

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 2

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, 2023/12/19 (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/Hw2**
 - **User ID: opencvdl2023** **Password: RL2023opencv**
- 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%) Hough Circle Transform (出題：You)

1.1 (15%) Draw Contour

1.2 (5%) Count Rings

2. (20%) Histogram Equalization (出題：Eric)

3. (20%) Morphology Operation (出題：Hsiang)

3.1 (10%) Closing

3.2 (10%) Opening

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 (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

* 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.

Load Image

1. Hough Circle Transform

1.1 Draw Contour

1.2 Count Coins

There are _ coins in the image.

2. Histogram Equalization

2. Histogram Equalization

3. Morphology Operation

3.1 Closing

3.2 Opening

4. MNIST Classifier Using VGG19

1. Show Model Structure

2. Show Accuracy an Loss

3. Predict

4. Reset

5. ResNet50

Load Image

5.1. Show Images

5.2. Show Model Structure

5.3. Show Comprasion

5.4. Inference

1. (20%) Hough Circle Transform

(出題 : You)

1.1 (15%) Draw Contour

1.2 (5%) Count Rings

1.1 Hough Circle Transform – Draw Contour

□ Given: One color image, “coins.jpg”

□ Q: 1) **Draw Contour**: Using OpenCV functions to find the contours of coins in image.

2) **Circle Center**: Show center points of the circle with most votes.

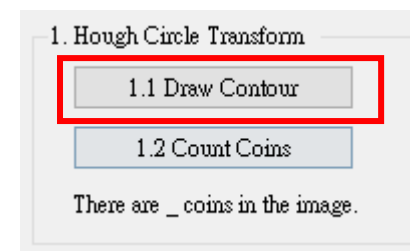
□ Hint: Textbook Chapter 6, p.153 ~ p.161

1. RGB □ Grayscale

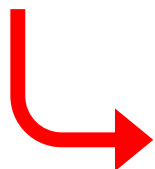
2. Remember to remove the noise. (Use **cv2.GaussianBlur(5, 5)**)

3. Using circle detection function to get result.
(Use **cv2.HoughCircles**)

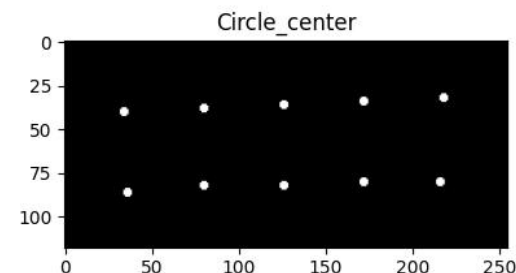
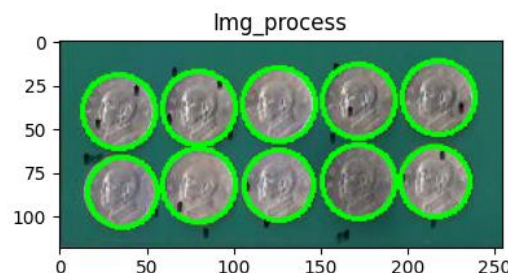
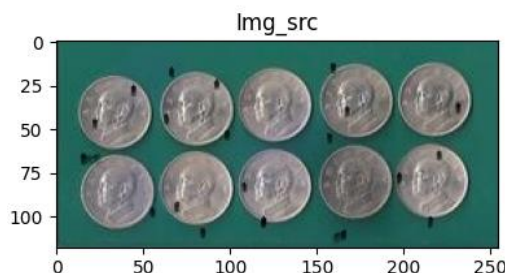
4. Display the original image, processed image, and circle center image at the same time.



coins.jpg: 255x118



Draw Contours



1.2 Hough Circle Transform – Count Coins

- ❑ Given: One color image, “coins.jpg”
- ❑ Q: 3) **Count coins**: Using OpenCV functions to find how many coins in the image.
- ❑ Hint: Textbook Chapter 6, p.153 ~ p.161
Calculate how many coins and show on your UI.

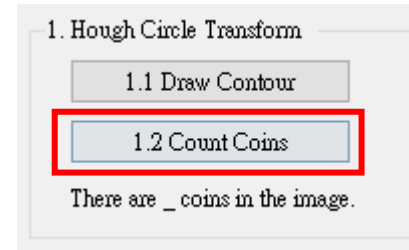


coins.jpg: 255x118

Calculate



There are 10 coins in the image.



2. (20%) Histogram Equalization

(出題 : Eric)

➤ Given: A single grayscale image will be provided as input.

Q: Perform histogram equalization on a grayscale image to enhance its contrast using OpenCV and manual(PDF&CDF).

Part 1: Using OpenCV

- 1) Load histoEqual.png (click “Load Image” button).
 - Hint: your_image_path = `filedialog.askopenfilename()`
- 2) Perform histogram equalization using `cv2.equalizeHist()` function from OpenCV to process image.
- 3) The equalized image will be displayed alongside the original image for comparison in the upper quadrants.
- 4) The histograms of both the original and the equalized images will be shown in the lower quadrants.
 - Hint: using `plt.bar()` to display the frequency (y-axis) of each grayscale value (x-axis).

Part 2: Using PDF and CDF (tutorial)

- 1) The **Probability Density Function (PDF)** represents the frequency of each grayscale level in the image.
- 2) The **Cumulative Distribution Function (CDF)** is the cumulative sum of the PDF and is used to map the old grayscale values to new ones.
- 3) The program calculates the histogram of the grayscale image using `numpy.histogram()`.
- 4) The **PDF** is obtained from the normalized histogram.
- 5) The **CDF** is calculated by **cumulatively summing the PDF**.
- 6) A lookup table is created based on the rounded CDF values, it is applied to the original image to create a new equalized image.
- 7) The histogram of the manually equalized image is also plotted, showing the redistributed grayscale frequencies.
- 8) Click “2. Histogram Equalization” button to show result.

2. (20%) Histogram Equalization

(出題 : Eric)

➤ Result:

Original Image



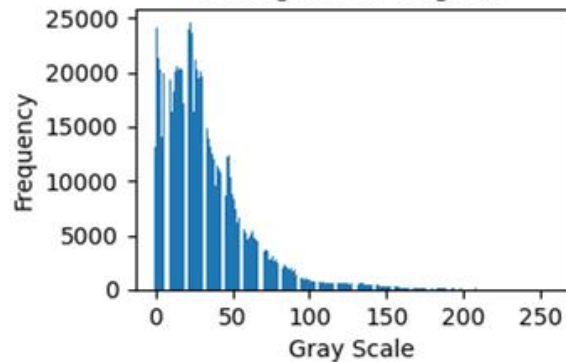
Equalized with OpenCV



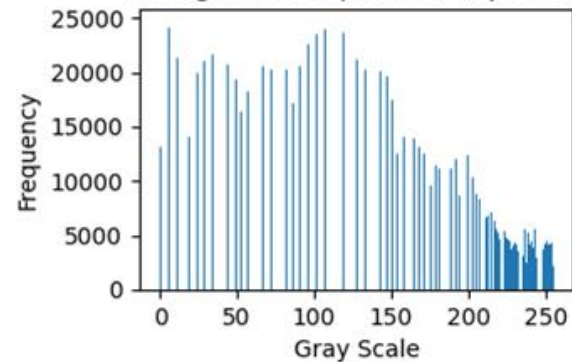
Equalized Manually



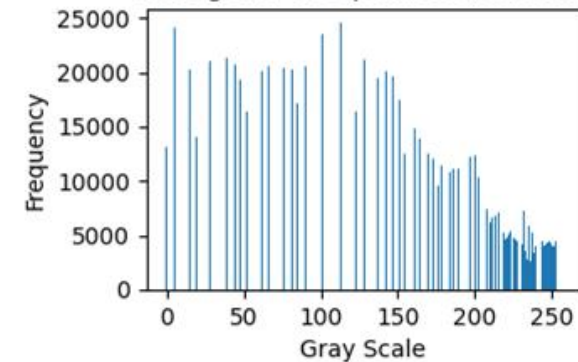
Histogram of Original



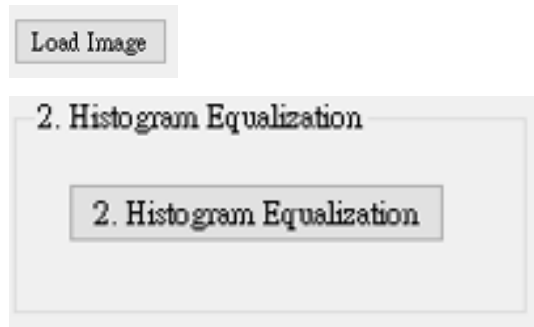
Histogram of Equalized (OpenCV)



Histogram of Equalized (Manual)



➤ GUI:



3. Morphology Operation (20%)

3.1 (10%) Closing

3.2 (10%) Opening

3. Morphology Operation (1/3)

Steps of Erosion Operation:

1. Define Structuring Element

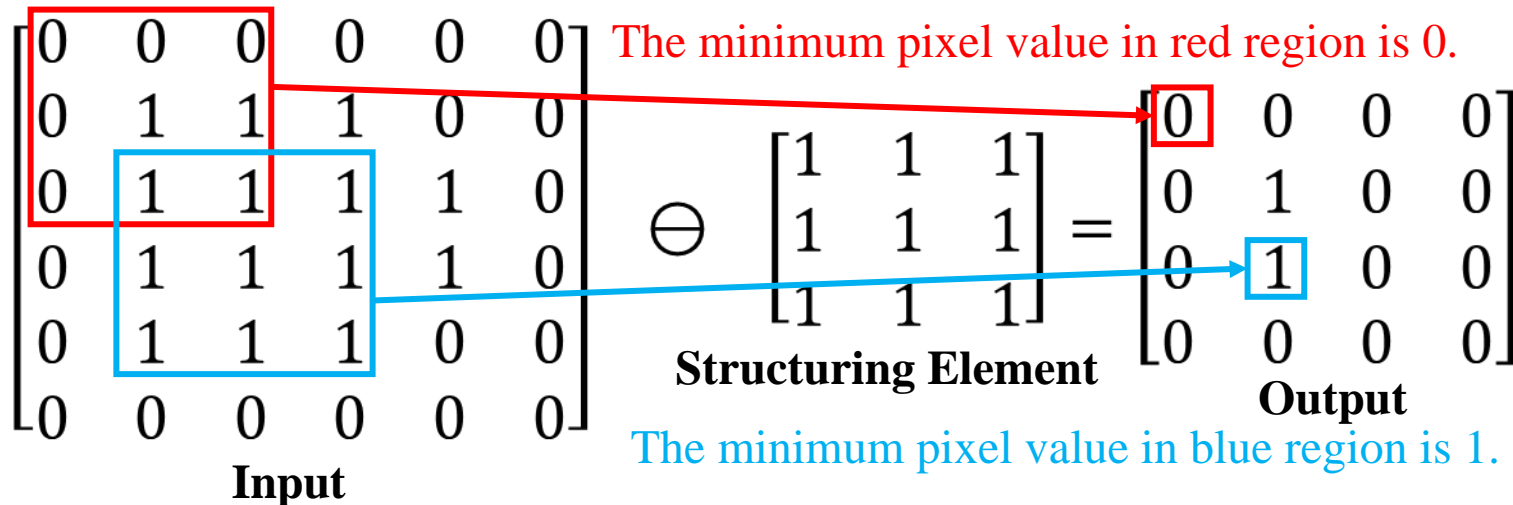
Choose a structuring element, typically a small square or rectangular convolution kernel. The size and shape of this element determine the effect of the erosion.

2. Scanning

Initiate the process by sliding the structuring element from the top-left corner of the image. Sequentially cover each pixel in the image, moving both horizontally from left to right and vertically from top to bottom.

3. Pixel Update

For each covered region, update the pixel values covered by the structuring element to the **minimum pixel value** in that region.



3. Morphology Operation (2/3)

Steps of Dilation Operation:

1. Define Structuring Element

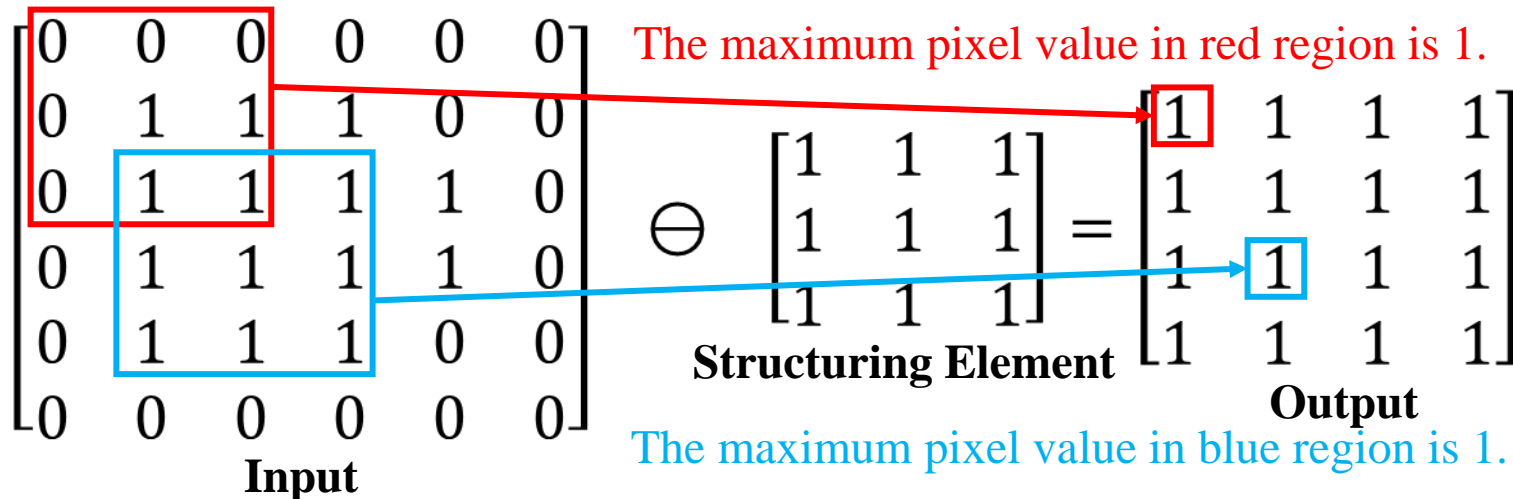
Choose a structuring element, typically a small square or rectangular convolution kernel. The size and shape of this element determine the effect of the erosion.

2. Scanning

Initiate the process by sliding the structuring element from the top-left corner of the image. Sequentially cover each pixel in the image, moving both horizontally from left to right and vertically from top to bottom.

3. Pixel Update

For each covered region, update the pixel values covered by the structuring element to the **maximum pixel value** in that region.



3. Morphology Operation (3/3)

1. Closing Operation :

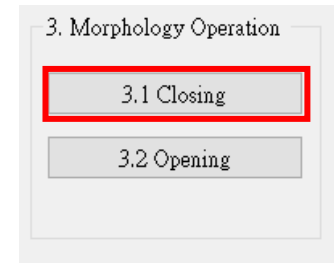
- An operation performed by first applying dilation followed by erosion :
 $\text{Closing}(A) = \text{Erosion}(\text{Dilation}(A))$
- The purpose is to **fill small holes** in an image while preserving the shape and size of large holes and objects in the image.

2. Opening Operation :

- An operation performed by first applying erosion followed by dilation :
 $\text{Opening}(A) = \text{Dilation}(\text{Erosion}(A))$
- The purpose is to **remove small objects and thin lines** from an image while preserving the shape and size of larger objects in the image

3.1 Closing (10%)

1. Given: “closing.png”
2. Constraint: **Can not use OpenCV Function cv2.dilate(), cv2.erode(), cv2.morphologyEx()**
3. Question: Perform opening operation on the image
 - 1) Using “Load image” button to load RGB image.
 - 2) Convert the RGB image to grayscale.
 - 3) Binarize the grayscale image, assigning values of 0 or 255 only. (**threshold = 127**)
 - 4) Pad the image with zeros based on the kernel size (**K=3**).
 - 5) Perform the **dilation** operation using a **3x3 all-ones structuring element**.
 - 6) Perform the **erosion** operation using a **3x3 all-ones structuring element**.
 - 7) Show the image in a popup window.

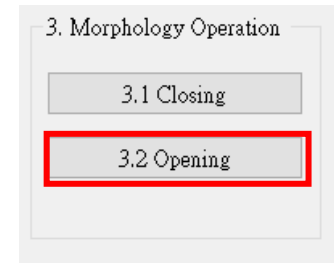


Closing Operation



3.1 Opening (10%)

1. Given: “opening.png”
2. Constraint: **Can not use OpenCV Function cv2.dilate(), cv2.erode(), cv2.morphologyEx()**
3. Question: Perform opening operation on the image
 - 1) Using “Load image” button to load RGB image.
 - 2) Convert the RGB image to grayscale.
 - 3) Binarize the grayscale image, assigning values of 0 or 255 only. (**threshold = 127**)
 - 4) Pad the image with zeros based on the kernel size (**K=3**).
 - 5) Perform the **erosion** operation using a **3x3 all-ones structuring element**.
 - 6) Perform the **dilation** operation using a **3x3 all-ones structuring element**.
 - 7) Show the image in a popup window.



Opening Operation



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%)

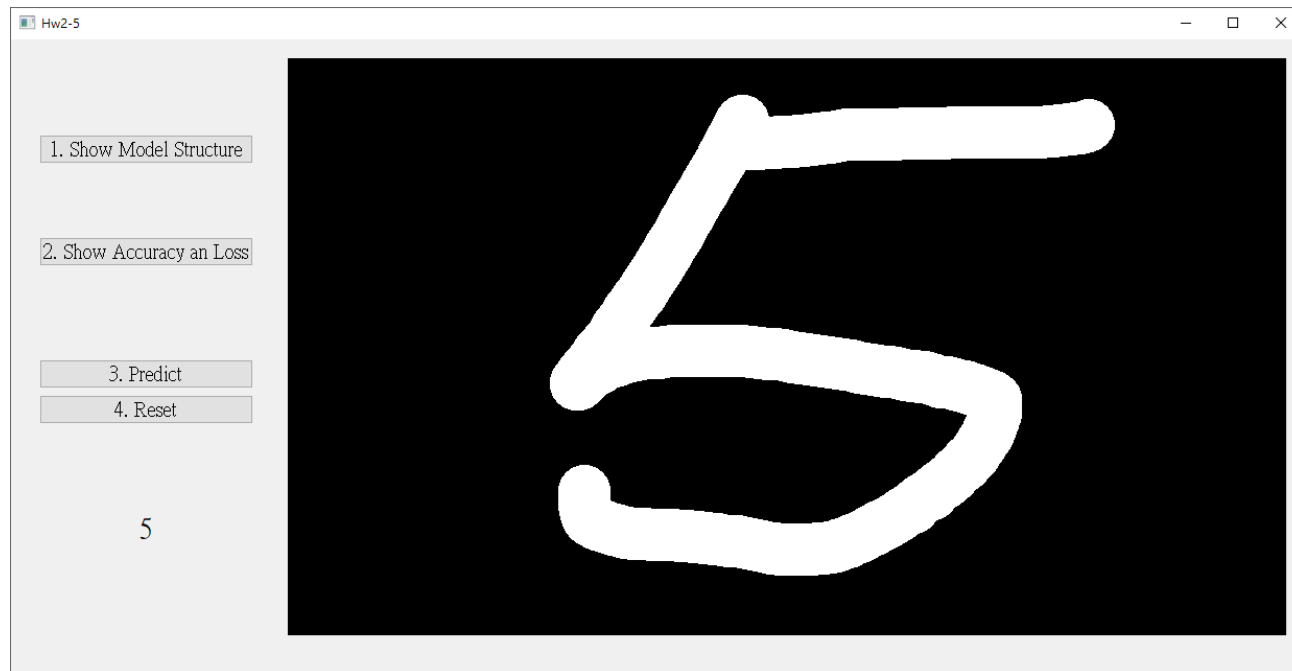


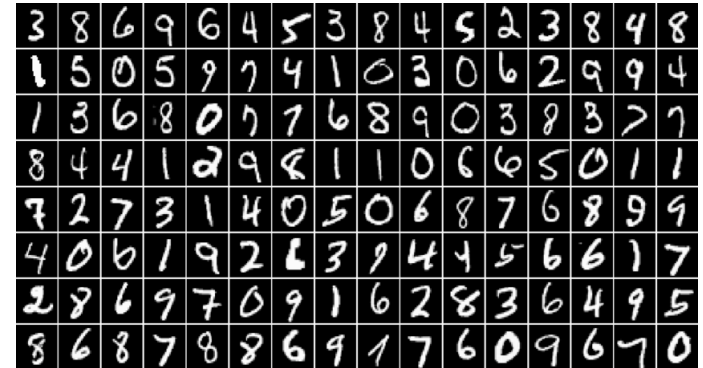
Figure: GUI example

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

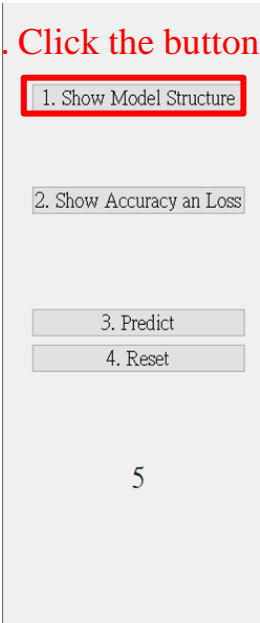
4.1 Show the Structure of VGG19 with BN (6%)

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

(出題 : Shang)

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

1. Click the button.

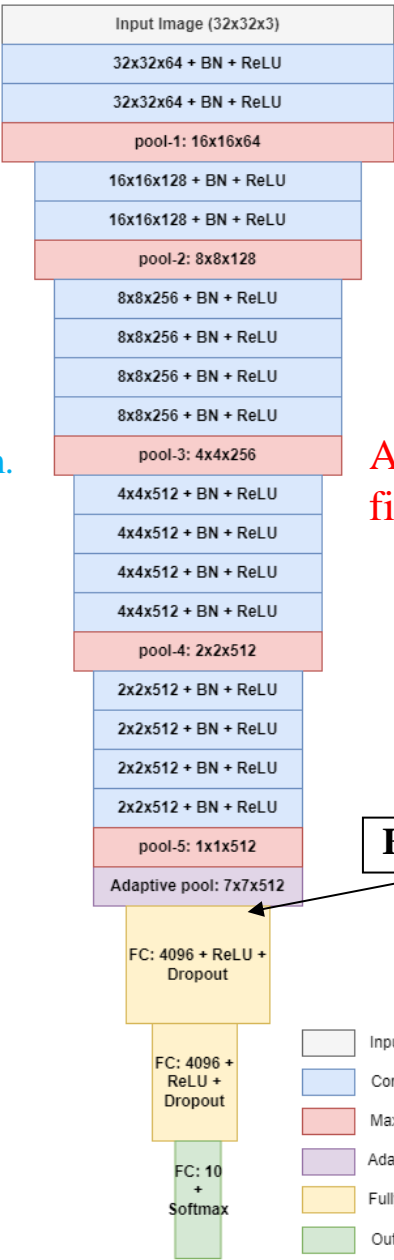


Layer (type)	Feature map shape (Batch, Channels, Height, Width)	Num. of param.
BatchNorm2d-38	[-1, 512, 4, 4]	1,024
ReLU-39	[-1, 512, 4, 4]	0
MaxPool2d-40	[-1, 512, 2, 2]	0
Conv2d-41	[-1, 512, 2, 2]	2,359,808
BatchNorm2d-42	[-1, 512, 2, 2]	1,024
ReLU-43	[-1, 512, 2, 2]	0
Conv2d-44	[-1, 512, 2, 2]	2,359,808
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

Flatten Here

Figure: VGG19 with BN model structure 19

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

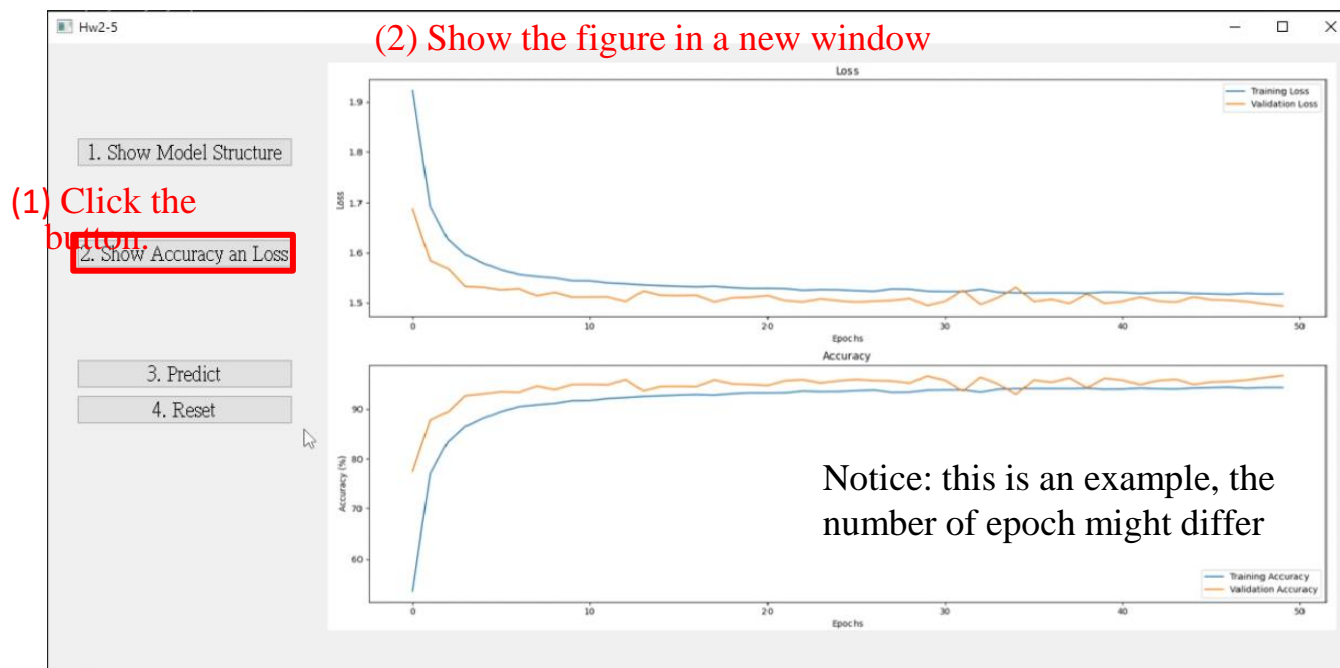
(出題 : Shang)

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.

1. When the demo:

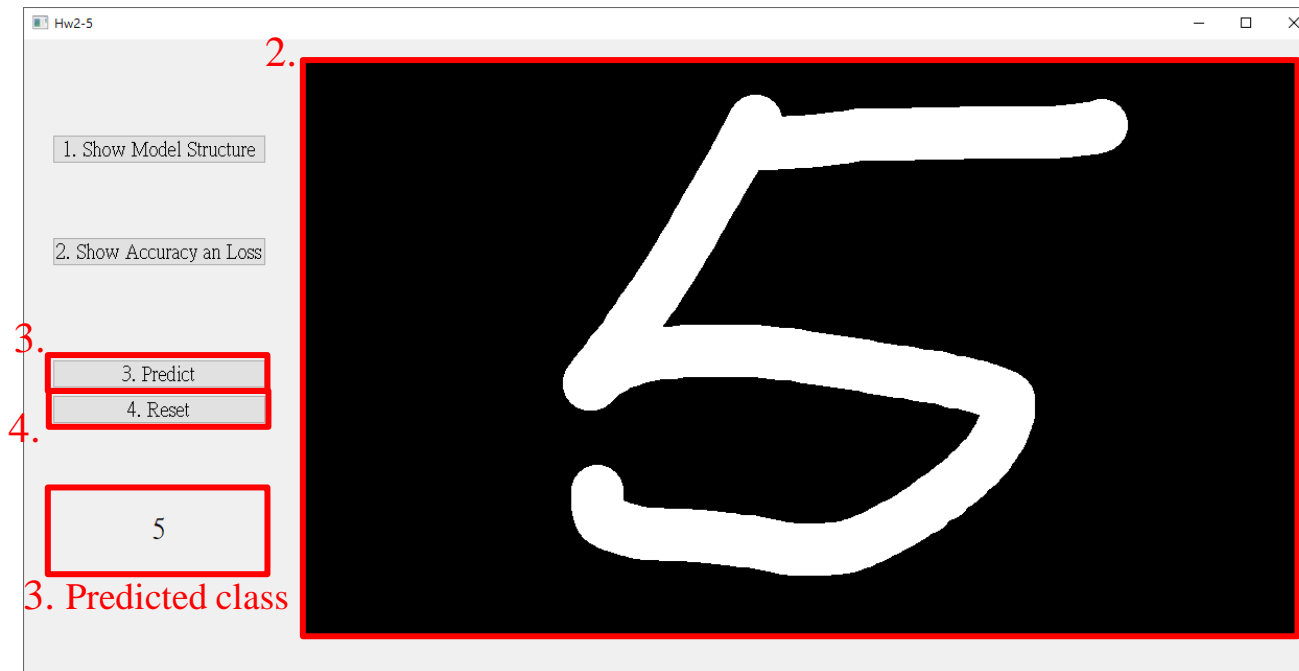
- (1) Click the button “**2. Show Accuracy and Loss**”
- (2) Show the **saved figure** of Training/Validating loss and accuracy in a new window



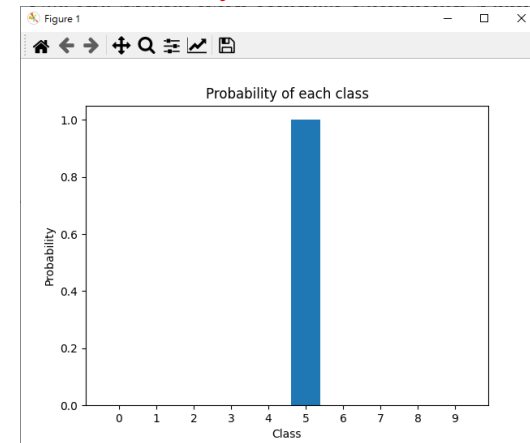
4.3 Use the Model with Highest Validation Accuracy to Run Inference

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.



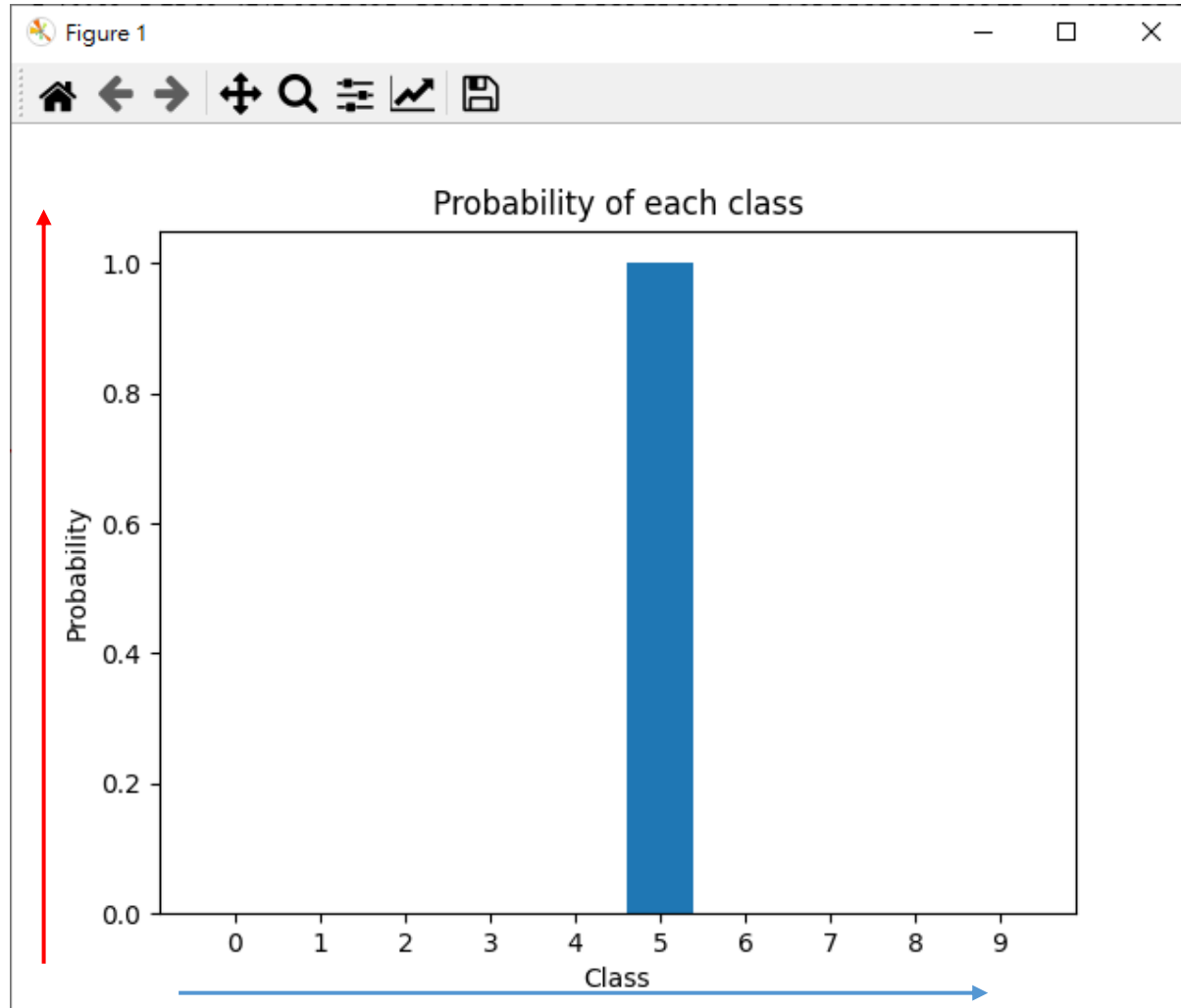
3. Probability of each class



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

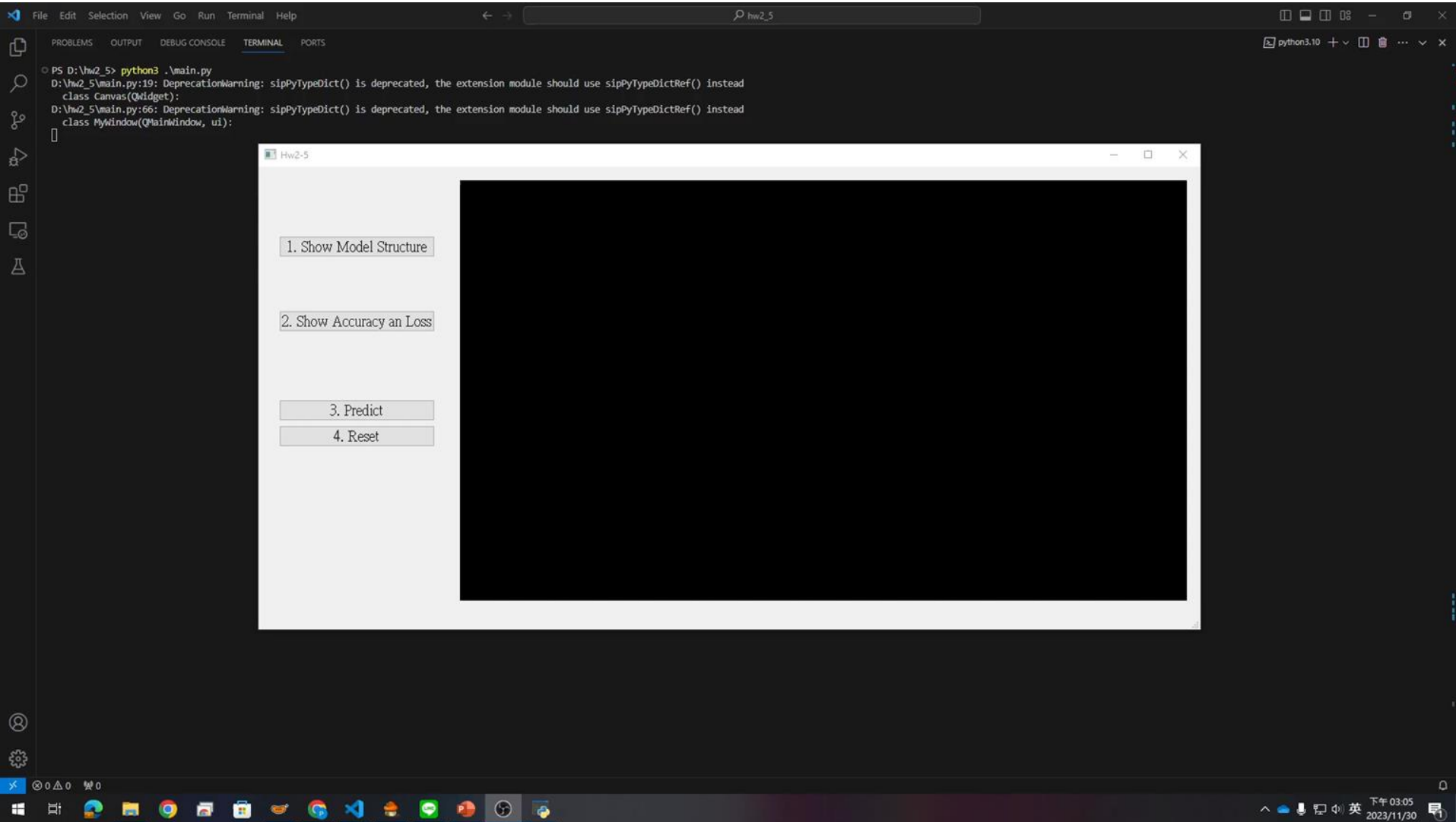


X-axis: class name

4. Training a MNIST Classifier Using VGG19 – Example Video

(出題 : Shang)

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



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

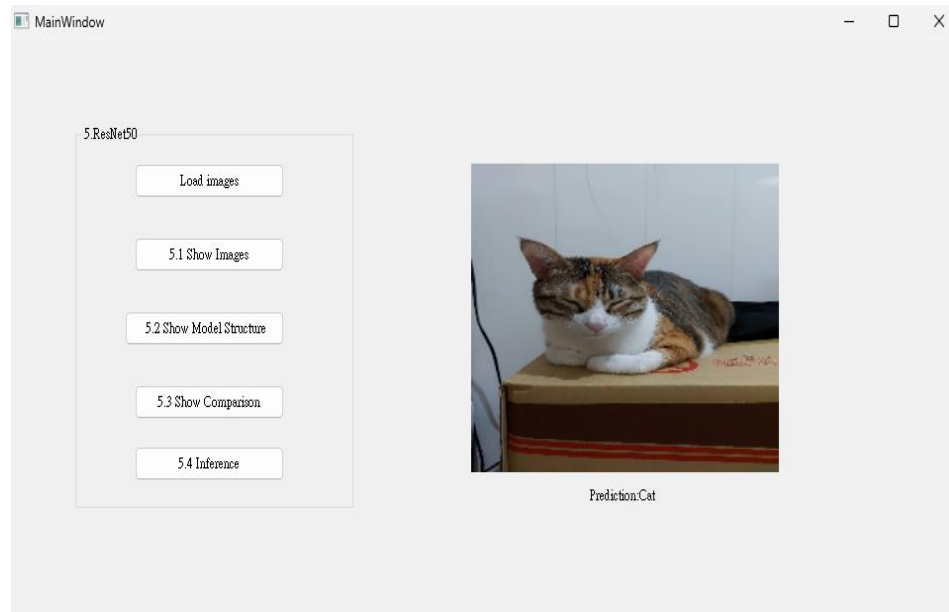


Figure: GUI Example

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 training dataset and validation 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) Load the **inference dataset**

(5) Resize images to **224×224×3c** (RGB)

(6) Get 1 image from **each class** in the inference dataset

(7) 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**

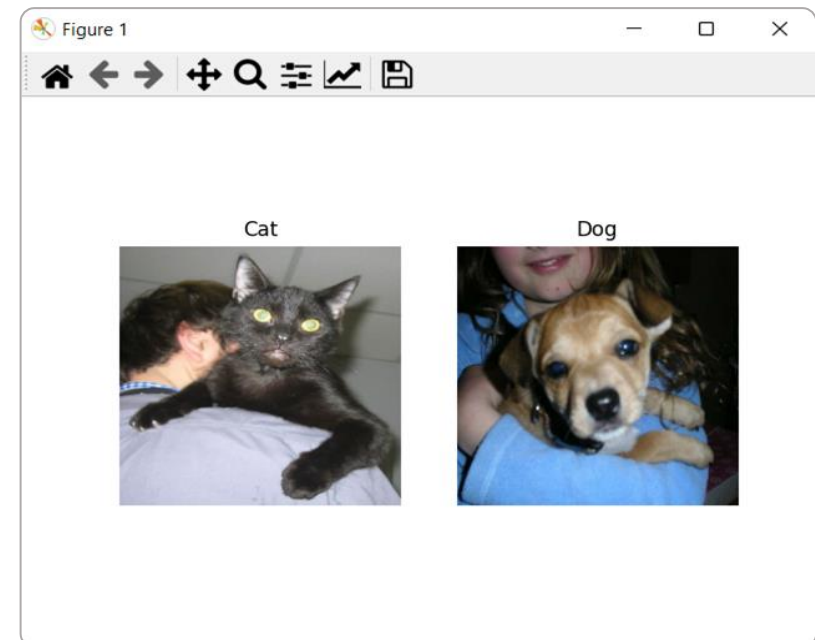
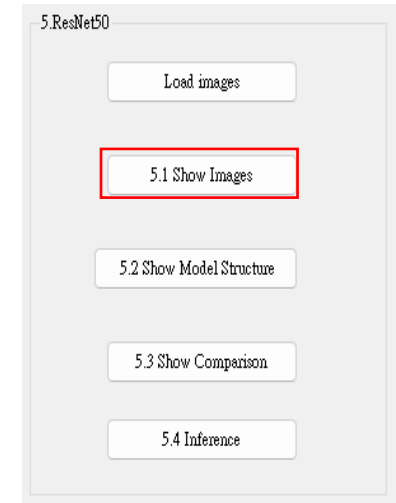


Figure: 1 image from each class
Notice: this is an example, the images might differ

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**

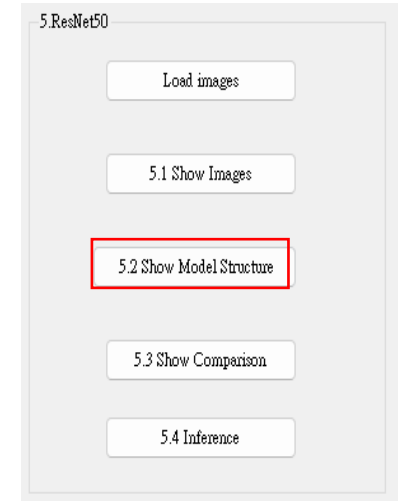
🔗 Hint:

(a) PyTorch: `torchsummary`

2) When the demo:

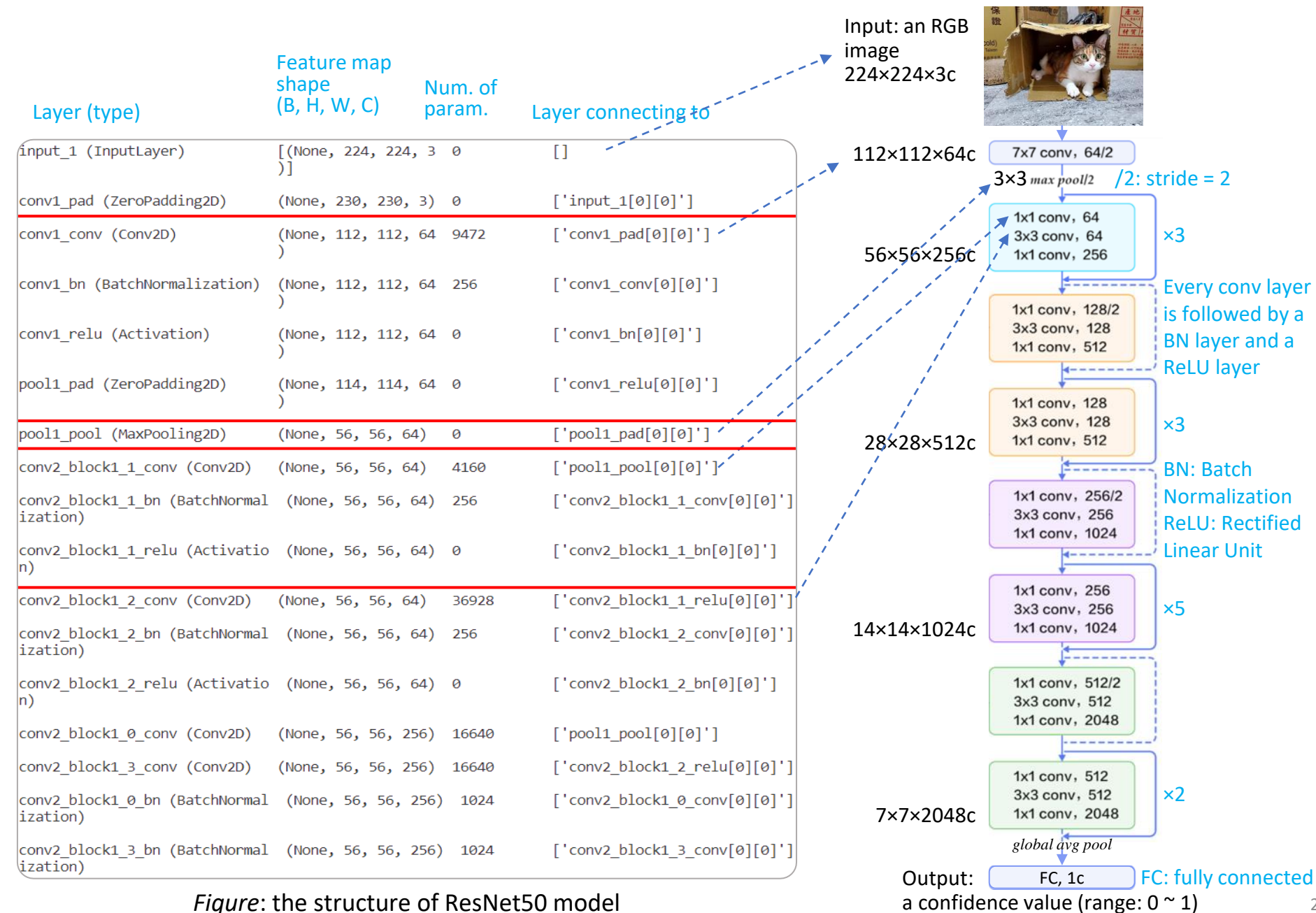
(1) Click the button **“3. Show Model Structure”**

(2) Run the function to show the structure **in the terminal**



5.2 (5%) Show the structure of ResNet50 model

(出題：Shan)



5.3 (5%) Improve ResNet50 with Random-Erasing

- 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

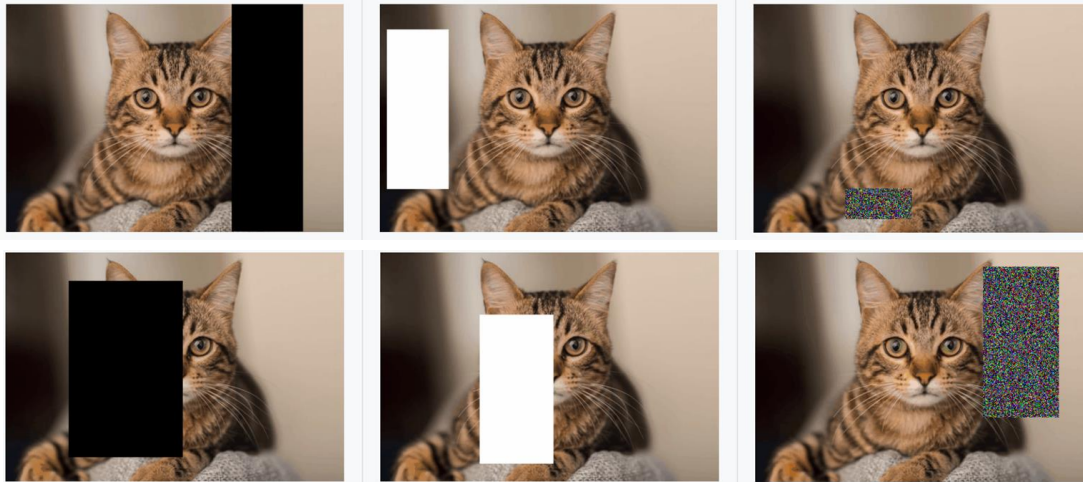


Figure1: Examples of the use of Random-Erasing

- 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

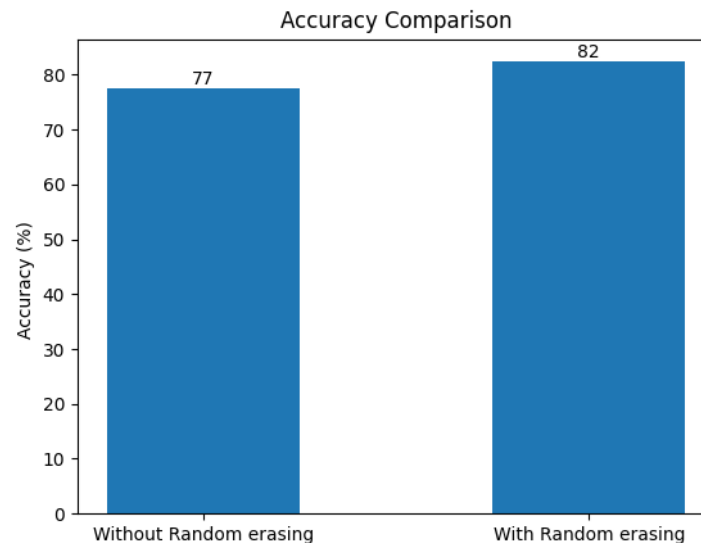
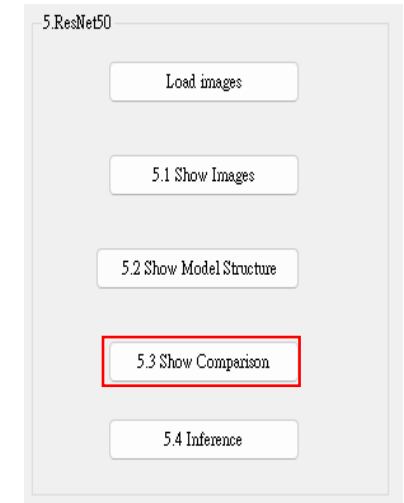


Figure1: Accuracy Comparison

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

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

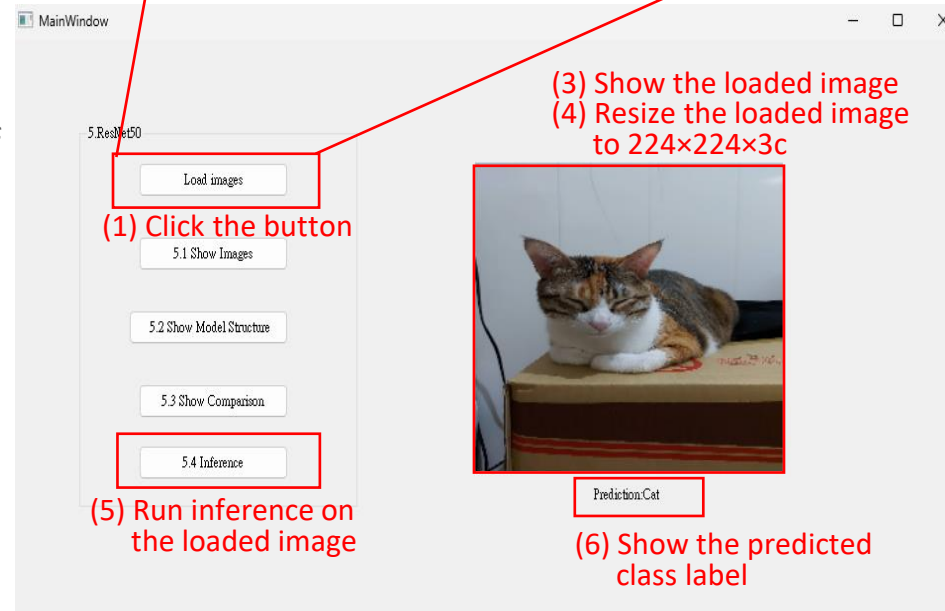
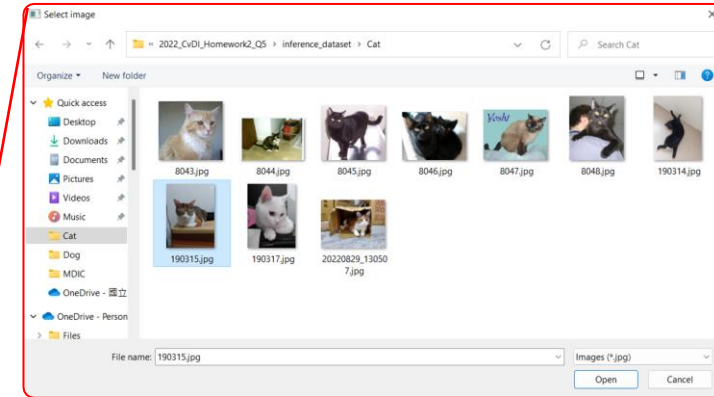
→ Hint: decide the class label with a **threshold** of the **output value**.

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



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

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

The screenshot shows a Jupyter Notebook interface with a dark theme. The notebook is titled 'main.py' and contains the following code:

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

The notebook output shows the ResNet50 model architecture and parameters:

```
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
```

Summary statistics:

```
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
```

The notebook also shows a traceback error:

```
QCoreApplication::exec: The event loop is already running
Traceback (most recent call last):
  File "C:\Users\User\anaconda3\envs\torch\lib\site-packages\PIL\fp.py", line 154, in seek
    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 University\Homework\computer_vision_master\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\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\W2_5>
c:\cd "c:\Users\User\OneDrive - 國立成功大學 National Cheng Kung University\Homework\computer_vision_master\W2_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\W2_5\main.py"
```

The 'MainWindow' dialog box is overlaid on the notebook, containing the following buttons:

- Load images
- 5.1 Show Images
- 5.2 Show Model Structure
- 5.3 Show Comparison
- 5.4 Inference

The dialog box also has a 'Prediction:' label.