電腦視覺與深度學習
(Computer Vision and Deep Learning)
Homework 2

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, 2023/12/21 (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/Hw2
  - ➢ User ID: cvdl2023 Password: RL2023cvdl
- Format
  - ➢ Filename: Hw2_StudentID_Name_Version.rar
    - Ex: Hw2_F71234567_林小明_V1.rar
    - If you want to update your file, you should update your version to be V2,
    - Ex: Hw2_F71234567_林小明_V2.rar
  - ➢ Content: Project folder *(Excluding the pictures)

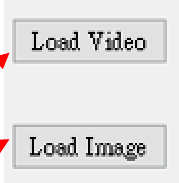
    *Note: Remove your “Debug” folder to reduce file size.

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%) Background Subtraction (出題： Chen)
2. (20%) Optical Flow (出題： Jimmy)
   - 2.1 (10%) Preprocessing
   - 2.2 (10%) Video tracking
3. (20%) PCA - Dimension Reduction (出題：Zhong)
4. (20%) Training a MNIST Classifier Using VGG19 with BN (出題：Shang)
   - 4.1 (6%) Load Model and Show Model Structure.
   - 4.2 (6%) Show Training/Validating Accuracy and Loss.
   - 4.3 (8%) Use the Model with Highest Validation Accuracy to Run Inference, Show the Predicted Distribution and Class Label.
5. (20%) Train a Cat-Dog Classifier Using ResNet50 (出題：Shan)
   - 5.1 (4%) Load the dataset and resize images
   - 5.2 (4%) Plot class distribution of training dataset
   - 5.3 (4%) Show the structure of ResNet50 model
   - 5.4 (4%) Improve ResNet50 with Random-Erasing and Compare the accuracies of 2 ResNet50 models on validation dataset
   - 5.5 (4%) Use the trained model to run inference and show the predicted class label

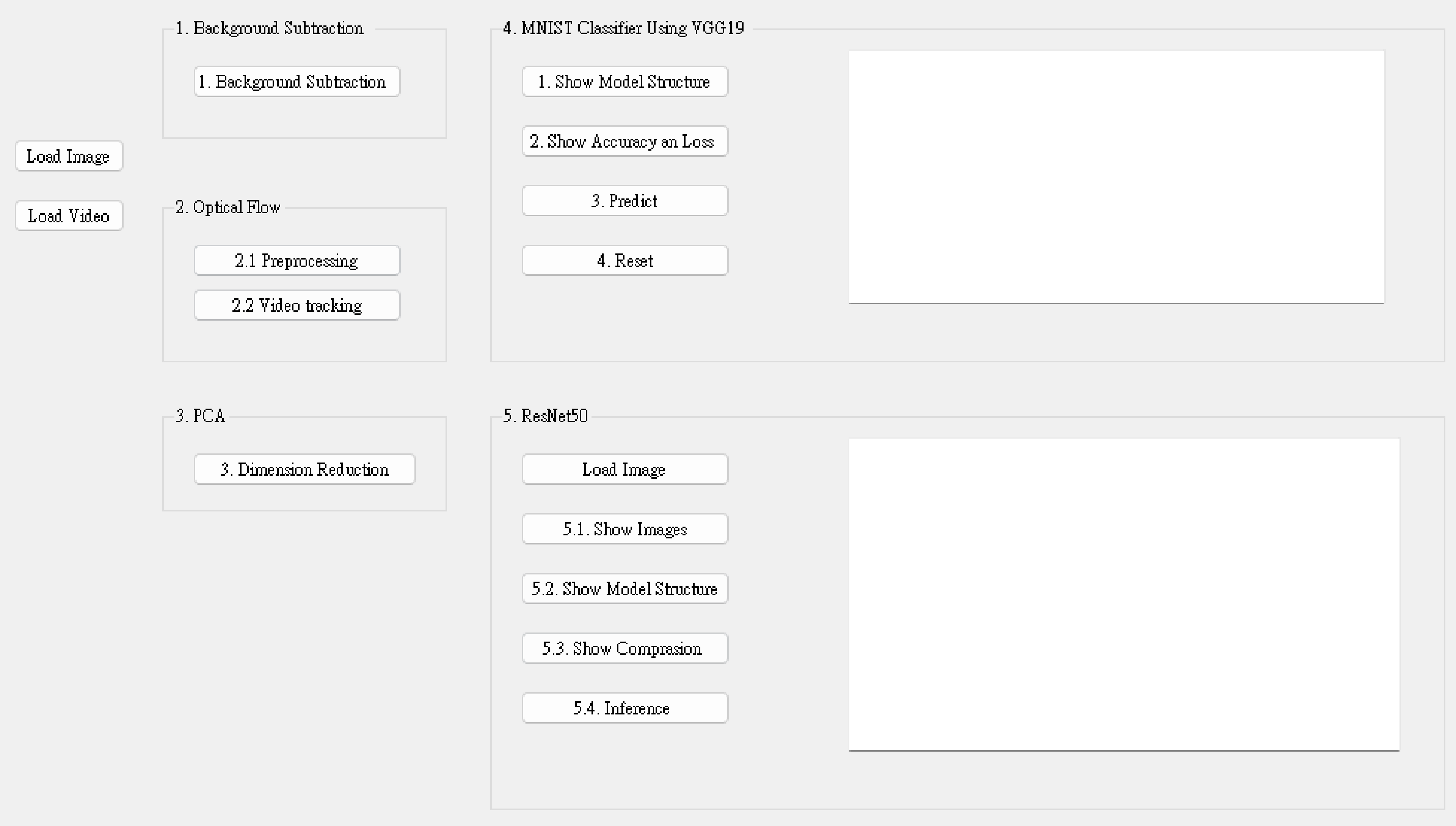


* Don't fix your image and video path
(There is another dataset for demonstration)

Load image and video please use the following function to read the path.
QFileDialog.getOpenFileName

Assignment scoring (Total: 100%)

- Use one UI to present 5 questions.



1. Background Subtraction (20%)

(出題：Chen)

- Given: a traffic video : “traffic.mp4”
- Q: Please remove background and show the result (Result should be like the Demo below)

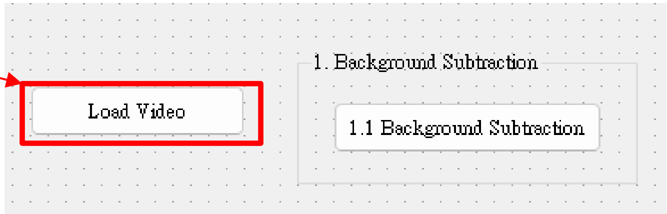
1) Load Video from File Dialog
2) Create subtractor using cv2.createBackgroundSubtractorKNN(history,dist2Threshold,detectShadows=True)

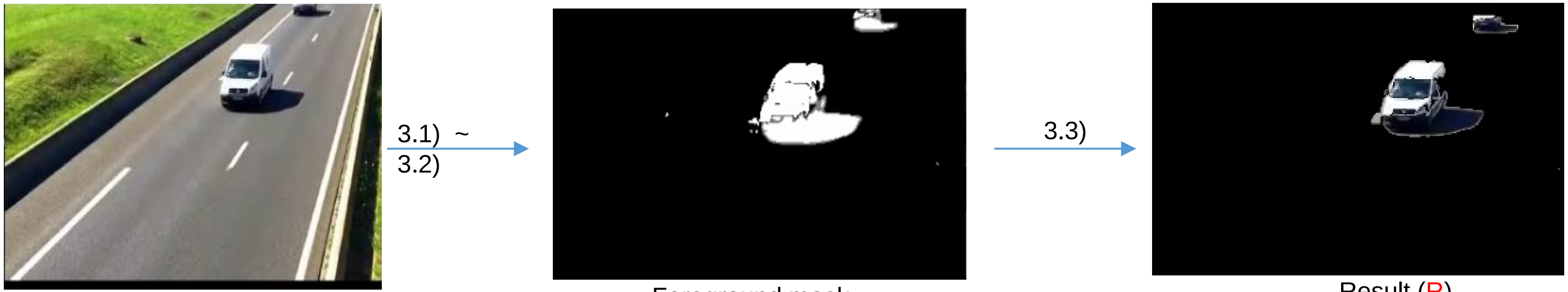
For each frame in video:

3.1) Blur frame using cv2.GaussianBlur(frame, (5, 5), 0)

3.2) Get background mask (M) by subtractor.apply

3.3) Generate Frame (R) with only moving object by cv2.bitwise_and





rgb frame

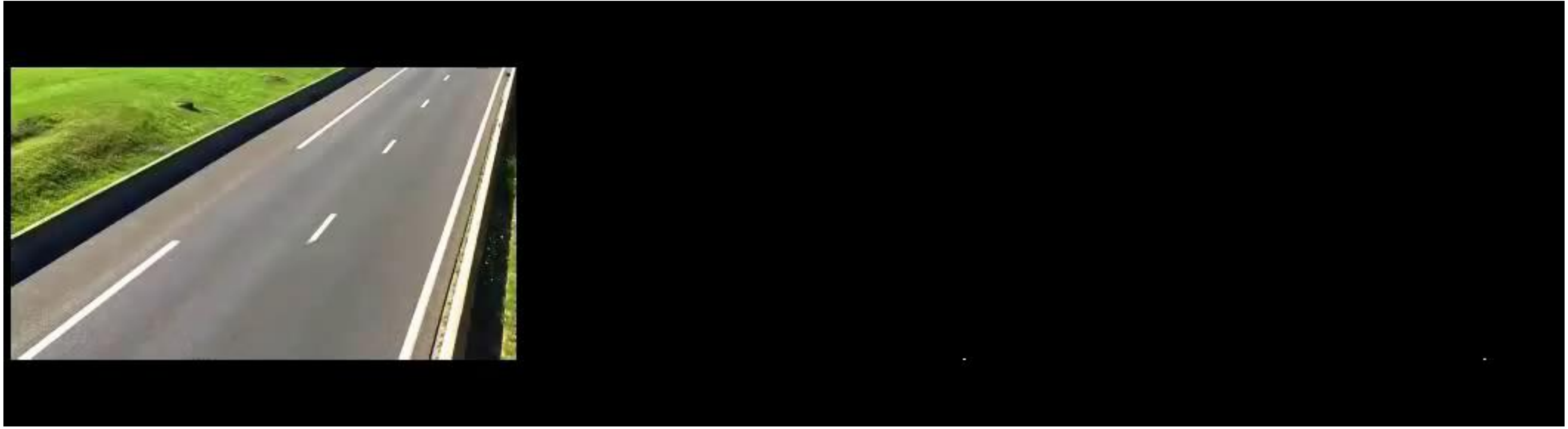
Foreground mask (M)

Result (R)

Demo (show result when frame number >4)

(M)

(R)



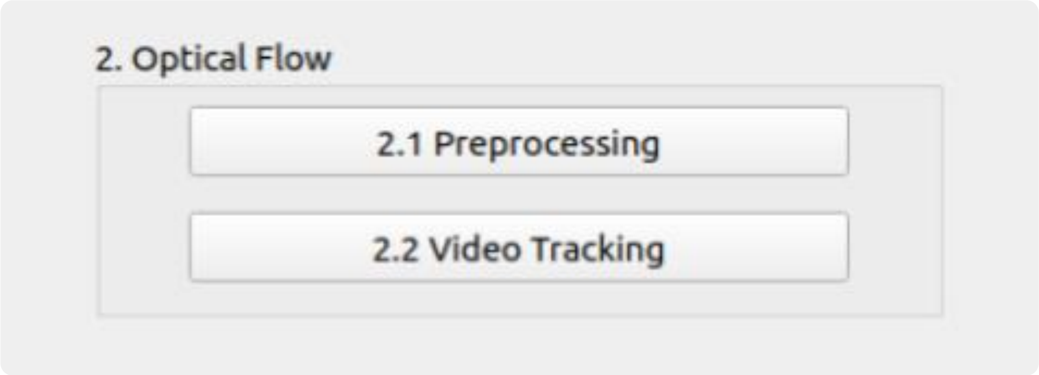
2. (20%) Optical Flow

(出題：Jimmy)

2.1 (10%) Preprocessing

2.2 (10%) Video tracking

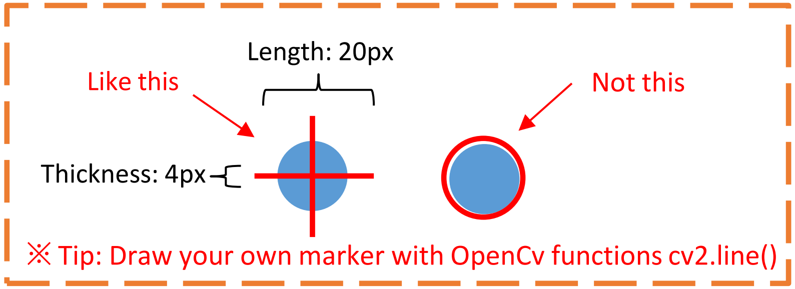
UI demo:



2.1 Preprocessing (10%)

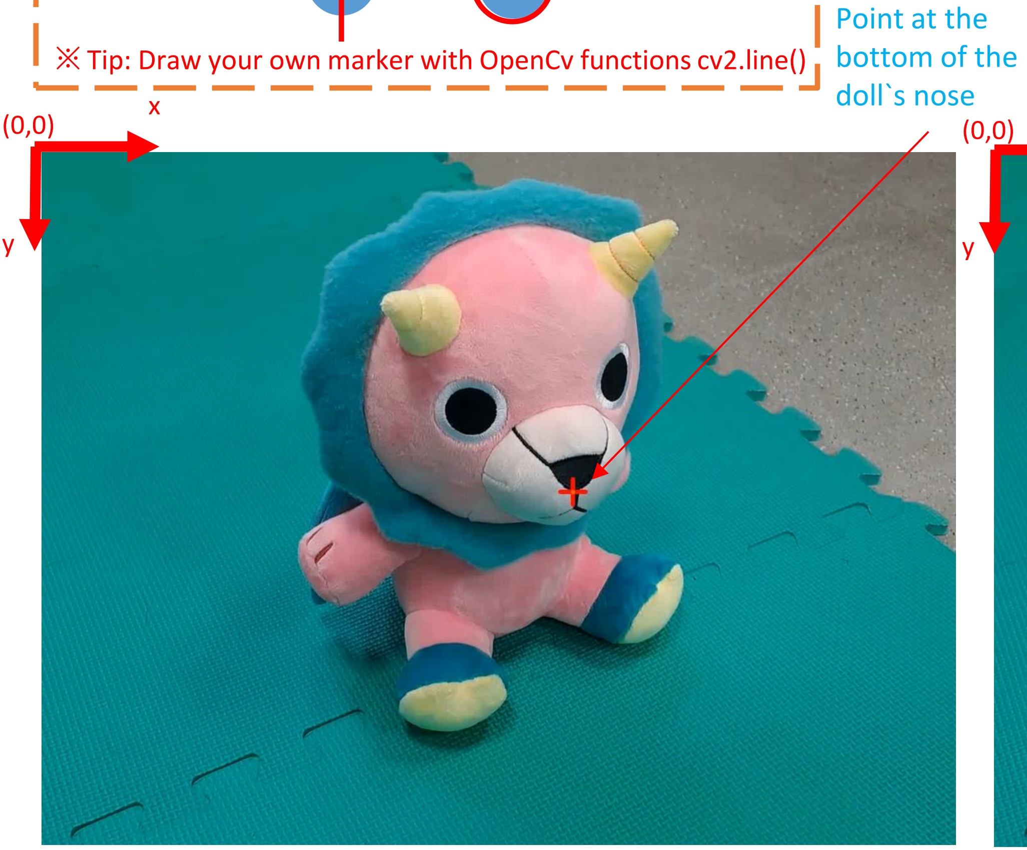
(出題：Jimmy)

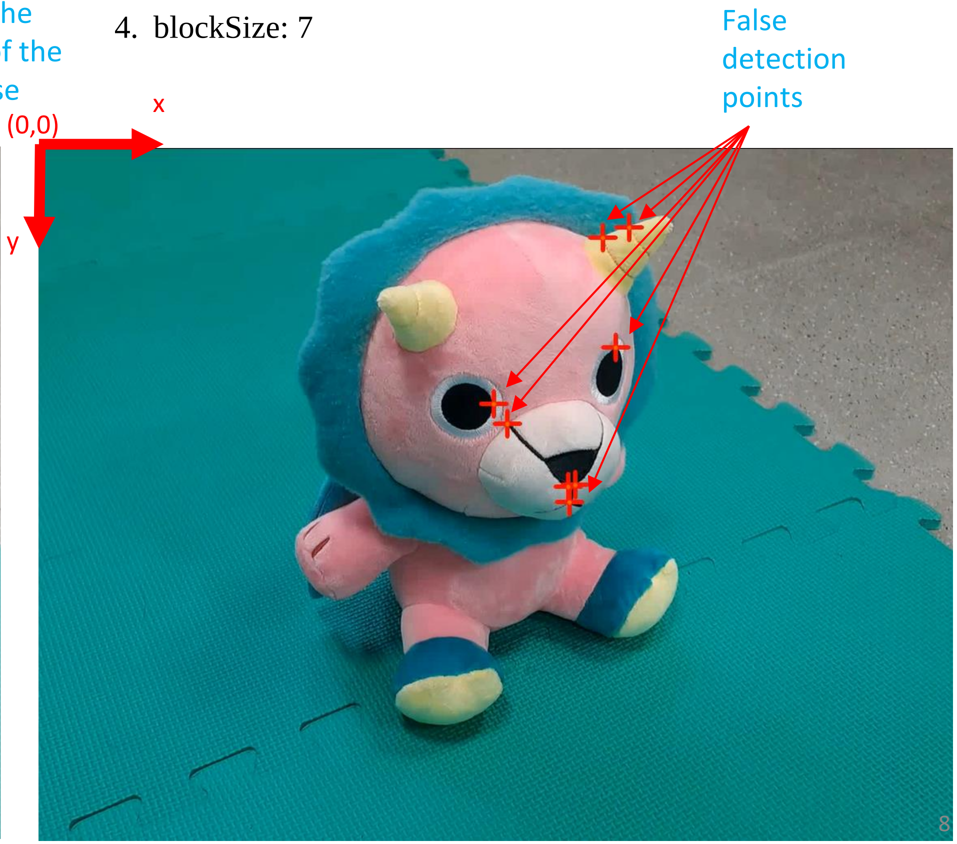
- ❑ Given a video: “optical_flow.mp4”
- ❑ Q: Click the button 2.1 to detect the point at the bottom of the doll`s nose in 1st Frame using cv2.goodFeaturesToTrack and show the point with a red cross mark using cv2.line



※ Tip: Draw your own marker with OpenCv functions cv2.line()

- Parameters in goodFeaturesToTrack should be adjusted:
  1. maxCorners: 1
  2. qualityLevel : 0.3
     (quality = min(λ_1, λ_2), λ: eigenvalue from the Hessian matrix)
  3. minDistance: 7
  4. blockSize: 7



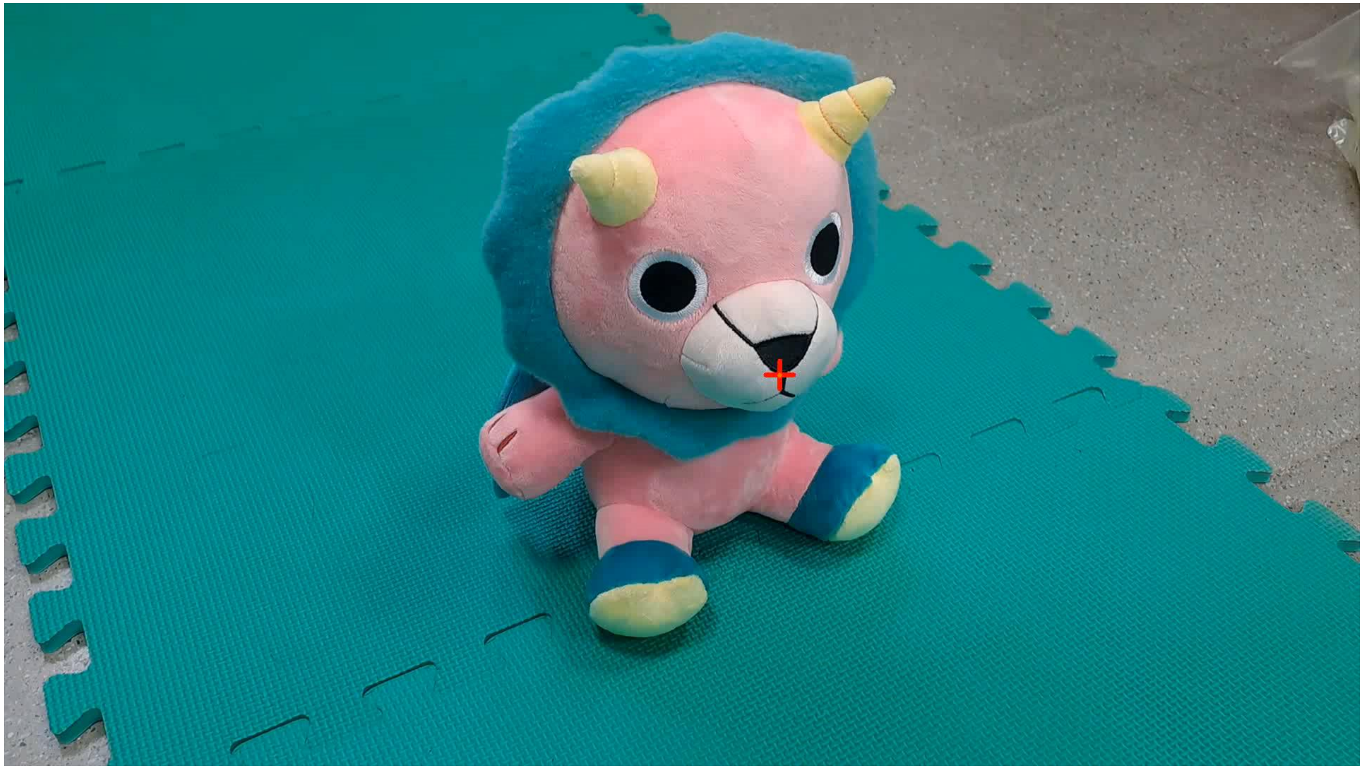


2.2 Video tracking (10%)

(出題：Jimmy)

- ❑ Q: Click button 2.2 to:
  1) (5%) Track the point (detected in 2.1) on the whole video using OpenCV function cv2.calcOpticalFlowPyrLK.
  2) (5%) Display the trajectory of the tracking point throughout the video using cv2.line. Pick a highly visible color. Ex: Yellow (0,100,255)

Demo videos:



- ❑ Tool site:
  1. Load video
  2. Optical flow
  3. Optical flow Tutorial

3. (20%) PCA – Dimension Reduction

(出題：Zhong)

❑ Given: A RGB image “logo.jpg”

❑ Q: Using PCA (Principal components analysis) to do dimension reduction on given image, find the minimum components that reconstruction error less or equal to 3.0

1) Convert RGB image to gray scale image, image shape will be (w,h).
2) Normalize gray scale image from [0,255] to [0,1]
3) Use PCA to do dimension reduction from min(w,h) to ***n***, then reconstruct the image.
4) Use MSE to compute reconstruction error, and find minimum ***n*** that error value less or equal to 3.0. Print out the ***n*** value. (10%)
5) Plot the gray scale image and the reconstruction image with ***n*** components. (10%)

❑ Hint: Use PCA from python library: sklearn.decomposition

Mean Square Error (MSE) Formula:

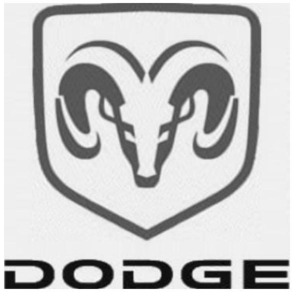
$$\frac{1}{n_{pixels}} * \sum_{i=1}^{n_{pixels}} (\vec{x_i} - \vec{y_i})^2 \text{ , } where$$

$\vec{x_i}$ *is the gray value of pixel in original image (gray scale image)*

$\vec{y_i}$ *is the gray value of pixel in reconstruction image*

logo.jpg

Gray scale image



Reconstruction image (i.e. n=50)

4. Training a MNIST Classifier Using VGG19 with BN (20%)

(出題：Shang)

4.1 Load Model and Show Model Structure. (6%)

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

4.3 Use the Model with Highest Validation Accuracy to Run Inference, Show the Predicted Distribution and Class Label. (8%)

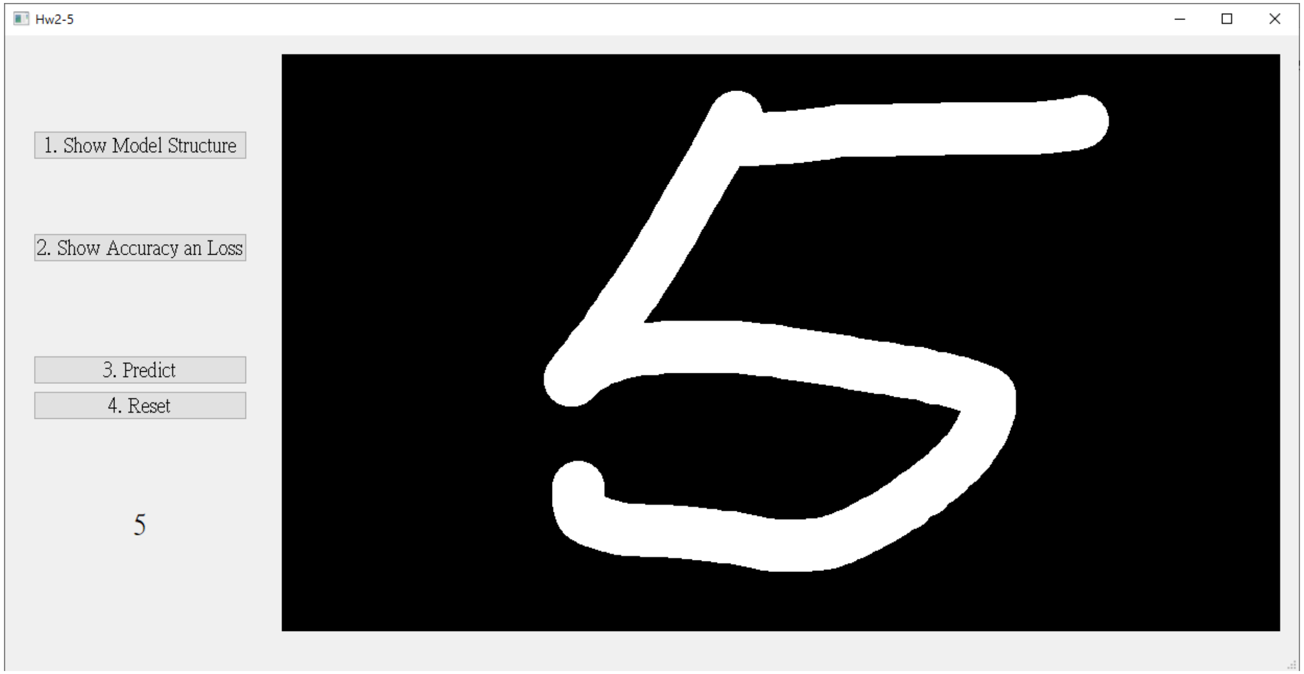


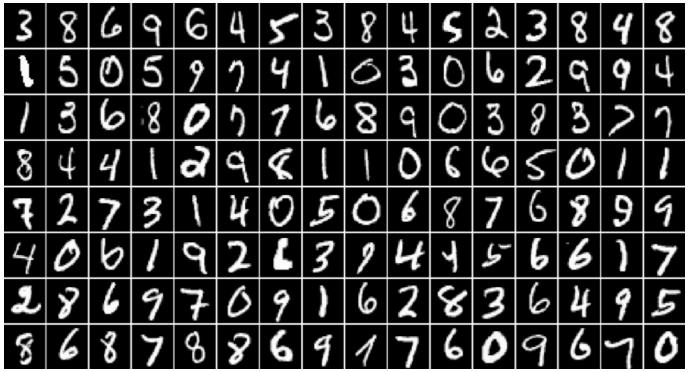
Figure: GUI example

4.0 Training a MNIST Classifier Using VGG19 with BN (20%)

(出題：Shang)

- Requirement

1) Train VGG19 model with batch normalization (BN) using PyTorch.
2) Download dataset using torchvision.datasets.MNIST() (tutorial)
   - Training data: 60000 images
   - Validation data: 10000 images
   - Resize image to (32, 32)



3) Parameters
   - At least 30 epochs.
   - Cross entropy loss
   - Adam optimizer
4) Record training/validation loss and accuracy in .jpg or .png format.
5) In the submitted file, you need to include
   - Weight file for VGG19 with BN in .pth format. (File size is approximately 540MB)
   - Figure of training/validating loss and accuracy in .jpg or .png format.
   - Code for your GUI program
   - Code for model training.
6) Please do not include image data in the submitted file.

R. Reference

1) VGG19
2) Batch Normalization

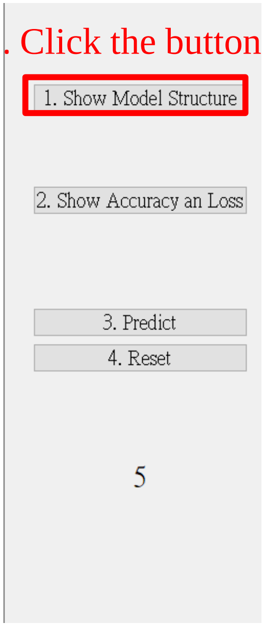
4.1 Show the Structure of VGG19 with BN (6%)

(出題：Shang)

1. Click the button “1. Show Model Structure”
2. Show the VGG19 with BN model on terminal using torchsummary.summary().

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

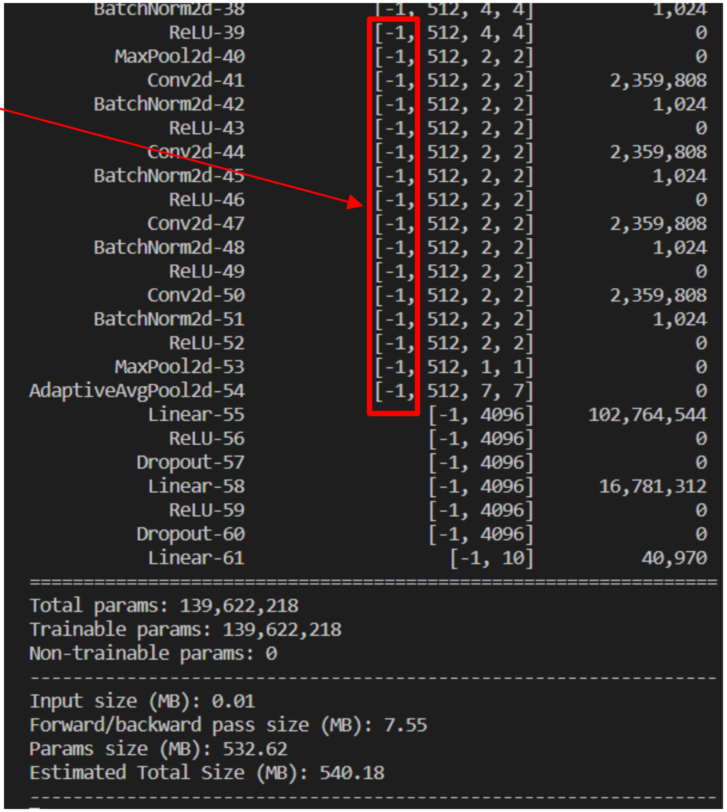
1. Click the button.



Layer (type)

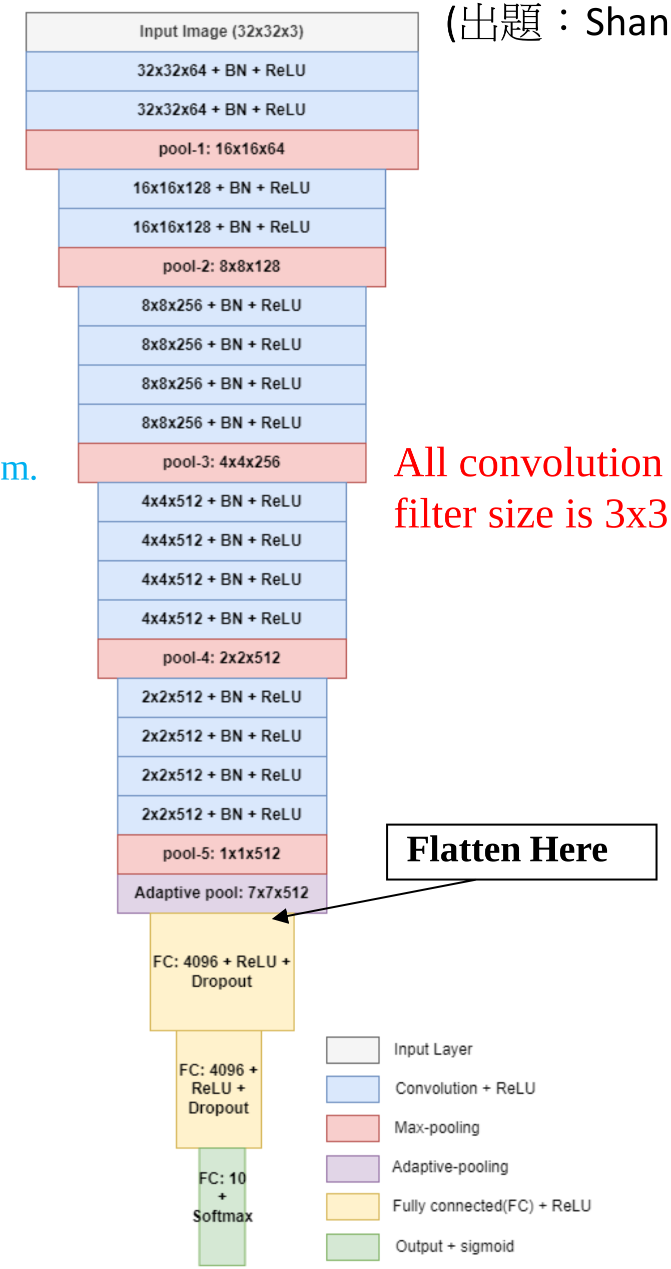
Feature map shape (Batch, Channels, Height, Width)

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



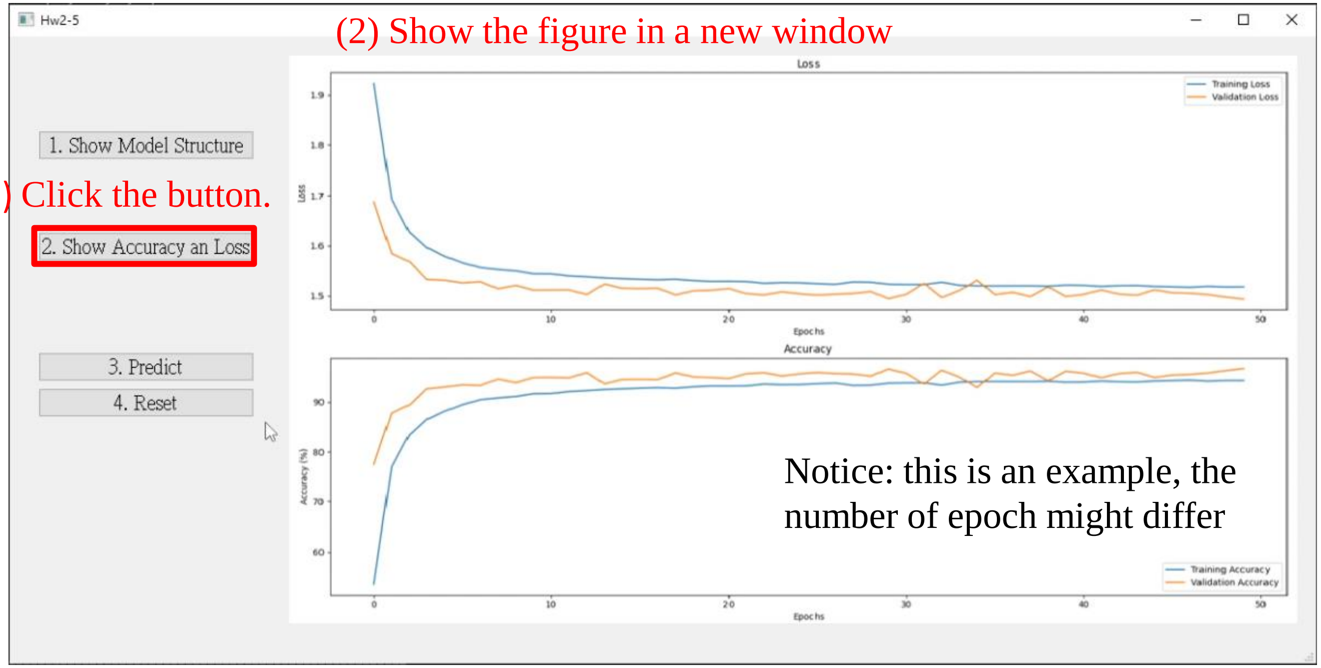
All convolution filter size is 3x3

Figure: VGG19 with BN model structure

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

(出題：Shang)

1. **At home:**
   1) Download the training and validation datasets. (tutorial)
   2) Training and validating VGG19 with BN at least 30 epochs at home (tutorial) and record the training/validating accuracy and loss in each epoch (tutorial).
   3) If your validation accuracy is low, you can try
      - Adjust the learning rate of the optimizer.
      - 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 and save the figure.
2. **When the demo:**
   (1) Click the button "2. Show Accuracy and Loss"
   (2) Show the saved figure of Training/Validating loss and accuracy in a new window



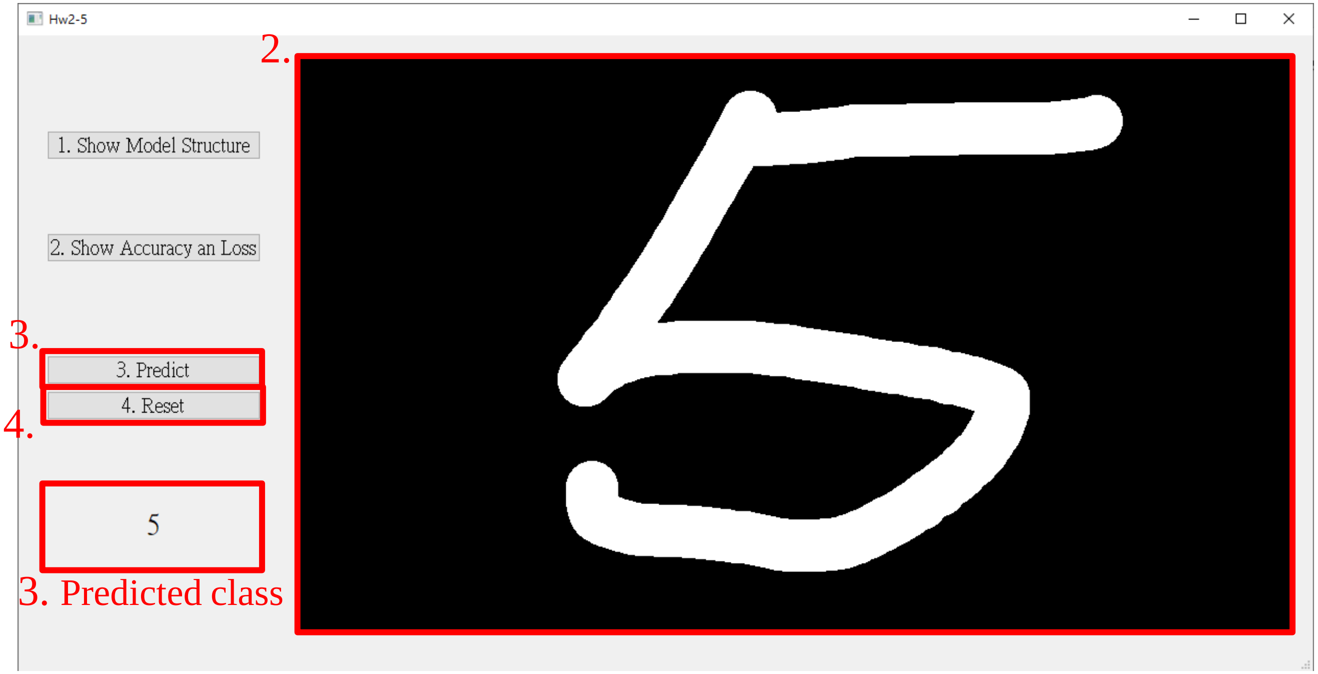
(2) Show the figure in a new window

(1) Click the button.

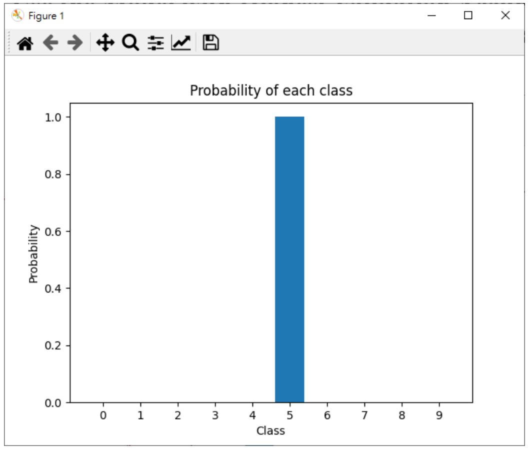
Notice: this is an example, the number of epoch might differ

4.3 Use the Model with Highest Validation Accuracy to Run Inference, Show the Predicted Distribution and Class Label. (8%) (出題：Shang)

1. Load the model with highest validation accuracy which trained at home.
2. Draw a number on graffiti board using mouse. (tutorial)
   - Background: black
   - Pen: white
3. Click the button “3. Predict” to run inference on the image you drew.
   - Show the predicted class label on the GUI.
   - Show the probability distribution of model predictions using a histogram in a new window.
4. Click the button “4. reset” to clear the graffiti board.



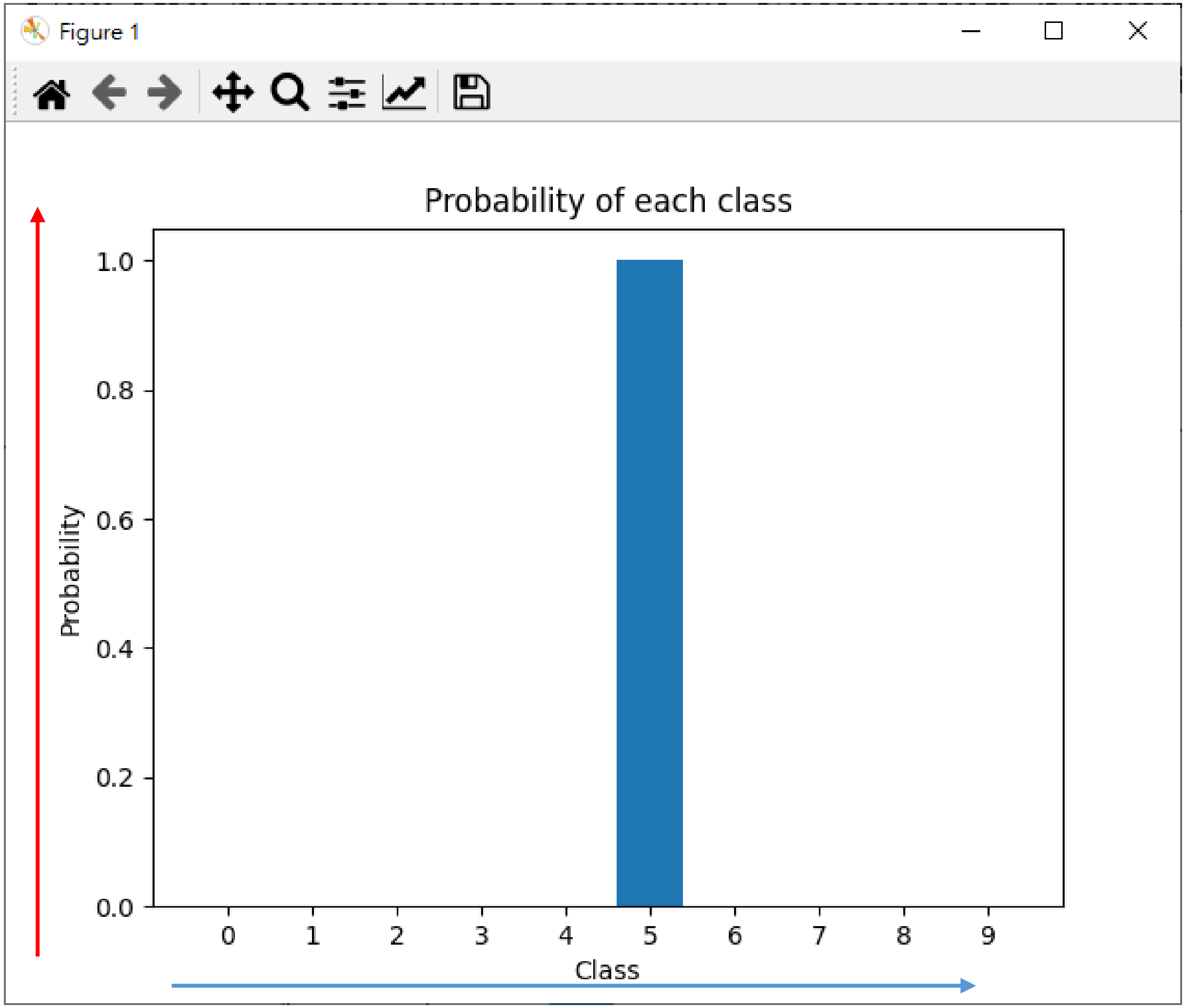
3. Probability of each class



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

- The probability distribution of model prediction using a histogram.

Y-axis: probability

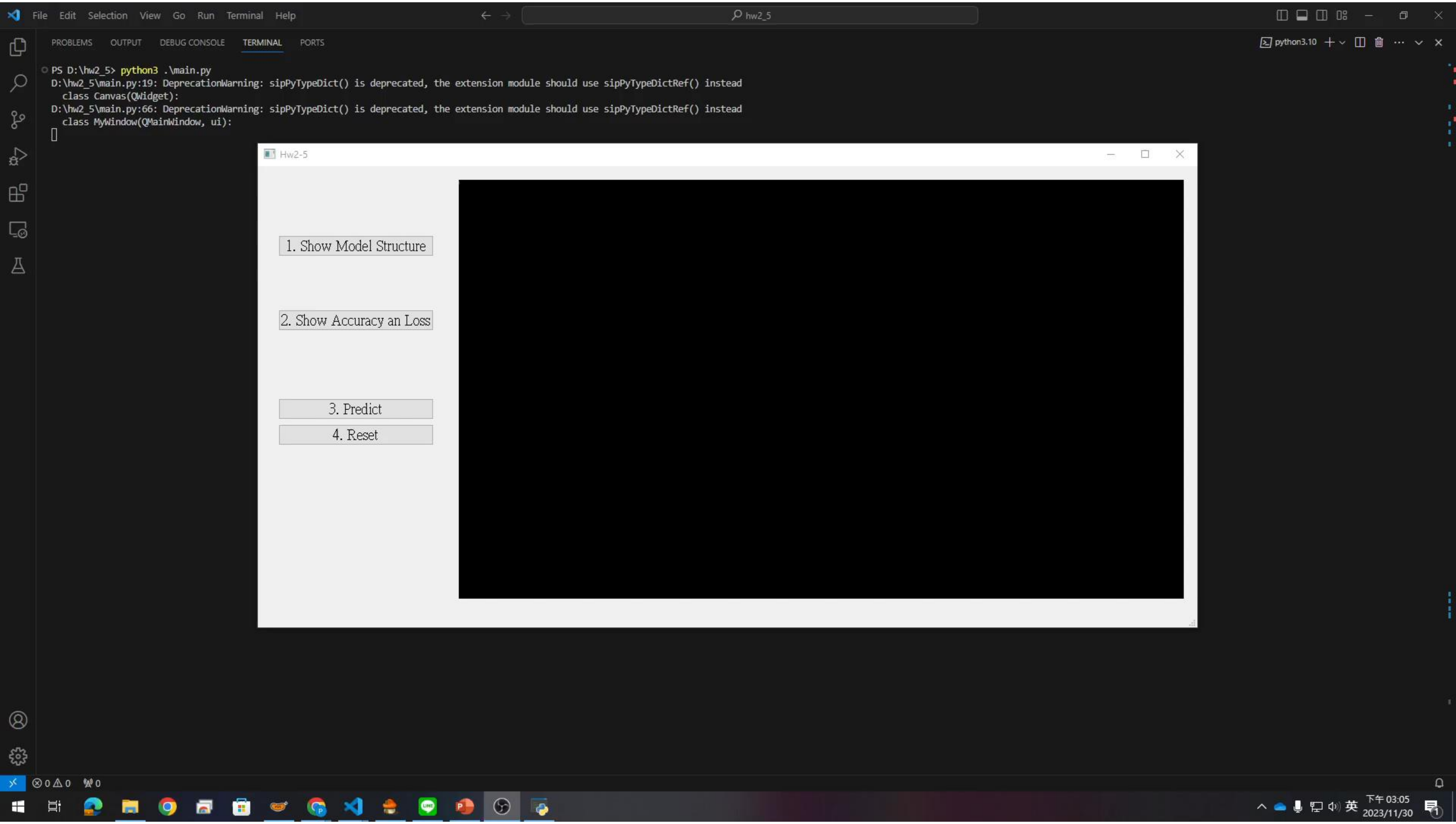


X-axis: class name

4. Training a MNIST Classifier Using VGG19 – Example Video

(出題：Shang)

- This is an example illustrating the objectives from 4.1 ~ 4.3.



```
PS D:\hw2_5> python3 .\main.py
D:\hw2_5\main.py:19: DeprecationWarning: sipPyTypeDict() is deprecated, the extension module should use sipPyTypeDictRef() instead
  class Canvas(QWidget):
D:\hw2_5\main.py:66: DeprecationWarning: sipPyTypeDict() is deprecated, the extension module should use sipPyTypeDictRef() instead
  class MyWindow(QMainWindow, ui):
```

5. Train a Cat-Dog Classifier Using ResNet50 (20%) (出題：Shan)

5.1 (5%) Load the dataset and resize images

5.2 (5%) Show the structure of ResNet50 model

5.3 (5%) Improve ResNet50 with Random-Erasing and Compare the accuracies of 2 ResNet50 models on validation dataset

5.4 (5%) Use the trained model to run inference and show the predicted class label

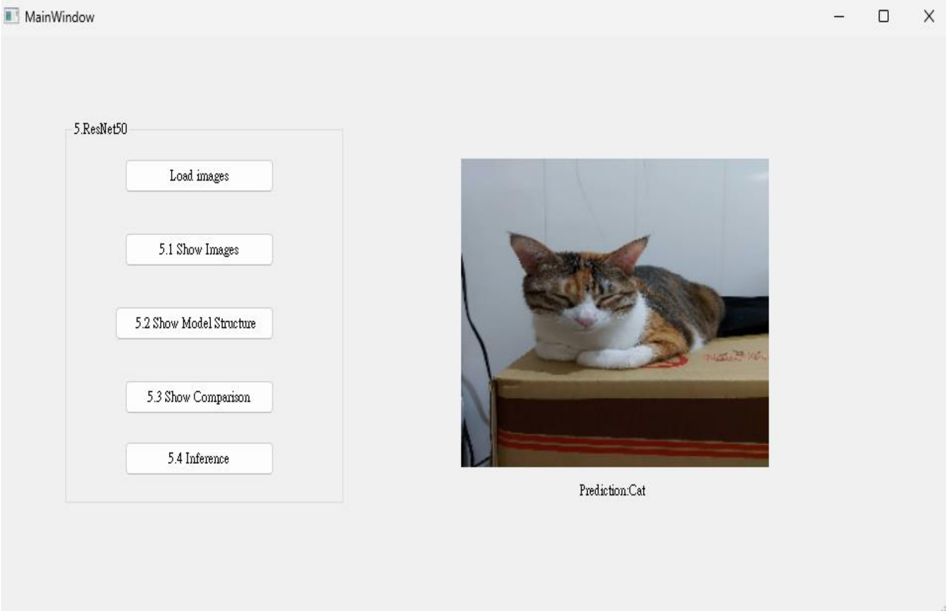


Figure: GUI Example

5.0 Train a Cat-Dog Classifier Using ResNet50

(出題： Shan)

1. Objective

1) Learn how to train a ResNet50 model to classify images of cats and dogs using PyTorch (tutorial)

2. Download Cats and Dogs Dataset from FTP

1) Data type: JPG images
2) 2 classes: Cat and Dog
3) Datasets
   (1) Training dataset: 16,200 JPG images in total.
   (2) Validation dataset: 1,800 JPG images in total.
   (3) Inference dataset: 10 JPG images in total.
       It is for testing the inference function in your GUI program.

3. In the submitted file

1) Organize the files in this structure:

```
Hw2_StudentID_Name_Version // project folder
|-- model              // folder to put trained models
|-- inference_dataset
    |-- Cat
    |-- Dog
|-- main.py          // codes for your GUI program
|-- train.py         // codes for model training
|--   ⋮              // other files or folders you need
```

Notice: Please include the inference dataset in your homework file.

R. Reference

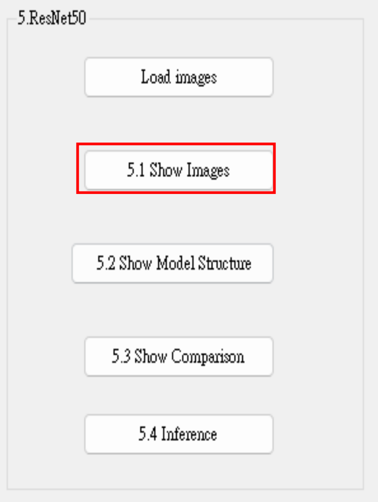
1) Deep Residual Learning for Image Recognition
2) Kaggle Cats and Dogs Dataset

5.1 (5%) Load the dataset and resize images

(出題： Shan)

1) At home:
   (1) Load the inference dataset
       → Hint:
         (a) PyTorch (tutorial): torch.utils.data.Dataset
   (2) Resize images to 224×224×3c (RGB)
       → Hint:
         (a) PyTorch (tutorial): torchvision.transform
   (3) Click the button "1. Show Images"
   (4) Get 1 image from each class in the inference dataset
   (5) Show images in a new window
       → Hint: use matplotlib.pyplot functions to show images (tutorial):
         (a) figure()
         (b) imshow()
         (c) subplot()
         (d) title()

2) When the demo:
   (1) Click the button "1. Show Images"
   (2) Show images in a new window



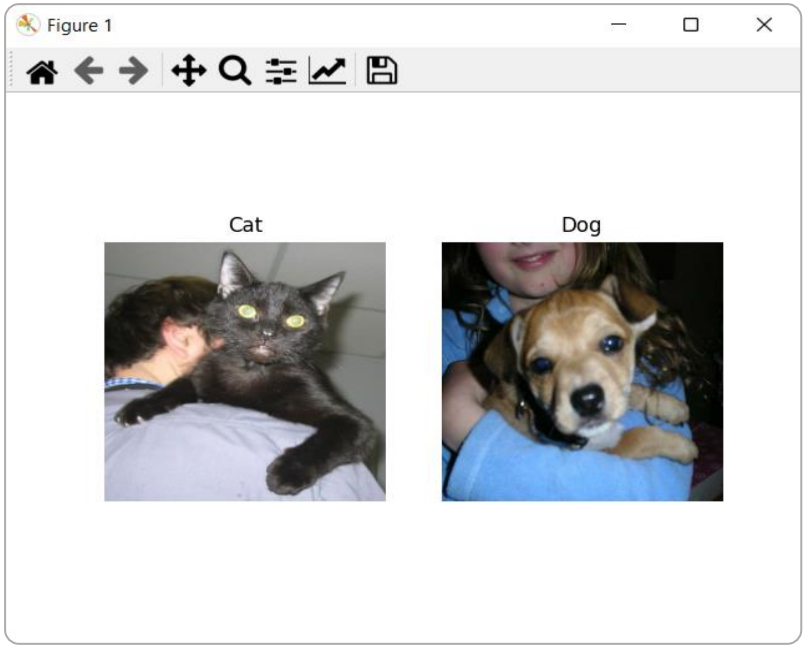
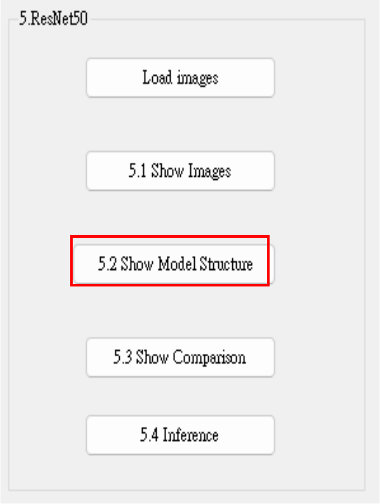


Figure: 1 image from each class
Notice: this is an example, the images might differ

5.2 (5%) Show the structure of ResNet50 model

(出題： Shan)

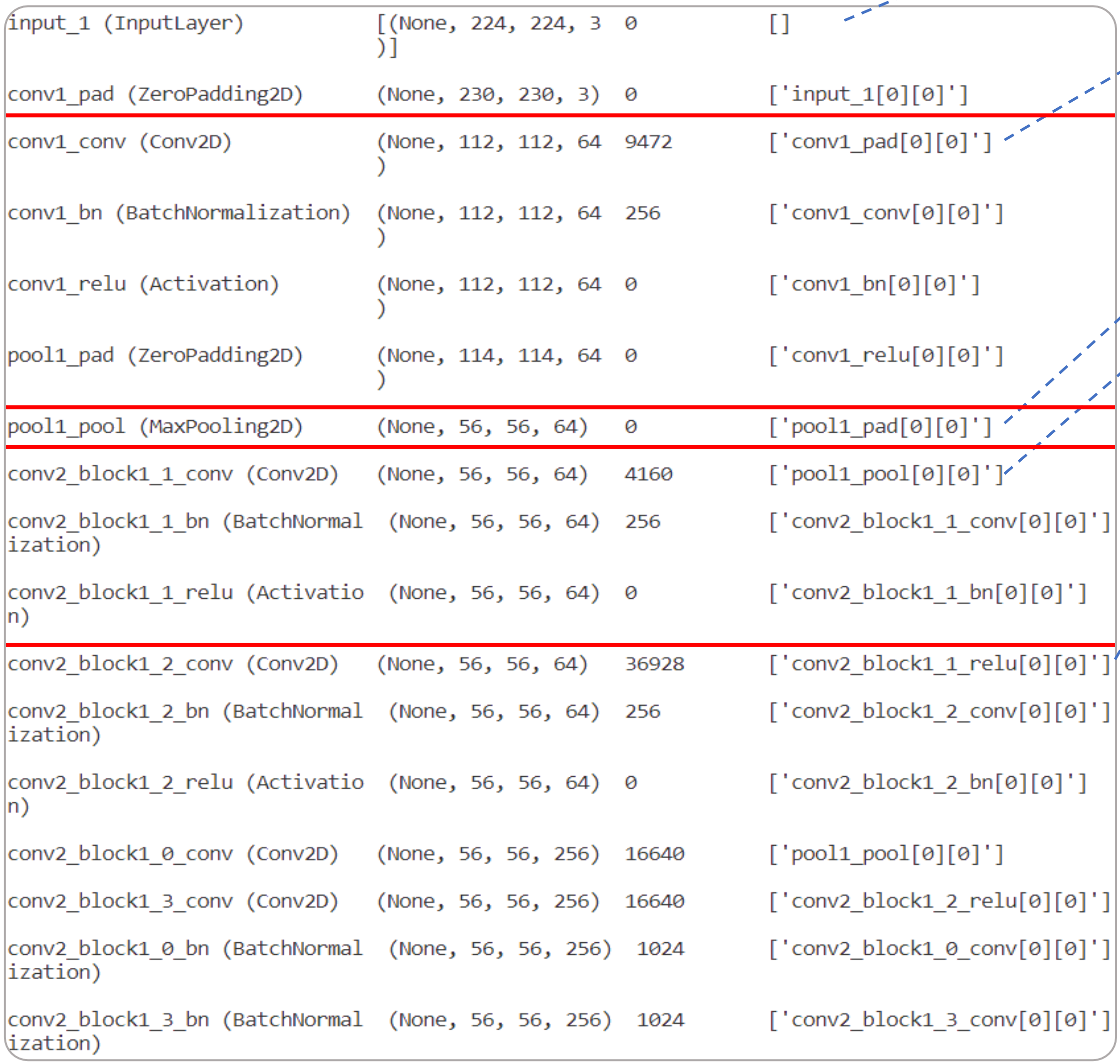
1) At home:
   (1) Build a ResNet50 model
   → Hint:
   (a) PyTorch: torchvision.models.resnet50()
   (2) Replace the output layer to a FC (Fully Connected) layer of 1 node with a Sigmoid activation function
   → Hint:
   (a) PyTorch (tutorial): torch.nn.Linear(2048, 1), torch.nn.Sigmoid
   If the class label of Cat is 1, the output value (range: 0 ~ 1) should be close to 1 for cat images, and vice versa.
   (3) Run the function to show the structure in the terminal
   → Hint:
   (a) PyTorch: torchsummary
2) When the demo:
   (1) Click the button "3. Show Model Structure"
   (2) Run the function to show the structure in the terminal



5.2 (5%) Show the structure of ResNet50 model

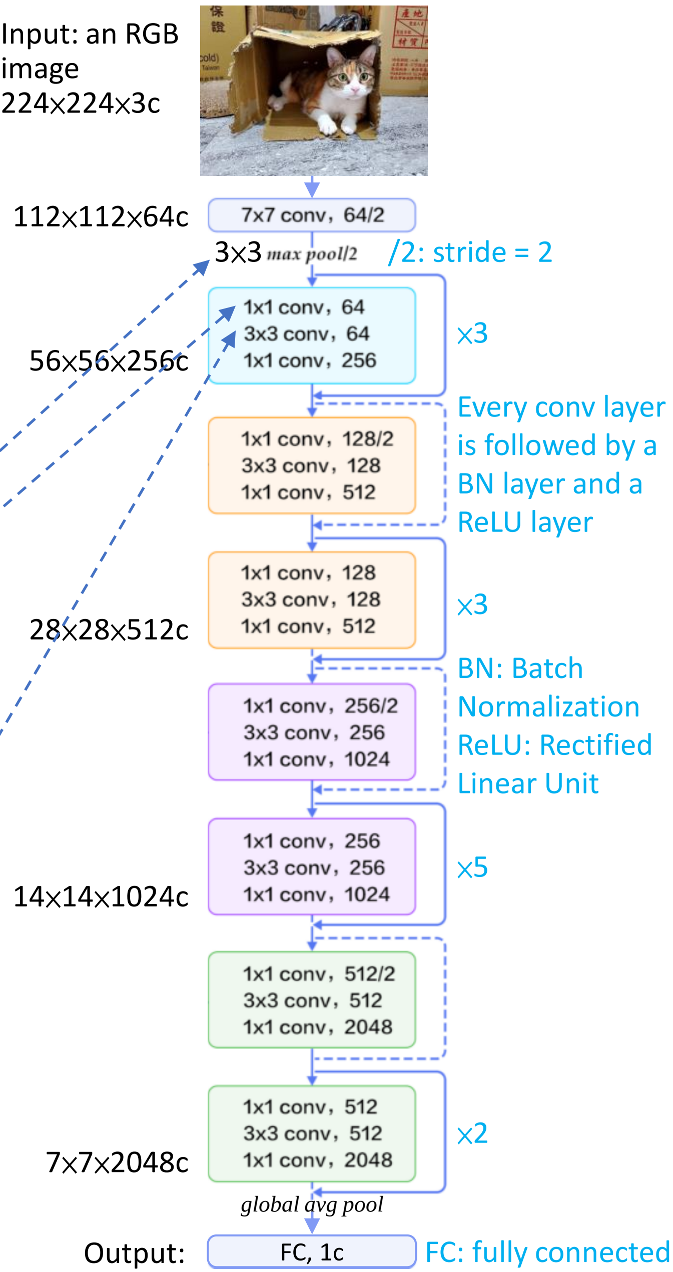
(出題： Shan)

Layer (type) | Feature map shape (B, H, W, C) | Num. of param. | Layer connecting to



```
input_1 (InputLayer)           [(None, 224, 224, 3  0           []
                               )]

conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input_1[0][0]']

conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1_pad[0][0]']
                               )

conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1_conv[0][0]']
                               )

conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1_bn[0][0]']
                               )

pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1_relu[0][0]']
                               )

pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0           ['pool1_pad[0][0]']

conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160        ['pool1_pool[0][0]']

conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block1_1_conv[0][0]']
ization)

conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block1_1_bn[0][0]']
n)

conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2_block1_1_relu[0][0]']

conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block1_2_conv[0][0]']
ization)

conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block1_2_bn[0][0]']
n)

conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1_pool[0][0]']

conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2_block1_2_relu[0][0]']

conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2_block1_0_conv[0][0]']
ization)

conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2_block1_3_conv[0][0]']
ization)
```

Figure: the structure of ResNet50 model



Input: an RGB image 224x224x3c

Every conv layer is followed by a BN layer and a ReLU layer

BN: Batch Normalization

ReLU: Rectified Linear Unit

FC: fully connected

Output: a confidence value (range: 0 ~ 1)

5.3 (5%) Improve ResNet50 with Random-Erasing

(出題： Shan)

1) At home: Set up **Random-Erasing** in codes for model training (train.py)
   (1) Train 2 ResNet50 models with training dataset
   → Hint:(a) PyTorch (tutorial): write a for loop to validate the model
   (a) With Random-Erasing
   (b) Without Random-Erasing

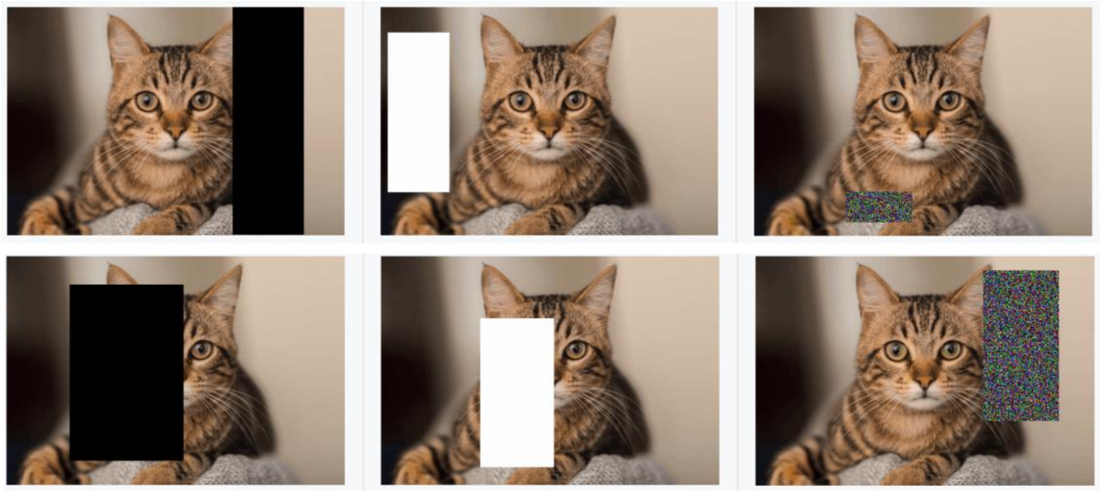
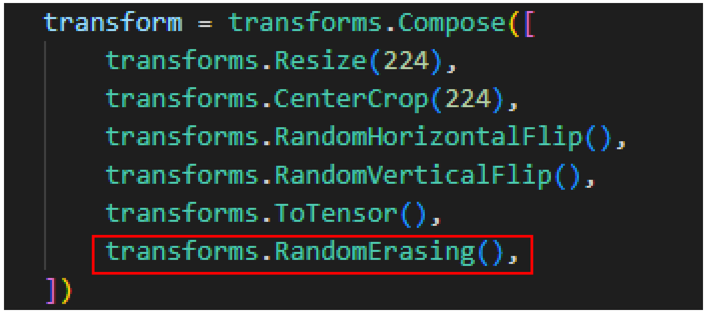


Figure1: Examples of the use of Random-Erasing

2) When the demo: Show your codes about Random-Erasing in train.py



```
transform = transforms.Compose([
    transforms.Resize(224),
    transforms.CenterCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.RandomVerticalFlip(),
    transforms.ToTensor(),
    transforms.RandomErasing(),
])
```

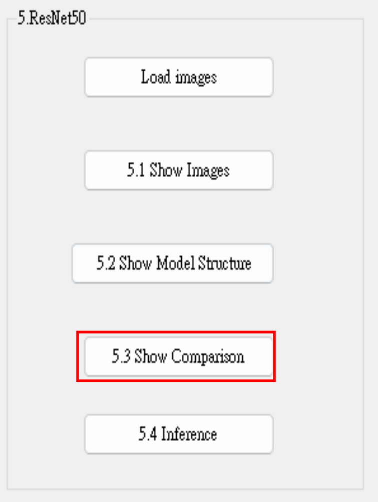
R. Reference

Random Erasing Data Augmentation

5.3 (5%) Compare the accuracies of 2 ResNet50 models on validation dataset

(出題： Shan)

1) At home:
   (1) Validate 2 ResNet50 models with validation dataset
       → Hint:
         (a) PyTorch (tutorial): write a for loop to validate the model
   (2) Plot the accuracy values with a bar chart
   (3) Save the figure
2) When the demo:
   (1) Click the button “4. Show Comparison”
   (2) Show the saved figure of accuracy comparison in a new window



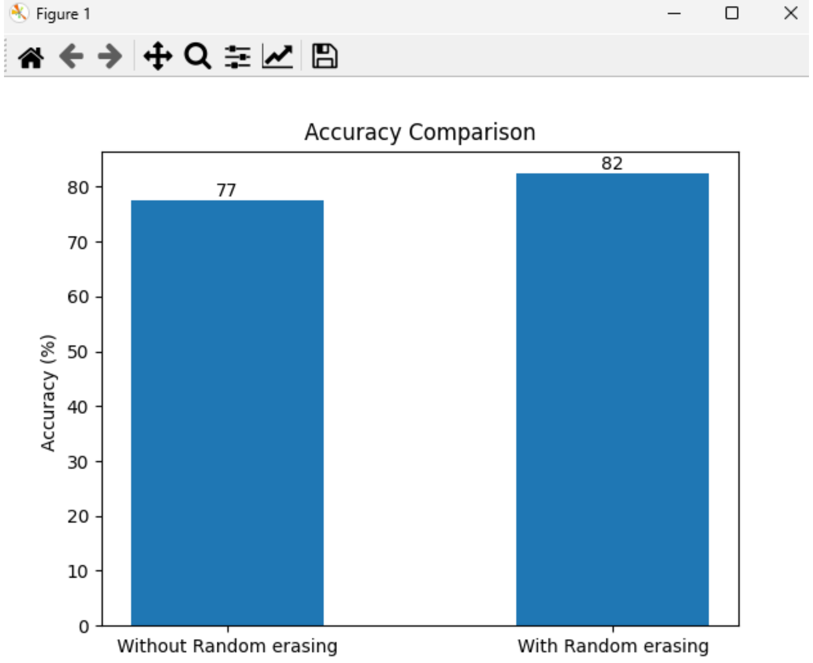


Figure1: Accuracy Comparison

Notice: this is an example, the numbers might differ

5.4 (5%) Use the better-trained model to run inference and show the predicted class label

(出題： Shan)

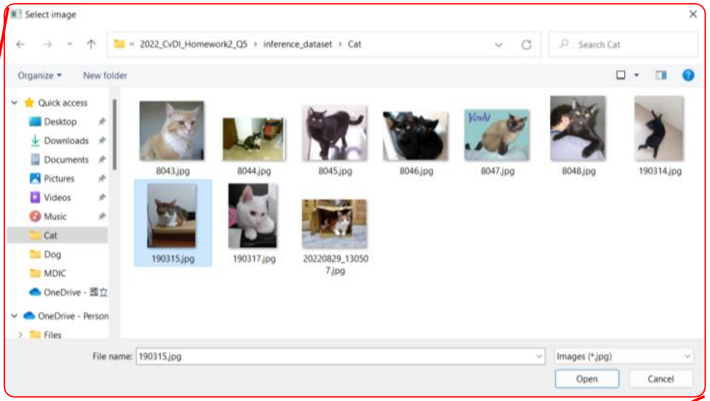
1) At home:
   (1) Load the trained model
      → Hint:
         (a) PyTorch: torch.nn.Module.load_state_dict()
   (2) Click the button "Load Image" to select 1 image arbitrarily
      → Hint: PyQt5.QtWidgets.QFileDialog.getOpenFileName()
   (3) Show the loaded image in the GUI
   (4) Resize the loaded image to 224×224×3c (RGB)
   (5) Click the button "5. Inference" to run inference on the resized image
      → Hint:
         (a) PyTorch: pass an image when calling torch.nn.Module object to run inference, ex: trained_model(img)
   (6) Show the predicted class label
      → Hint: decide the class label with a threshold of the output value.

$$\text{Ex: class label} = \begin{cases} \text{Cat}, & output < thresh \\ \text{Dog}, & output \geq thresh \end{cases}$$

$$thresh = 0.5$$

2) When the demo: repeat the process

(2) Select 1 image arbitrarily

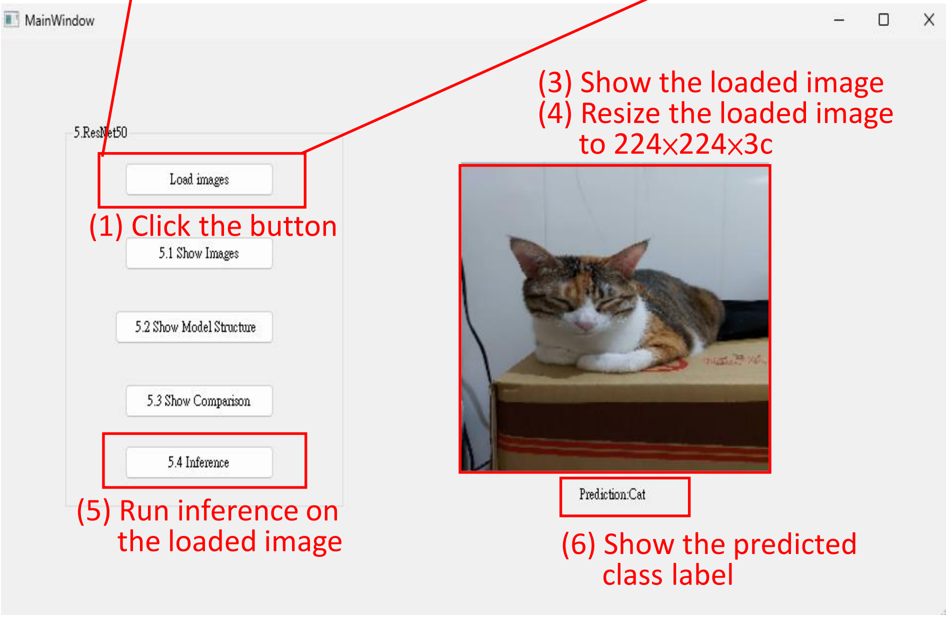


Select image

2022_CvDI_Homework2_Q5 › inference_dataset › Cat

File name: 190315.jpg

Images (*.jpg)



(1) Click the button

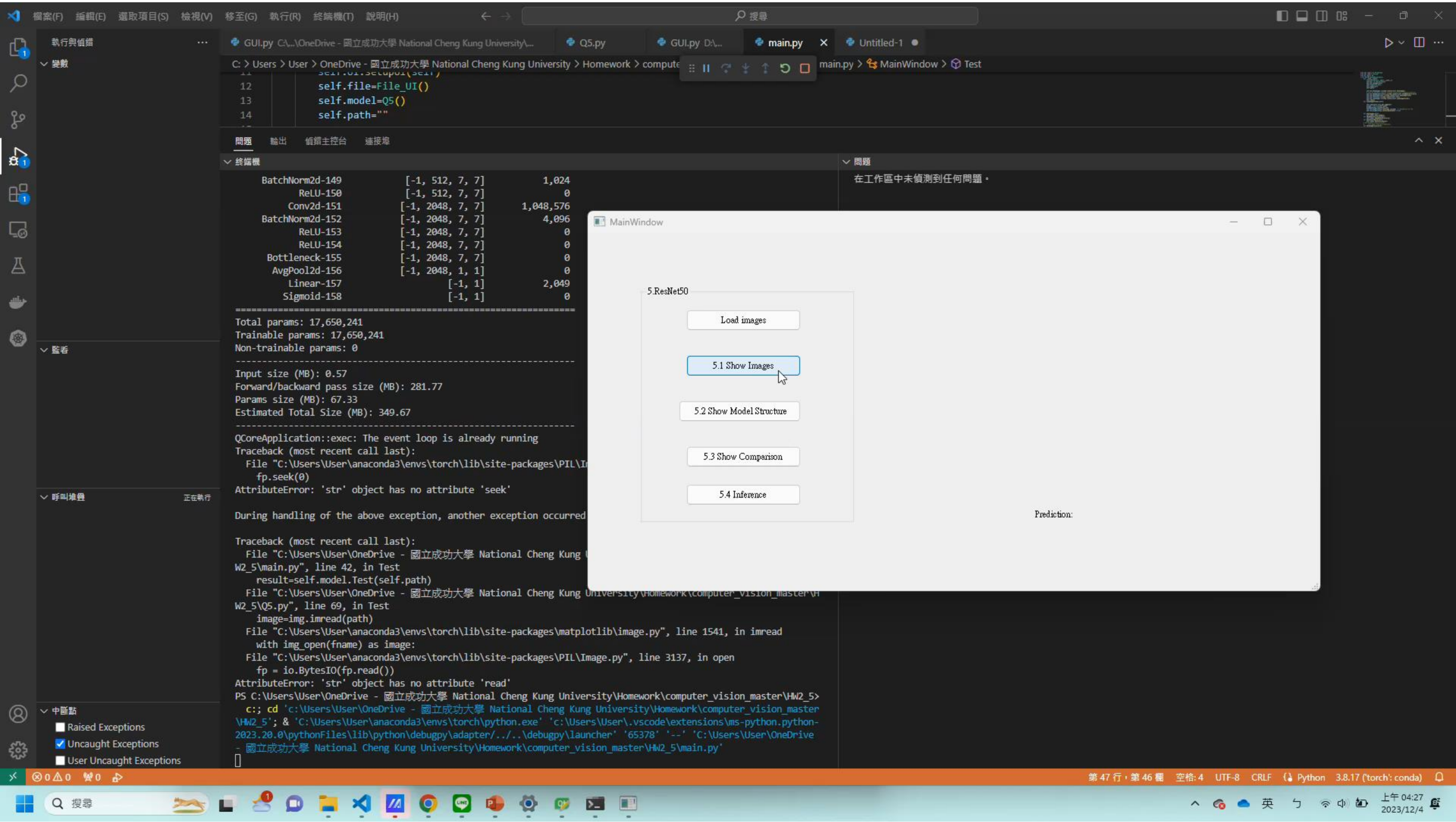
(3) Show the loaded image
(4) Resize the loaded image to 224×224×3c

(5) Run inference on the loaded image

(6) Show the predicted class label

5. Train a Cat-Dog Classifier Using ResNet50 – Demo Video

- This is an example illustrating the objectives from 5.1 ~ 5..



```
        self.file=File_UI()
        self.model=Q5()
        self.path=""
```

```
     BatchNorm2d-149           [-1, 512, 7, 7]           1,024
            ReLU-150           [-1, 512, 7, 7]               0
          Conv2d-151          [-1, 2048, 7, 7]       1,048,576
     BatchNorm2d-152          [-1, 2048, 7, 7]           4,096
            ReLU-153          [-1, 2048, 7, 7]               0
            ReLU-154          [-1, 2048, 7, 7]               0
      Bottleneck-155          [-1, 2048, 7, 7]               0
       AvgPool2d-156          [-1, 2048, 1, 1]               0
          Linear-157                   [-1, 1]           2,049
         Sigmoid-158                   [-1, 1]               0
================================================================
Total params: 17,650,241
Trainable params: 17,650,241
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 281.77
Params size (MB): 67.33
Estimated Total Size (MB): 349.67
----------------------------------------------------------------
QCoreApplication::exec: The event loop is already running
Traceback (most recent call last):
  File "C:\Users\User\anaconda3\envs\torch\lib\site-packages\PIL\I
    fp.seek(0)
AttributeError: 'str' object has no attribute 'seek'

During handling of the above exception, another exception occurred

Traceback (most recent call last):
  File "C:\Users\User\OneDrive - 國立成功大學 National Cheng Kung
W2_5\main.py", line 42, in Test
    result=self.model.Test(self.path)
  File "C:\Users\User\OneDrive - 國立成功大學 National Cheng Kung University\Homework\computer_vision_master\H
W2_5\Q5.py", line 69, in Test
    image=img.imread(path)
  File "C:\Users\User\anaconda3\envs\torch\lib\site-packages\matplotlib\image.py", line 1541, in imread
    with img_open(fname) as image:
  File "C:\Users\User\anaconda3\envs\torch\lib\site-packages\PIL\Image.py", line 3137, in open
    fp = io.BytesIO(fp.read())
AttributeError: 'str' object has no attribute 'read'
PS C:\Users\User\OneDrive - 國立成功大學 National Cheng Kung University\Homework\computer_vision_master\HW2_5>
  c:; cd 'c:\Users\User\OneDrive - 國立成功大學 National Cheng Kung University\Homework\computer_vision_master
\HW2_5'; & 'C:\Users\User\anaconda3\envs\torch\python.exe' 'c:\Users\User\.vscode\extensions\ms-python.python-
2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '65378' '--' 'C:\Users\User\OneDrive
- 國立成功大學 National Cheng Kung University\Homework\computer_vision_master\HW2_5\main.py'
```