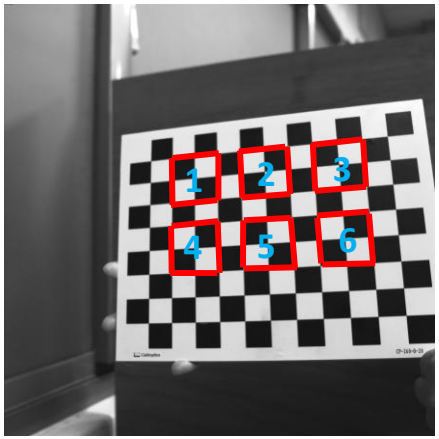
The coordinates are in Word Frame

   - 'K' consist of 3 lines, so the 'ch array' consists 3 pairs of 3D coordinates in Word Frame representing two ends of the line shown in the upper right image.

2) Chessboard Coordinates
   - The chessboard x, y, z axis and (0,0,0) coordinate are shown in the bottom left image
   - Each Char should be place in the order and position shown in the bottom right image



Chessboard Frame



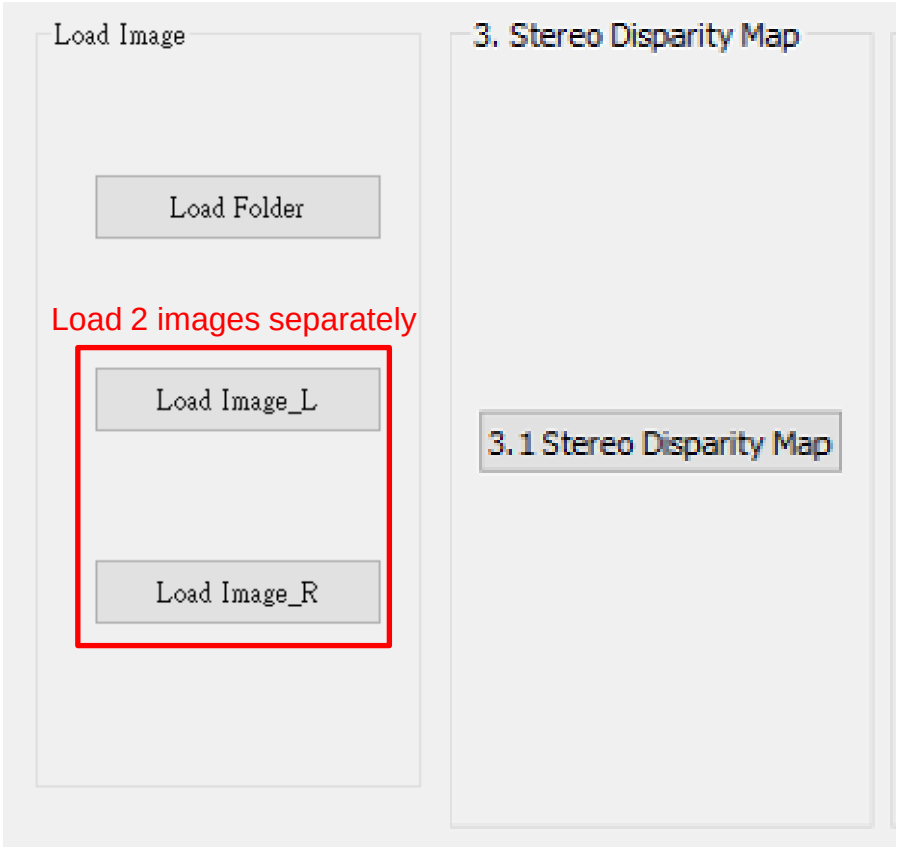
Position and Order

3. (20%) Stereo Disparity Map

(出題：Eric)

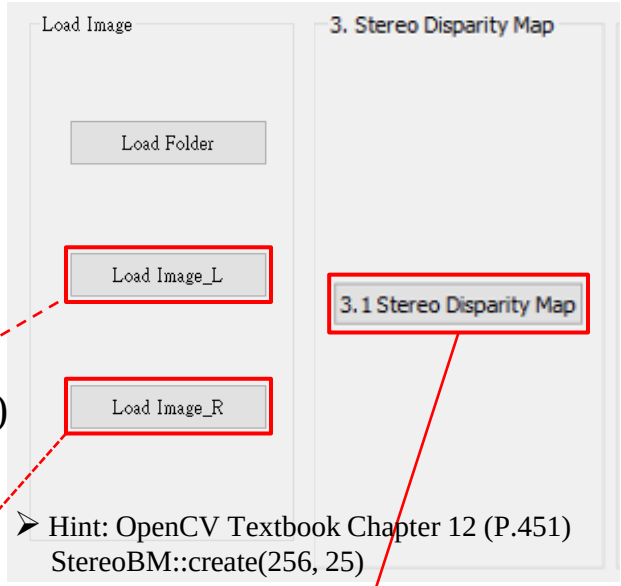
3.1 (10%) Stereo Disparity Map

3.2 (10%) Checking the Disparity Value



3.1 (10%) Stereo Disparity Map

(出題：Eric)



➢ Given: a pair of images, imL.png and imR.png (have been rectified).

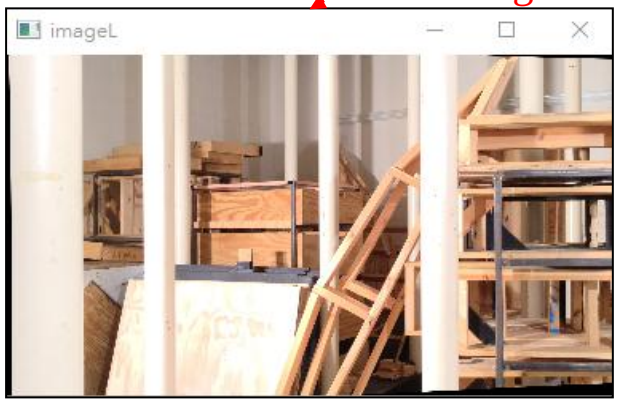
Q: **Find the disparity map/image based on Left and Right stereo images**

1) Load imL.png (click "Load Image_L").
2) Load imR.png (click "Load Image_R").
3) Click button "3.1 Stereo Disparity Map" to show result.

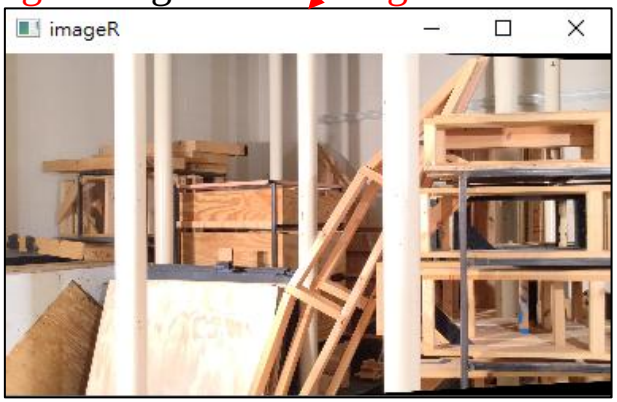
➢ Hint:

- Use OpenCV StereoBM class to build StereoBM objects.
- stereo = cv2.StereoBM_create(**numDisparities**=256, **blockSize**=25)
- The above parameters can be freely changed according to the following rules.
- **numDisparities**: The disparity search range. Must be positive and divisible by 16. In addition, map disparity range to gray value range 0~255 for the purpose of visualization.
- **blockSize**: The linear size of blocks compared by the algorithm, Must be odd and within the range [5, 50]. Larger block size implies smoother but less accurate disparity map. Smaller block size gives finer disparity details, yet increase the likelihood of algorithmic misalignment.
- If the left image is the reference image (the one used to cal. depth info for each pixel of that Img), then the search direction at right image will go from the right to left direction.

➢ Hint: OpenCV Textbook Chapter 12 (P.451)
StereoBM::create(256, 25)



imL.png
Left Image (Reference Image)



imR.png
Right Image

Result

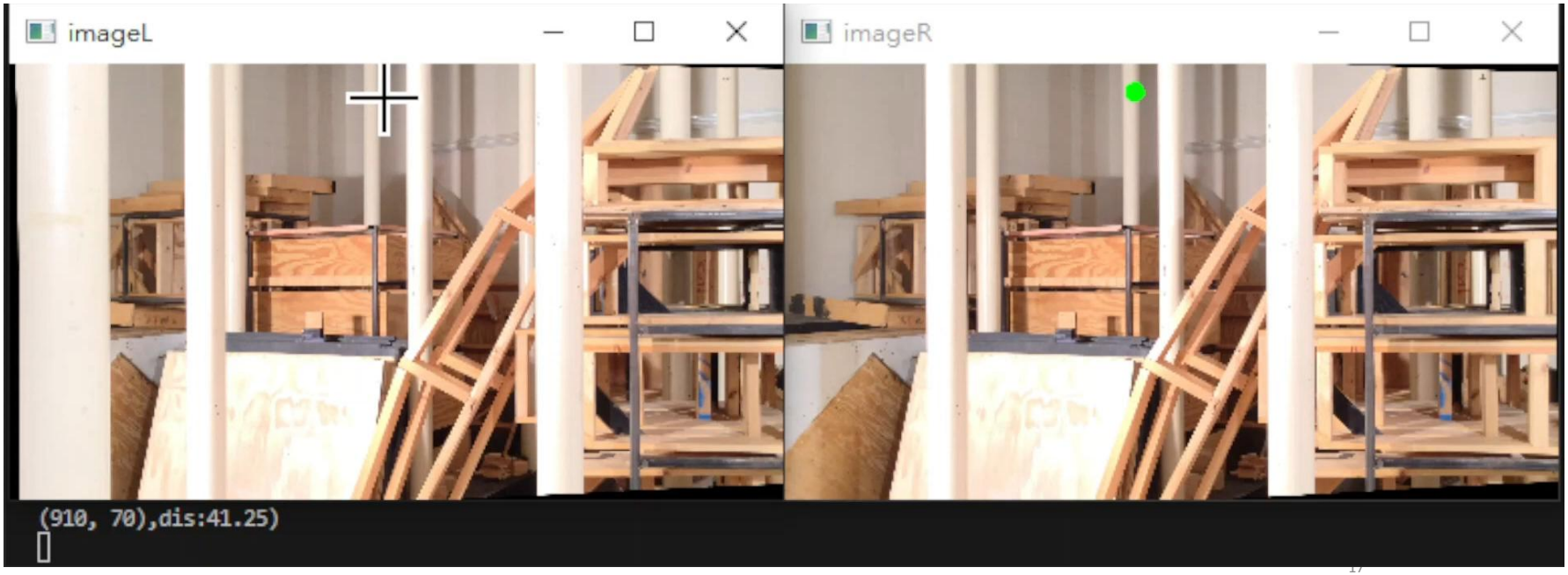
3.2 (10%) Checking the Disparity Value

(出題：Eric)

- Given: a pair of images, imL.png and imR.png and disparity map from 3.1
  Q: **Click at left image and draw the corresponding dot at right image.**
  1) Click at left image and draw the dot on the right image at accurate position.
  2) User should be allowed to repeat the action 1).

- Hint: Click at gray position at disparity map result from Q3.1, ignore the position with 0 disparity (e.g. Failure case).

- Result Video:

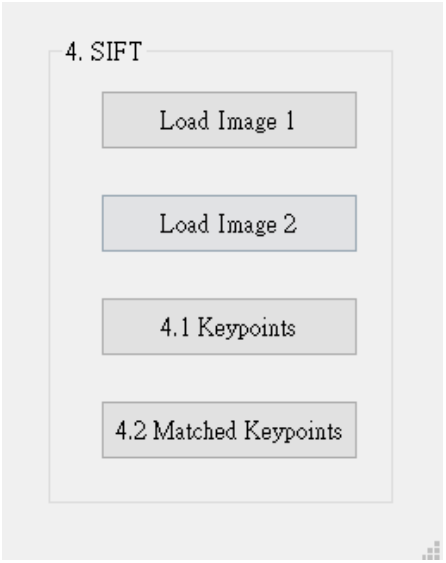


4. (20%) SIFT

(出題：Chen)

4.1 (10%) Keypoints

4.2 (10%) Matched Keypoints

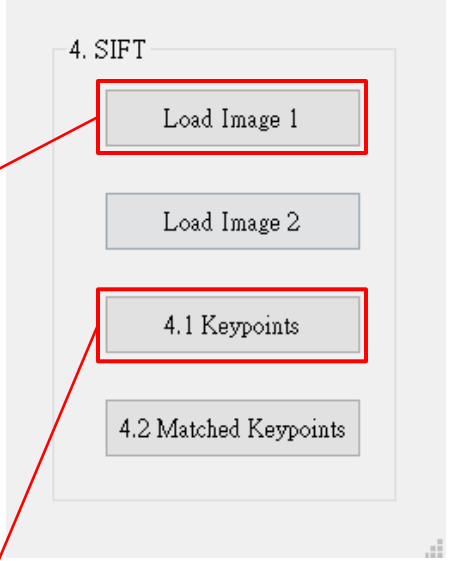


4.1 (10%) Keypoints

(出題：Chen)

- Click button “4.1 Keypoints” to show:
  1) Load Left.jpg (click “Load Image 1”)
  2) Based on SIFT algorithm, find key points on Left.jpg.
     - Hint: Use OpenCV SIFT detector to detect keypoints and descriptors.
     - sift = cv.SIFT_create() # Create a SIFT detector
     - sift.detectAndCompute(image,None)
  3) Then draw the keypoints of Left.jpg as I_1.
     - Hint: cv.drawKeypoints(gray,kp,None,color=(0,255,0))

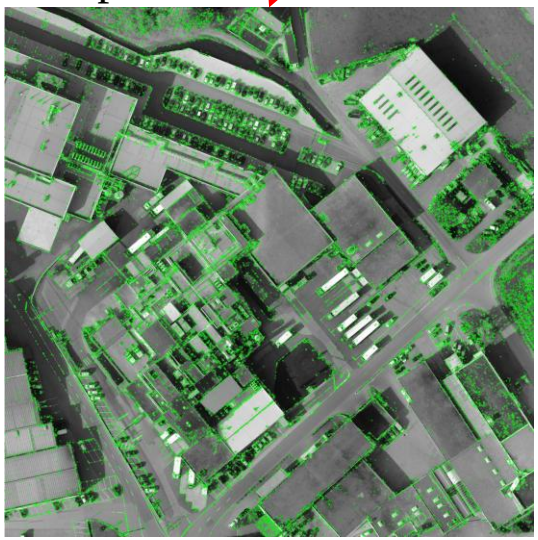
Please show image I_1



Input:

Left.jpg

Output:



I_1

Reference : OpenCV Textbook SIFT, p321, 464, 524

4.2 (10%) Matched Keypoints

(出題：Chen)

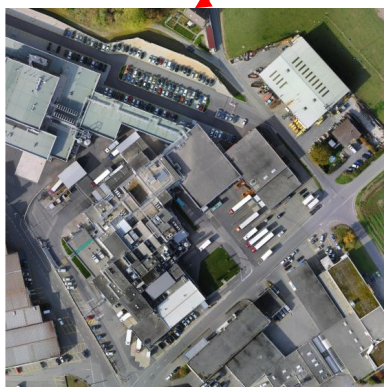
- Click button “4.2 Matched Keypoints”,
  1) Load Image 1 (Left.jpg) and Image 2 (Right.jpg)
  2) Based on SIFT algorithm, find the keypoints and descriptors at Image 1 and Image 2 (same as question 1)
  3) Find match keypoint of two images
     - Hint: Use BFMatcher.knnMatch(descriptors 1, descriptors 2,k=2) to locate the matched keypoints.
  4) Extract Good Matches from 3) result:
     - Hint:

```
for m,n in matches:
    if m.distance < 0.75*n.distance:
        good.append([m])
```

  5) Draw the matched feature points between two image
     - Hint:. Use “cv2.drawMatchesKnn()” to draw the matches,

```
cv.drawMatchesKnn(img1,kp1,img2,kp2,mat,good,flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
```

Input:

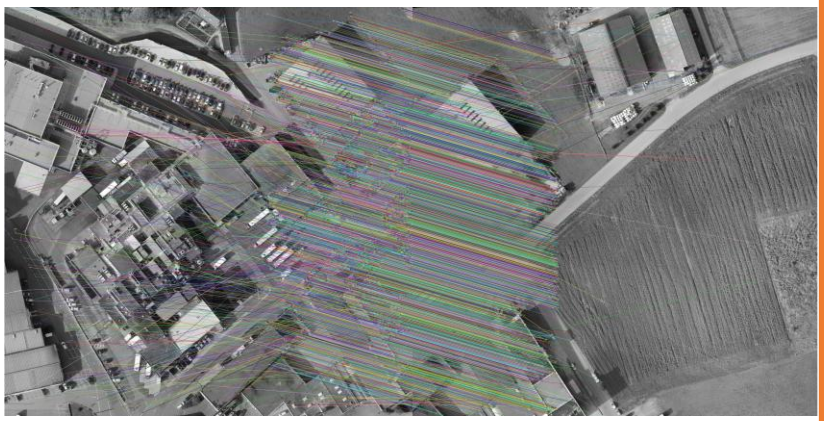


Left.jpg

Right.jpg

Output:

Please show image I_2



I_2

Reference: OpenCV Textbook SIFT, p321, 464, 524

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%)

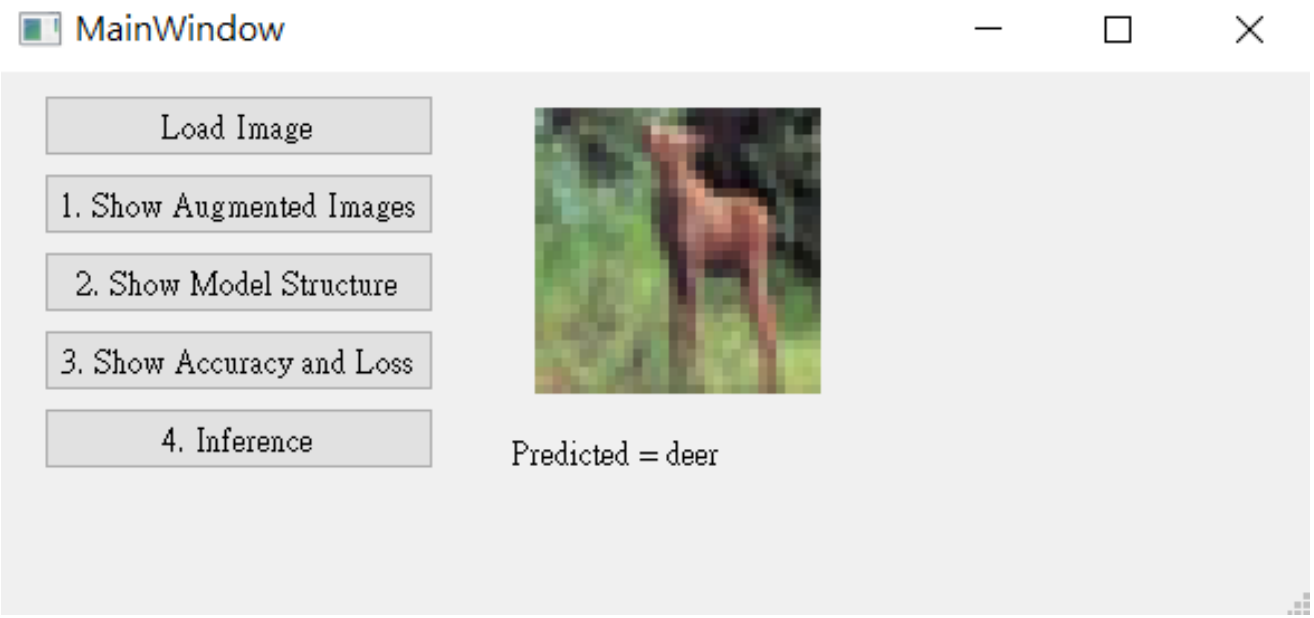


Figure: GUI example

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.)

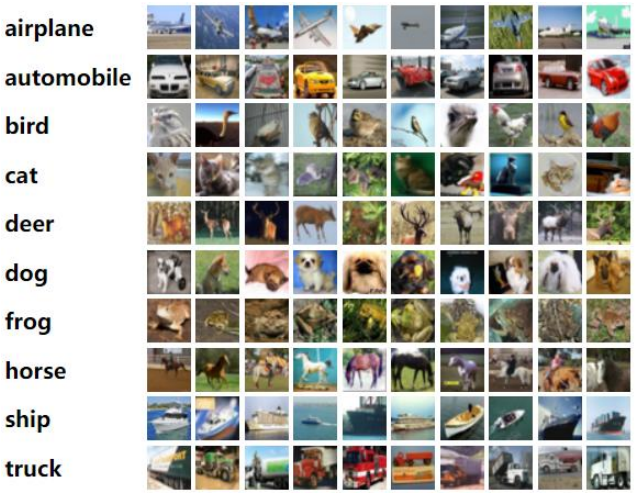


*Figure*1: CIFAR10

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.

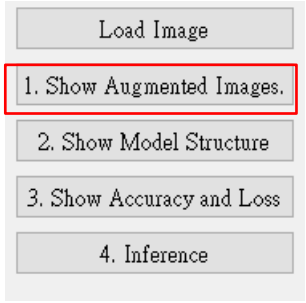
(2) Apply at least 3 different type of data augmentation (tutorial).

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

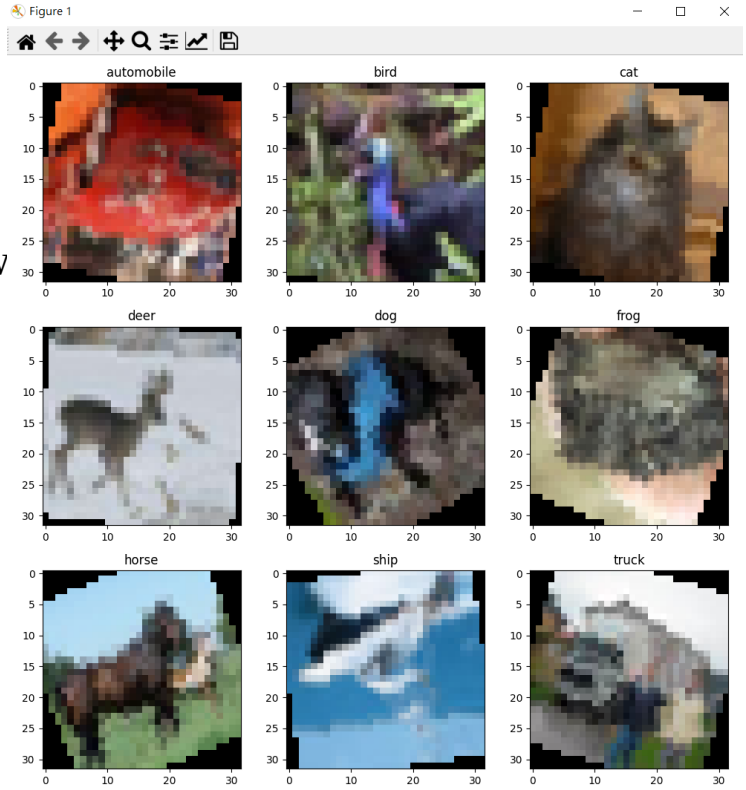


Figure1: 9 Augmented images

Notice: this is an example, the images might differ

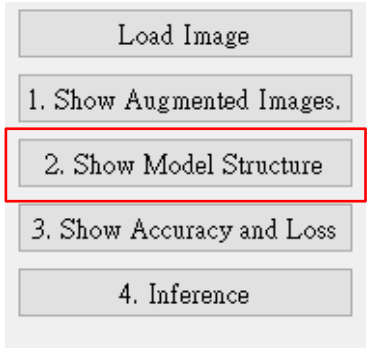
5.2 Show the Structure of VGG19 with BN (3%)

(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.



Layer (type) | Feature map shape (Batch, Channels, Height, Width) | 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

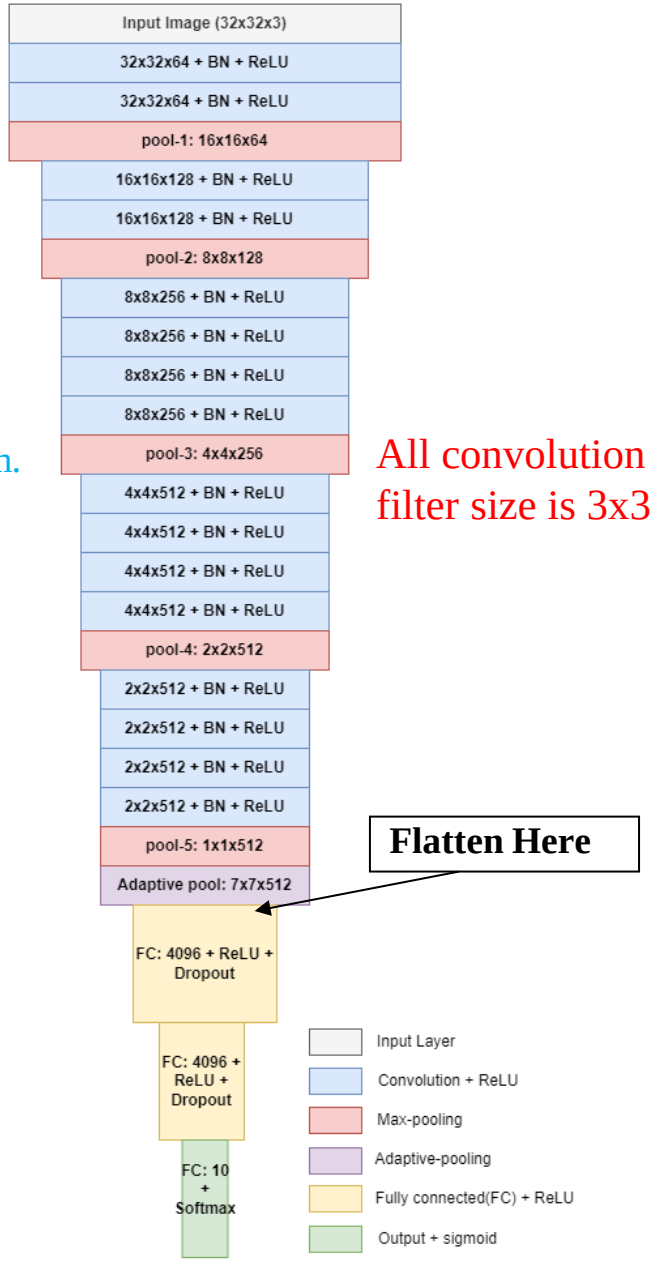


Figure: VGG19 with BN model structure

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.

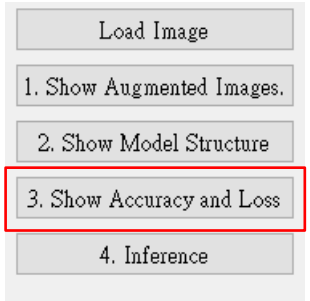
(6) Save the figure

2) When the demo:

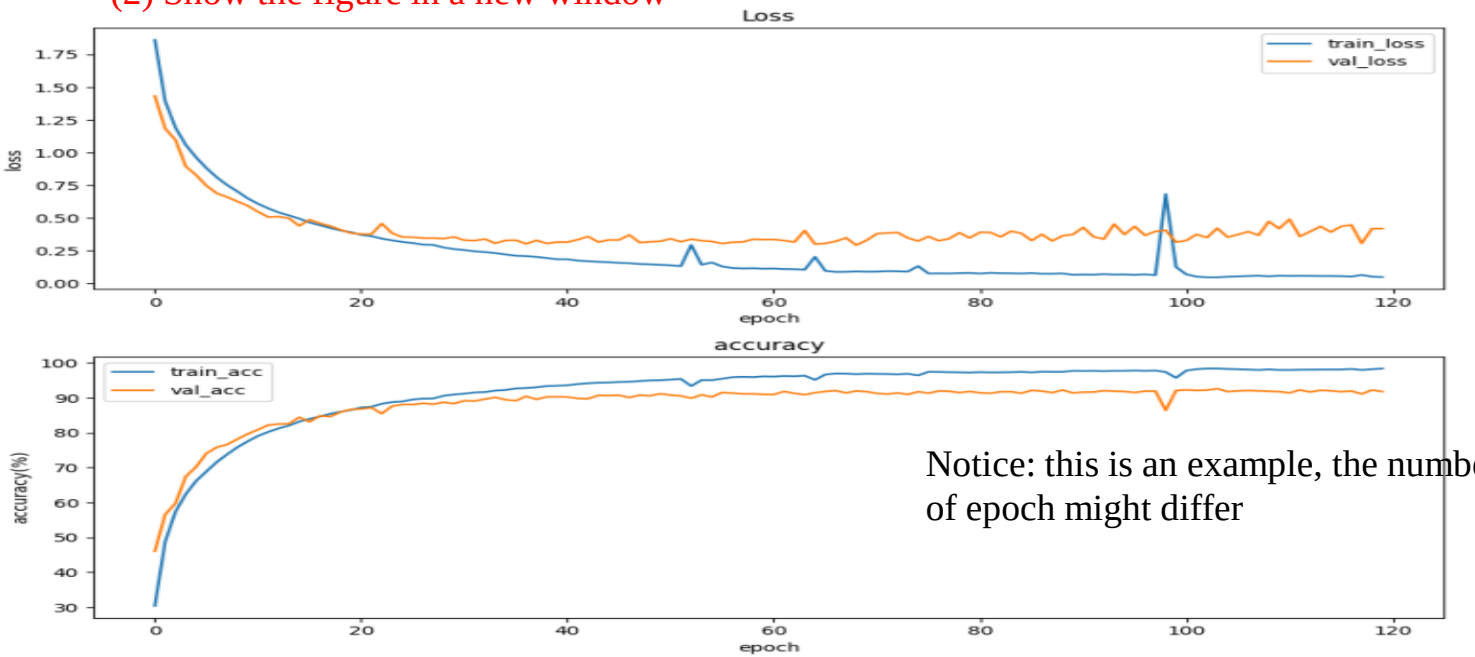
(1) Click the button "3. Show Accuracy and Loss"

(2) Show the saved figure of Training/Validating loss and accuracy in a new window

(1) Click the button.



(2) Show the figure in a new window

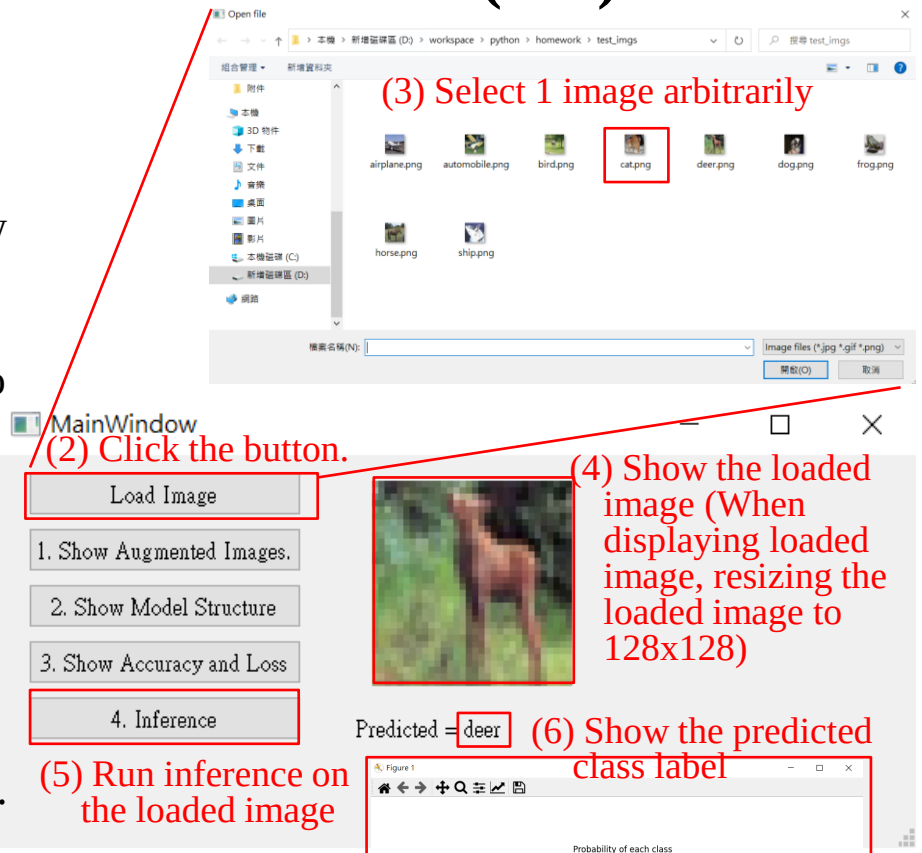


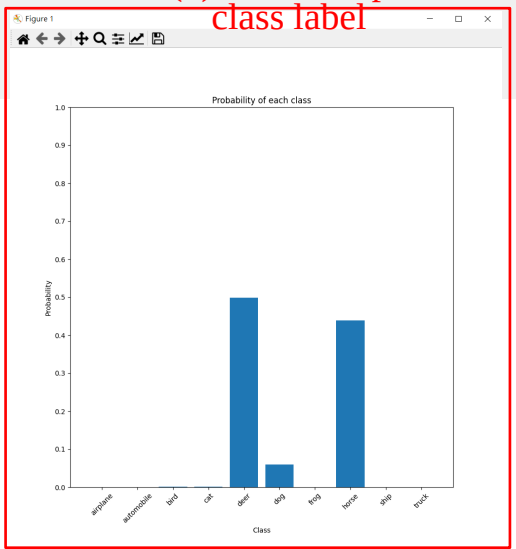
Notice: this is an example, the number of epoch might differ

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

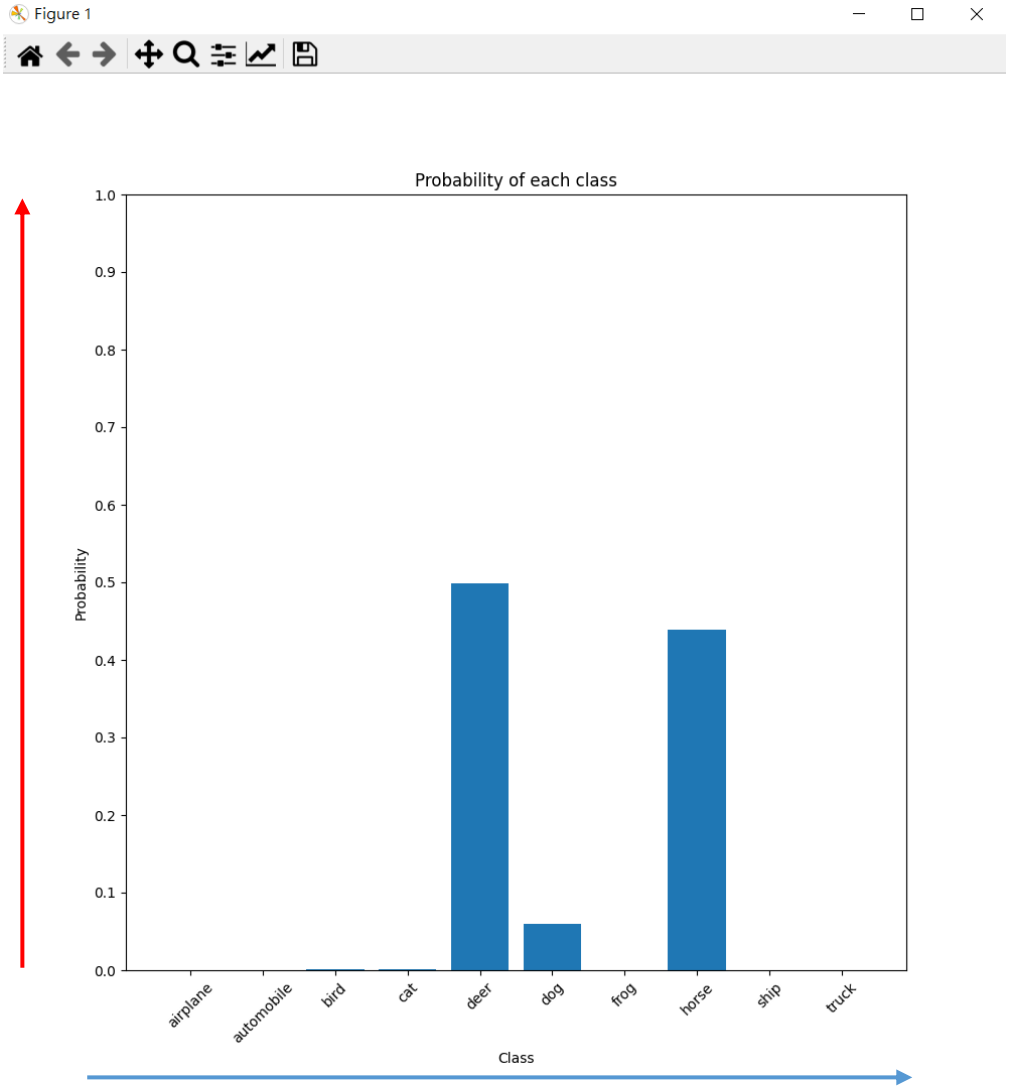




(7) Show the probability distribution of model prediction in new window.

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.