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: 吳又成

Gmail: nckubot65904@gmail.com

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

10:00~12:00, Thu.

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, 2024/12/24 (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: opencvdl2024 Password: RL2024opencv
- 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 3.9 (https://www.python.org/downloads/)
  - **Opencv-contrib-python (4.10.0.84)**
  - Matplotlib 3.7.3
  - UI framework: pyqt5 (5.15.11)
  - Pytorch 2.1.0
  - Torchvision 0.16.0
  - Torchsummary 1.5.1

Assignment scoring (Total: 100%)

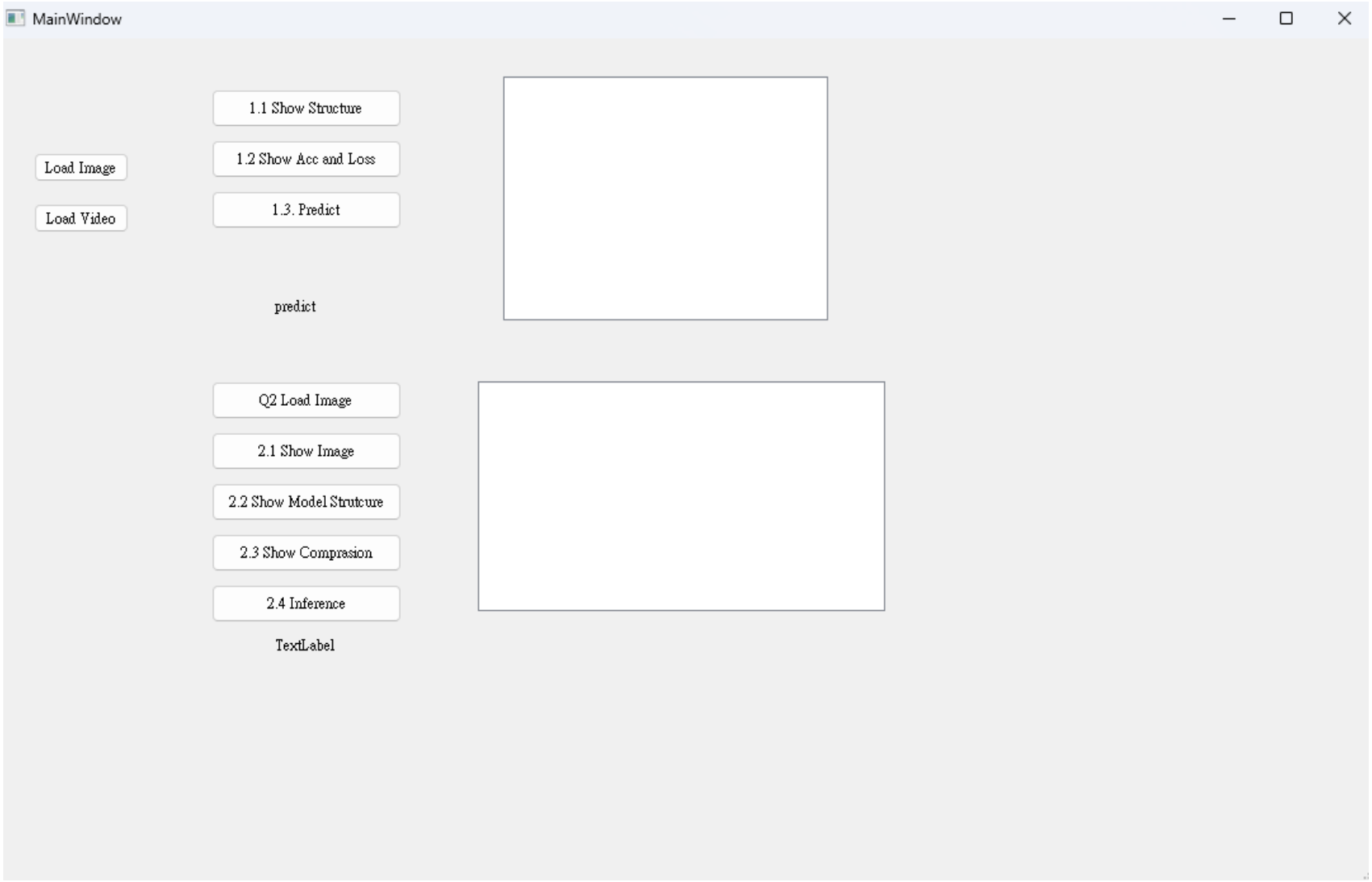
1. (50%) Training a MNIST Classifier using VGG16 with BN (出題：Allen)
   - 1.1 (15%) Load model and show model architecture.
   - 1.2 (15%) Show training/validating accuracy and loss.
   - 1.3 (20%) Use the model with highest validation accuracy to run inference, show the predicted distribution and class label.
2. (50%) Train a Cat-Dog Classifier using ResNet50 (出題：Kerwin)
   - 2.1 (10%) Load the dataset and resize images
   - 2.2 (10%) Show the architecture of ResNet50 model
   - 2.3 (15%) Improve ResNet50 with Random-Erasing and Compare the accuracies of 2 ResNet50 models on validation dataset
   - 2.4 (15%) 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 2 questions.



1. Training a MNIST Classifier Using VGG16 with BN (50%)

(出題：Allen)

1.1 Load model and show model architecture. (15%)

1.2 Show training/validating accuracy and loss. (15%)

1.3 Use the model with highest validation accuracy to run inference, show the probability distribution and class label. (20%)

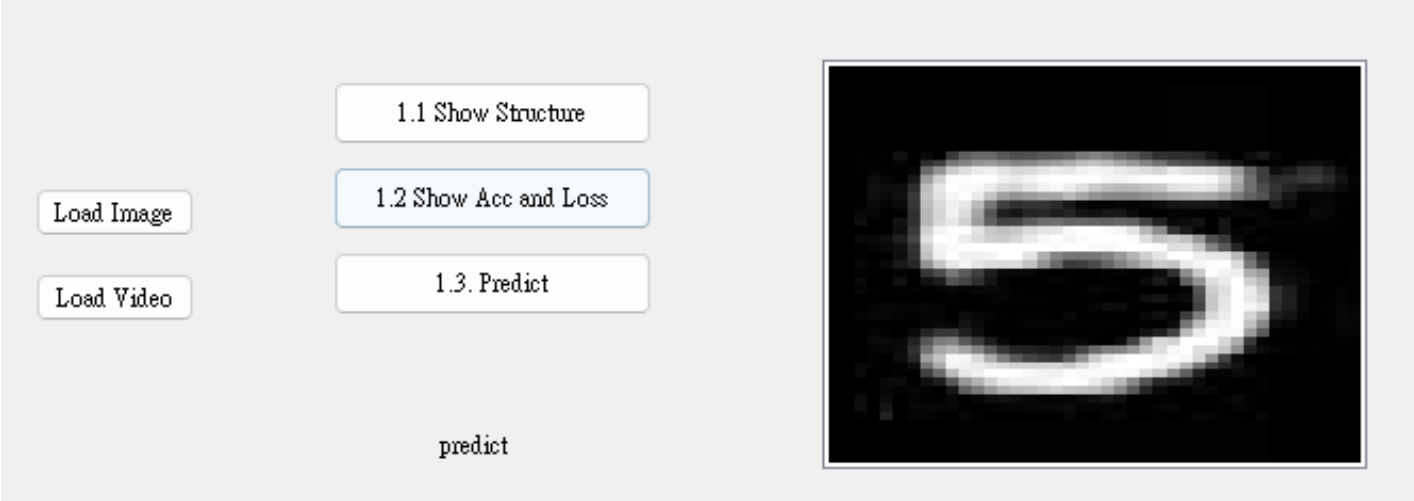
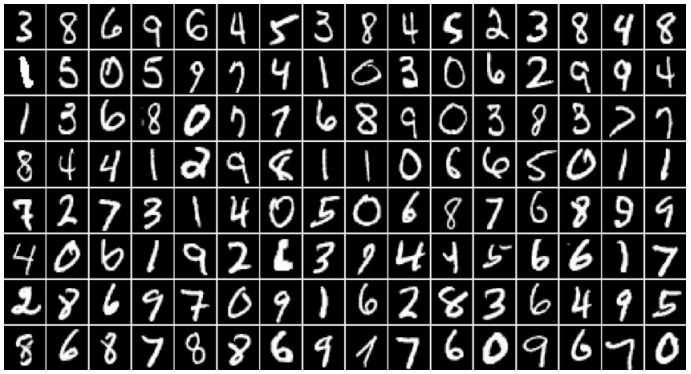


Figure: GUI example

1.0 Training a MNIST Classifier Using VGG16 with BN (50%)

■ Requirement (出題：Allen)

1) Train VGG16 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 from (28, 28) 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
   - 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

1.1 Show the architecture of VGG16 with BN (15%)

1. Click the button “1. Show Model architecture”
2. Show the VGG16 with BN model on terminal using torchsummary.summary().

(出題：Allen)

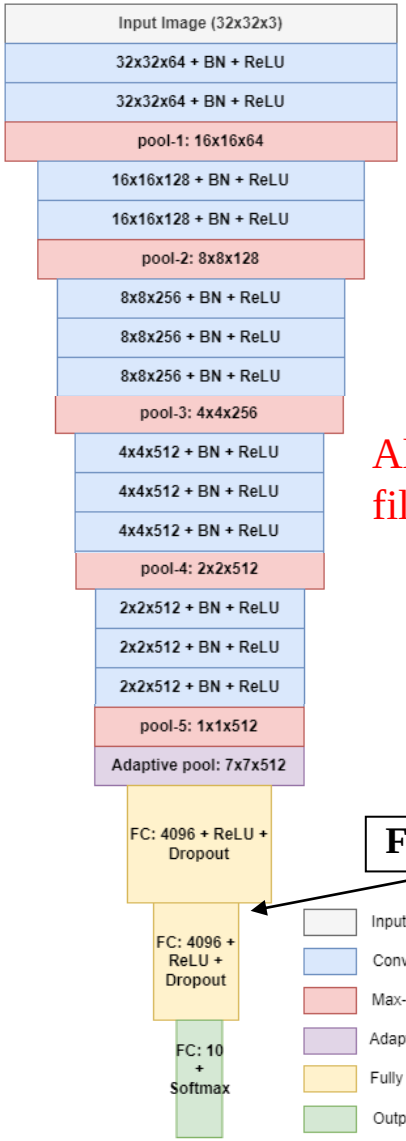


Figure: VGG16 with BN model architecture

All convolution filter size is 3x3

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

Layer (type)

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

Num. of paran

```
        MaxPool2d-34          [-1, 512, 2, 2]               0
            Conv2d-35          [-1, 512, 2, 2]       2,359,808
       BatchNorm2d-36          [-1, 512, 2, 2]           1,024
              ReLU-37          [-1, 512, 2, 2]               0
            Conv2d-38          [-1, 512, 2, 2]       2,359,808
       BatchNorm2d-39          [-1, 512, 2, 2]           1,024
              ReLU-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
         MaxPool2d-44          [-1, 512, 1, 1]               0
AdaptiveAvgPool2d-45          [-1, 512, 7, 7]               0
            Linear-46                [-1, 4096]     102,764,544
              ReLU-47                [-1, 4096]               0
           Dropout-48                [-1, 4096]               0
            Linear-49                [-1, 4096]      16,781,312
              ReLU-50                [-1, 4096]               0
           Dropout-51                [-1, 4096]               0
            Linear-52                  [-1, 10]          40,970
================================================================
Total params: 134,308,810
Trainable params: 119,586,826
Non-trainable params: 14,721,984
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 6.95
Params size (MB): 512.35
Estimated Total Size (MB): 519.30
----------------------------------------------------------------
```

1. Click the button.

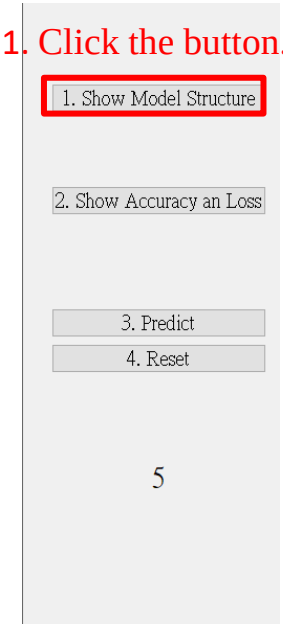


Figure: the architecture of VGG16 with BN

1.1 Show the architecture of VGG16 with BN (15%)

程式碼使用: (出題：Allen)

1. 使用VGG16

```
model = torchvision.models.vgg16_bn(pretrained=False)
```

2. 由於此作業使用灰階圖片, 所以須將3個通道修改成1個通道

```
model.features[0] = nn.Conv2d(1, 64, kernel_size=3, stride=1, padding=1)
```

3. 修改VGG16最後一層的輸出數量

```
model.classifier._modules['6'] = nn.Linear(model.classifier._modules['6'].in_features, num_class)
```

4. 顯示架構

```
torchsummary.summary(model, (1, 32, 32))
```

5. 下載MNIST 資料集，並將其設置為模型的訓練資料集

```
train_ds = datasets.MNIST(root='./data', train=True, transform=train_transform, download=True)
```

設定batch size

```
train_loader = DataLoader(train_ds, batch_size, shuffle=True)
```

6. 損失函數 (loss function)使用交叉熵損失函數 (Cross-Entropy Loss)

```
criterion = nn.CrossEntropyLoss(reduction='mean')
```

7. 資料預處理和增強操作

```
train_transform = transforms.Compose([
                  transforms.Resize((32, 32)),
                  transforms.RandomHorizontalFlip(),
                  transforms.RandomVerticalFlip(),
                  transforms.RandomRotation(degrees=15),
                  transforms.ToTensor(),
                  transforms.Normalize(mean=[0.5],std=[0.5])])
```

1.2 Show Training/Validating Accuracy and Loss (15%)

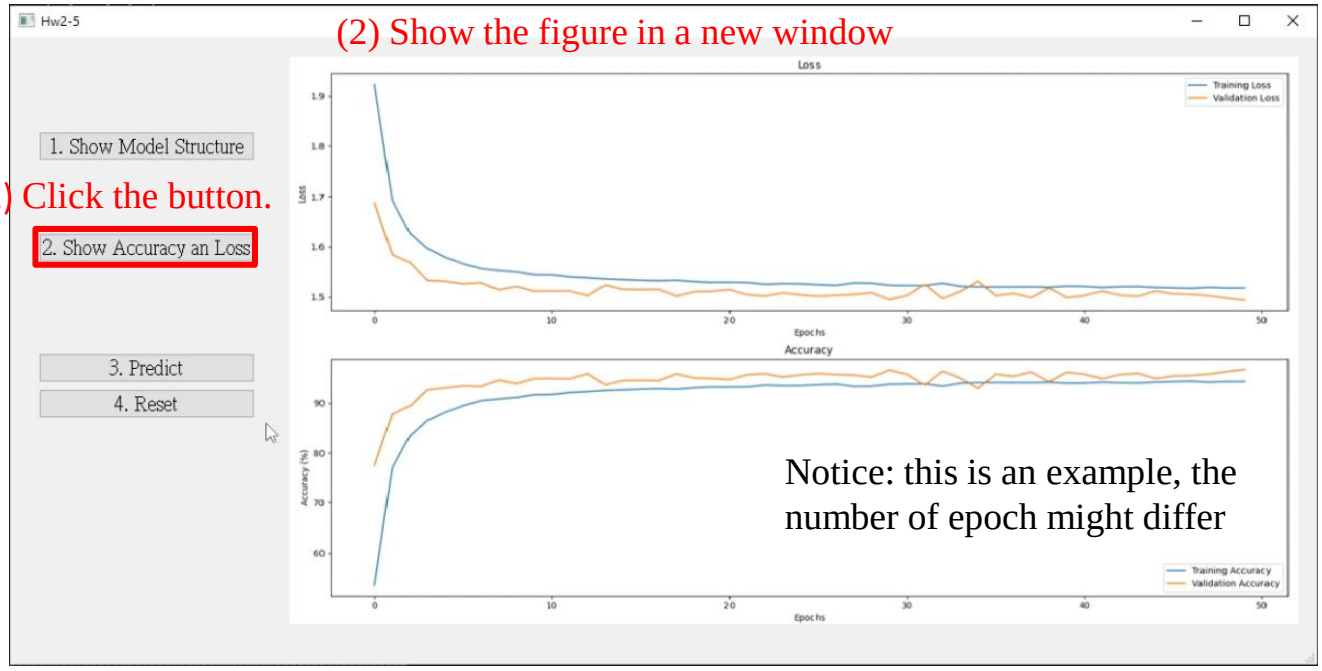
(出題：Allen)

1. At home:

1) Download the training and validation datasets.
2) Training and validating VGG16 with BN at least 30 epochs at home and record the training/validating accuracy and loss in each epoch .
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:

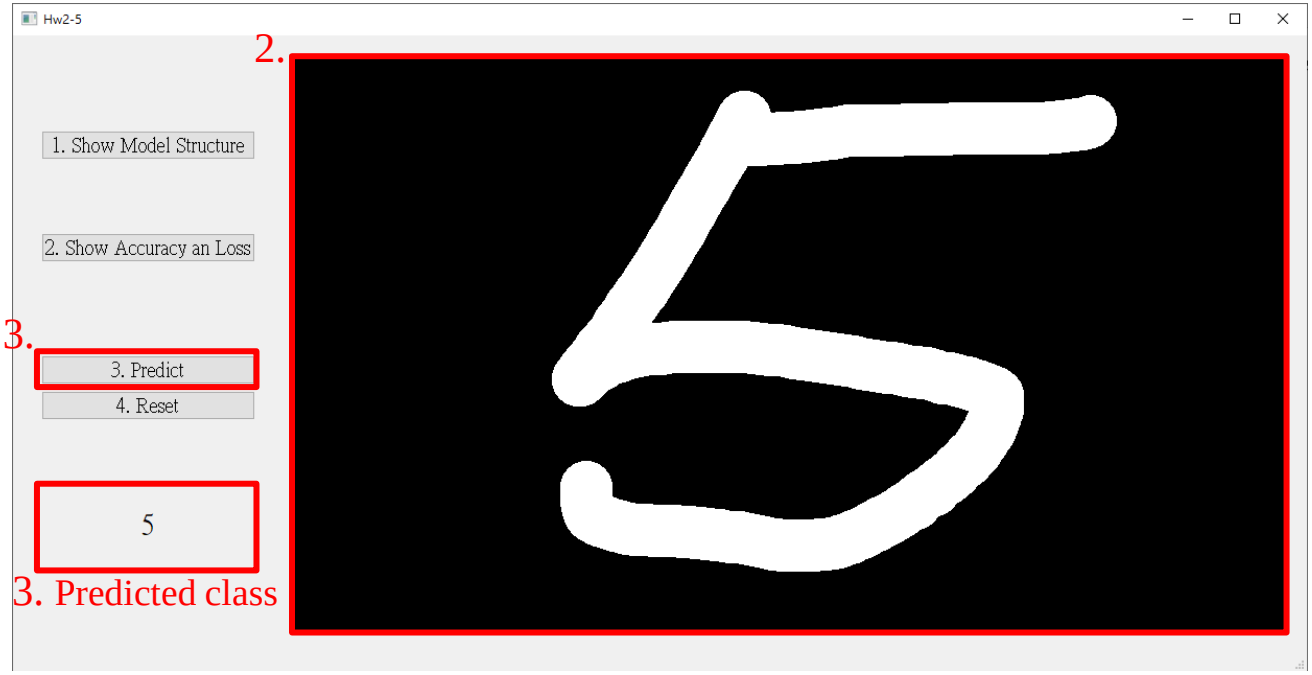
(1) Click the button "2. Show Accuracy and Loss"
(2) Show the saved figure of Training/Validating loss and accuracy in a new window



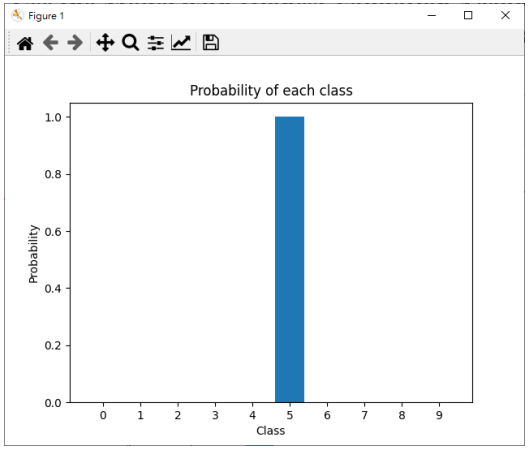
1.3 Use the Model with Highest Validation Accuracy to Run Inference, Show the Probability Distribution and Class Label. (20%)

(出題：Allen)

1. Load the model with highest validation accuracy which trained at home.
2. Load the image from 140.116.154.28 -> Download/02_Homework/Hw2/Q1_Dataset
3. Click the button "3. Predict" to run inference on the image you selected.
   - Show the predicted class label on the GUI.
   - Show the probability distribution of model predictions using a histogram in a new window.(不計分)



3. Probability of each class

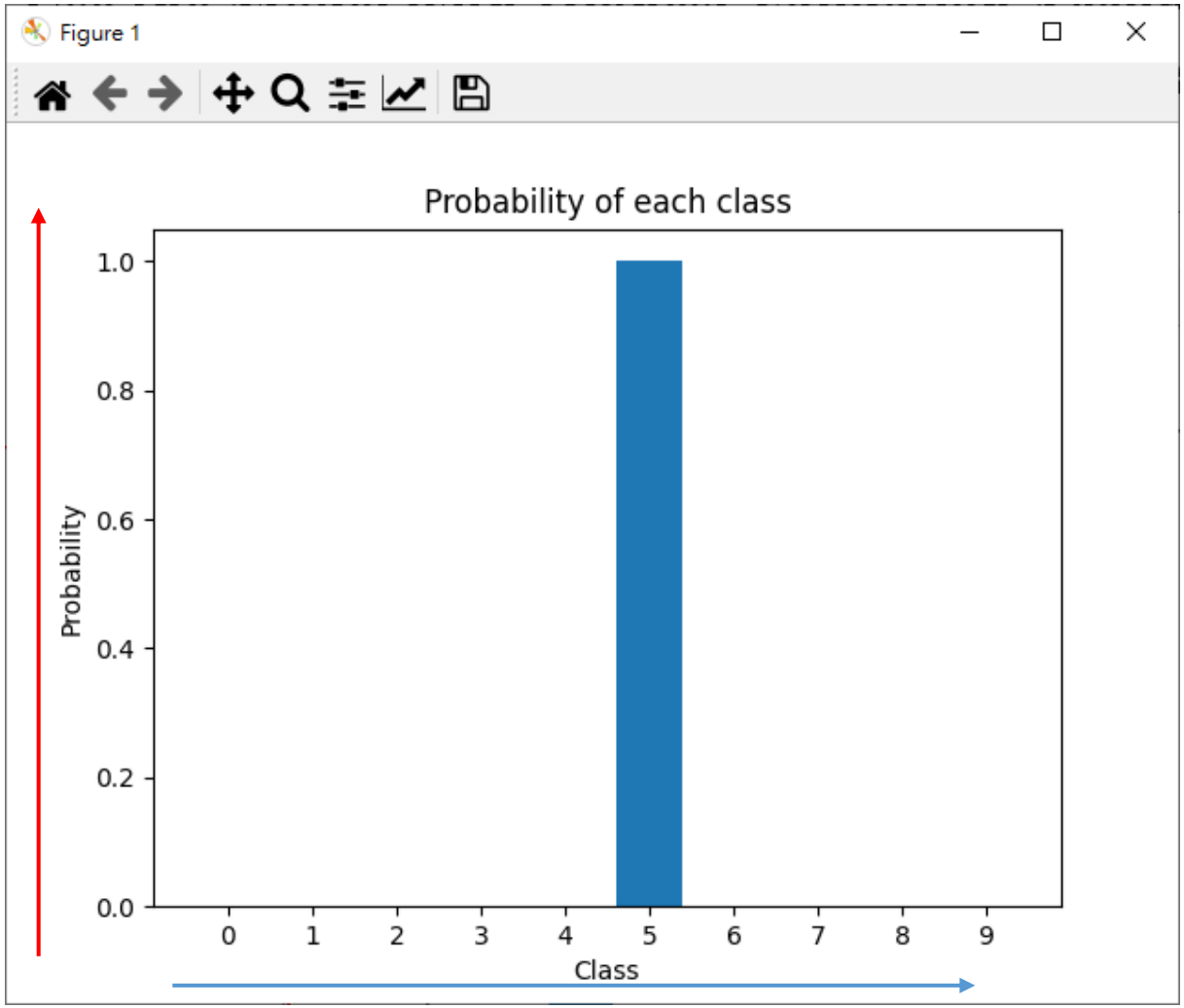


1.3 Use the Model with Highest Validation Accuracy to Run Inference, Show the Probability Distribution and Class Label. (20%)

(出題：Allen)

- The probability distribution of model prediction using a histogram.(不計分)

Y-axis: probability

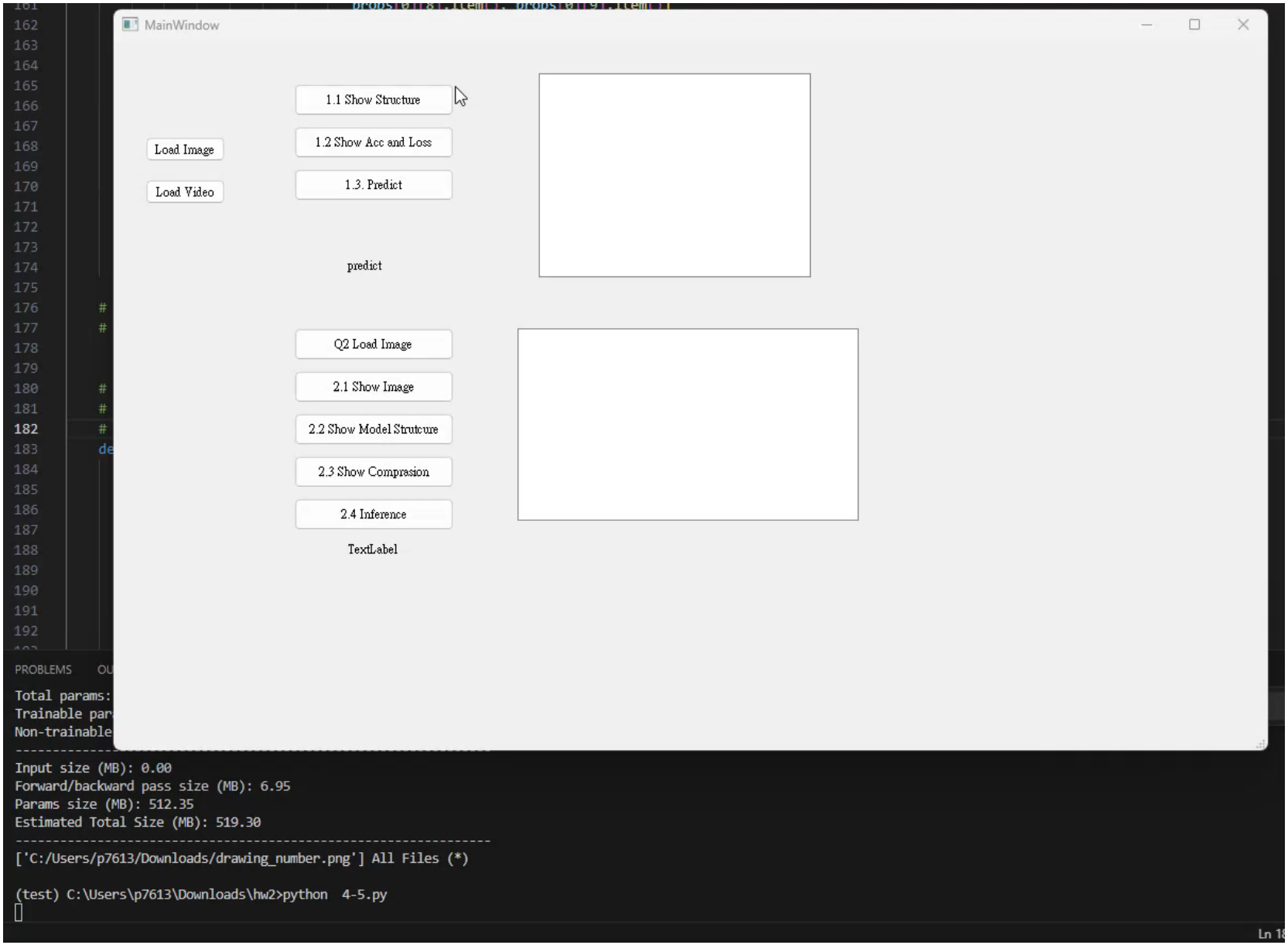


X-axis: class name

1. Training a MNIST Classifier Using VGG19 – Example Video

(出題：Allen)

- This is an example illustrating the objectives from 1.1 ~ 1.3.



2. Train a Cat-Dog classifier using ResNet50 (50%)

(出題：Kerwin)

1.1 (10%) Load the dataset and resize images

1.2 (10%) Show the architecture of ResNet50 model

1.3 (15%) Improve ResNet50 with Random-Erasing and compare the accuracies of 2 ResNet50 models on validation dataset

1.4 (15%) Use the trained model to run inference and show the predicted class label

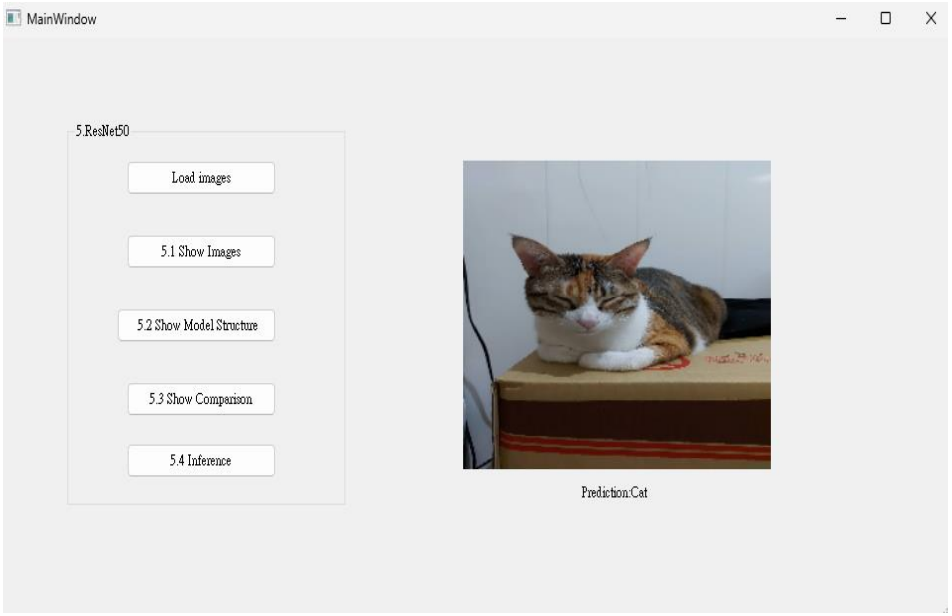


Figure: GUI Example

2.0 Train a Cat-Dog Classifier Using ResNet50

(出題：Kerwin)

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

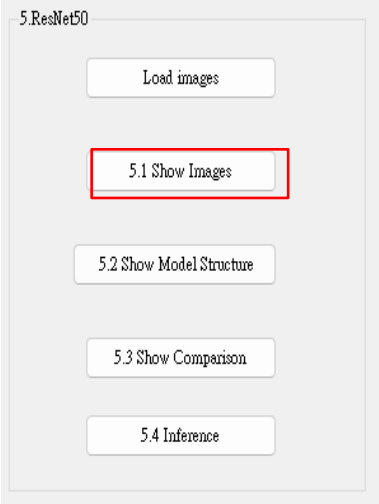
1) Deep Residual Learning for Image Recognition
2) Kaggle Cats and Dogs Dataset

2.1 (10%) Load the dataset and resize images

(出題：Kerwin)

1) In GUI:
   (1) Load the inference dataset
      → Hint:
         (a) PyTorch (tutorial): `torch.utils.data.Dataset`
            ➔ `You have to inherits this class with overriding the method __getitem__(self). In this method, you have to specify the load folders in order to get the inference dataset.`

   (2) Resize images to 224×224×3c (RGB)
   (3) Click the button "1. Show Images"
   (4) Show images with each class in the inference dataset in a new window



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

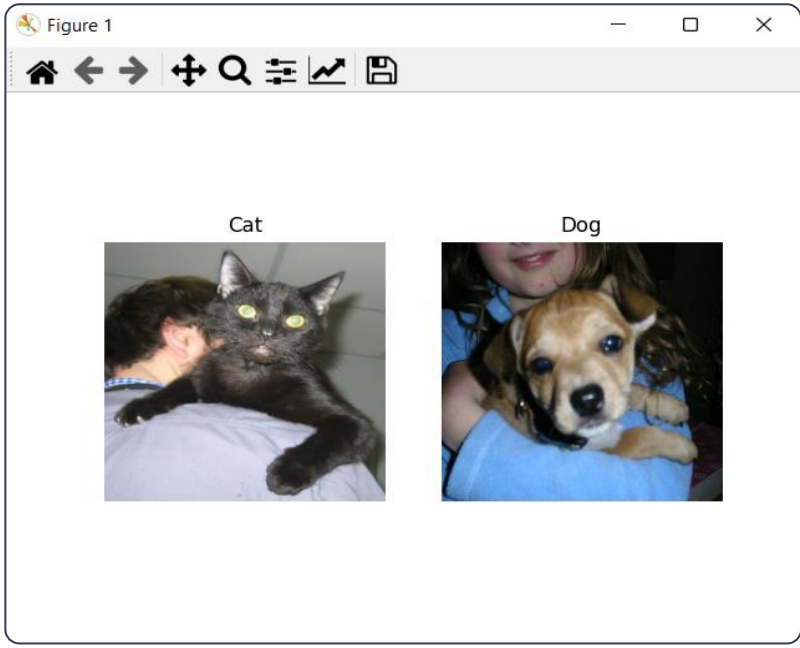


Figure: 1 image from each class
Notice: this is an example, the images might differ

2.2 (10%) Show the architecture of ResNet50 model (出題：Kerwin)

1) In GUI:
   (1) Build a ResNet50 model
   → Hint:
   (a) PyTorch:

O/P resnet50 = torchvision.models.resnet50() ➔ in order to load ResNet50 model.

O/P resnet50: the resnet50 model.

   (2) Replace the output layer to a FC (Fully Connected) layer of 2 node with a Softmax activation function

It's because that the node of FC Layer in ResNet50 is 1000.
In our assignment, it only needs 2 node to classify cat or dog.
As a result, we have to replace it.

O/P resnet.50.fc: the resnet50 FC Layer.
I/P resnet50.fc.in_features: original input size of FC Layer.
I/P torch.nn.Linear(resnet50.fc.in_features, 2): creating a FC node with original input size of FC Layer and output size of 2.
I/P torch.nn.Softmax(): softmax activation function.

   → Hint:
   (a) PyTorch (tutorial):

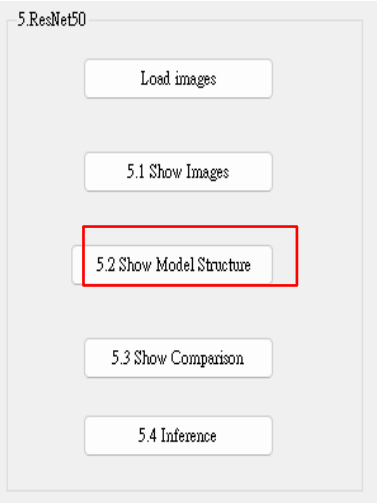
O/P resnet50.fc = torch.nn.Sequential(I/P torch.nn.Linear(resnet50.fc.in_features , 2), I/P torch.nn.Softmax()) ➔ in order to change the original FC Layer of model.

   (3) Run the function to show the structure in the terminal
   → Hint:
   (a) PyTorch:

I/P I/P
torchsummary(resnet50, (3, 224, 224)) ➔ to show the model structure on terminal.

I/P (3, 224, 224): input size of model.

2) When the demo:
   (1) Click the button "3. Show Model Structure"
   (2) Run the function to show the structure in the terminal



2.2 (10%) Show the architecture of ResNet50 model (出題：Kerwin)

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 (BatchNormalization)	(None, 56, 56, 64)	256	['conv2_block1_1_conv[0][0]']
conv2_block1_1_relu (Activation)	(None, 56, 56, 64)	0	['conv2_block1_1_bn[0][0]']
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36928	['conv2_block1_1_relu[0][0]']
conv2_block1_2_bn (BatchNormalization)	(None, 56, 56, 64)	256	['conv2_block1_2_conv[0][0]']
conv2_block1_2_relu (Activation)	(None, 56, 56, 64)	0	['conv2_block1_2_bn[0][0]']
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 (BatchNormalization)	(None, 56, 56, 256)	1024	['conv2_block1_0_conv[0][0]']
conv2_block1_3_bn (BatchNormalization)	(None, 56, 56, 256)	1024	['conv2_block1_3_conv[0][0]']

Figure: the structure of ResNet50 model

2.3 (15%) Improve ResNet50 with Random-Erasing (出題：Kerwin)

1) At GUI: Set up **Random-Erasing** in codes for model training (train.py)
   (1) Train 2 ResNet50 models with training dataset
   Data augmentation using random erasing in order to improve the accuracy.

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

Here is an example of four data augmentation: horizontal flip, vertical flip, center crop, random erasing.

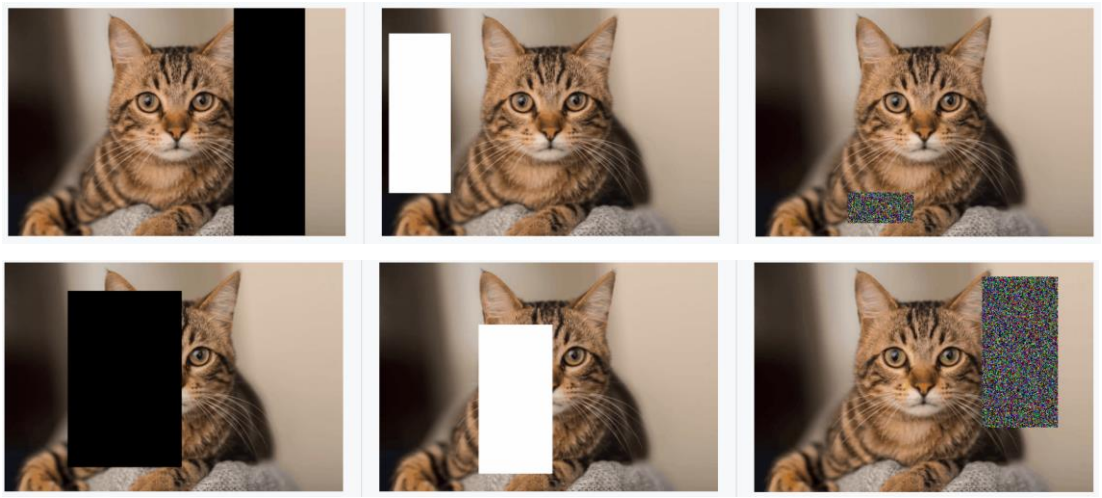


Figure1: Examples of the use of Random-Erasing

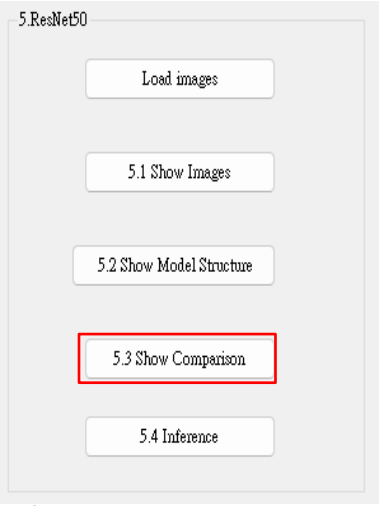
R. Reference

[Random Erasing Data Augmentation]

2.3 (15%) Compare the accuracies of 2 ResNet50 models on validation dataset

(出題：Kerwin)

1) At GUI:
   (1) Validate 2 different ResNet50 models with validation dataset
       first: without random erasing.
       second: with random erasing.
       → Hint:
         (a) PyTorch (tutorial): write a `for loop` to validate the model
             In a for loop, you have to predict different model result using validation dataset.
   (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



O/P

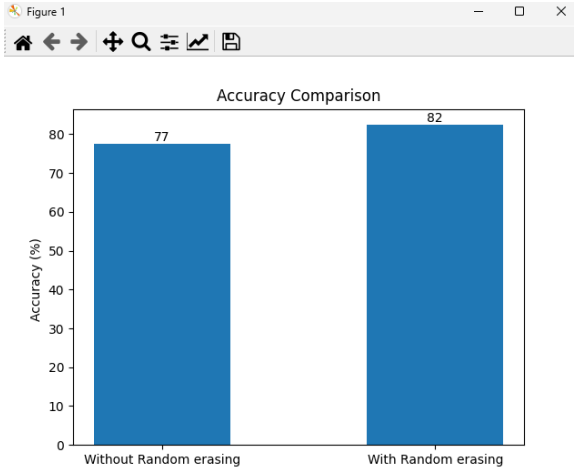


Figure1: Accuracy Comparison

Notice: this is an example, the numbers might differ

2.4 (15%) Use the better-trained model to run inference and show the predicted class label

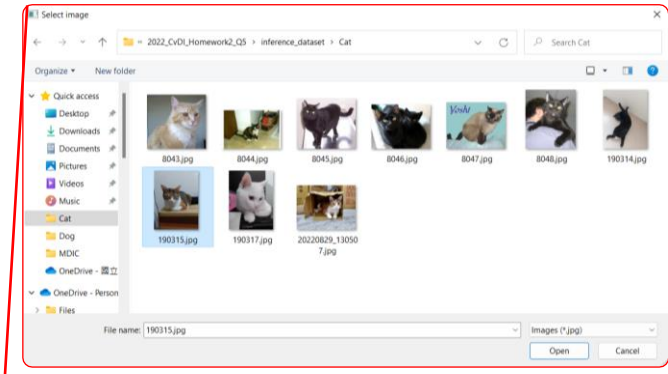
(出題：Kerwin)

1) At GUI:
   (1) Click the button "Load Image"
   (2) Select 1 image arbitrarily
   (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:

```
resnet50.eval() ➔ change model into evaluation
                  model in order to test the data.
O/P                                  I/P
predicted = resnet50(resized_img)
                  ➔ test the model with resized_img.
```

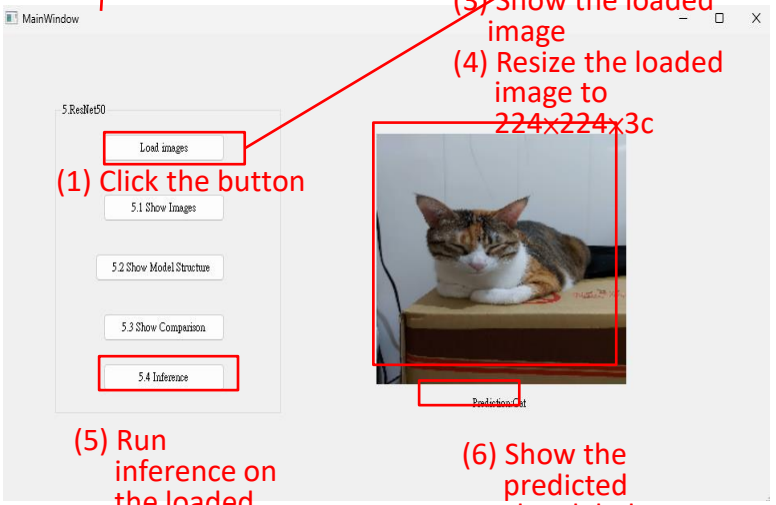
   (6) Show the predicted class label
   → Hint:
   Ex: predicted = [0.6, 0.4] ➔ cat
   predicted = [0.4, 0.6] ➔ dog
   predicted = [0.5, 0.5] ➔ cannot classify
2) When the demo: repeat the process

(2) Select 1 image arbitrarily



O/P predicted: the value of 0~1, which means the output of the model which test the source

I/P resized_img: (224x224x3c)



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

(出題：Kerwin)

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