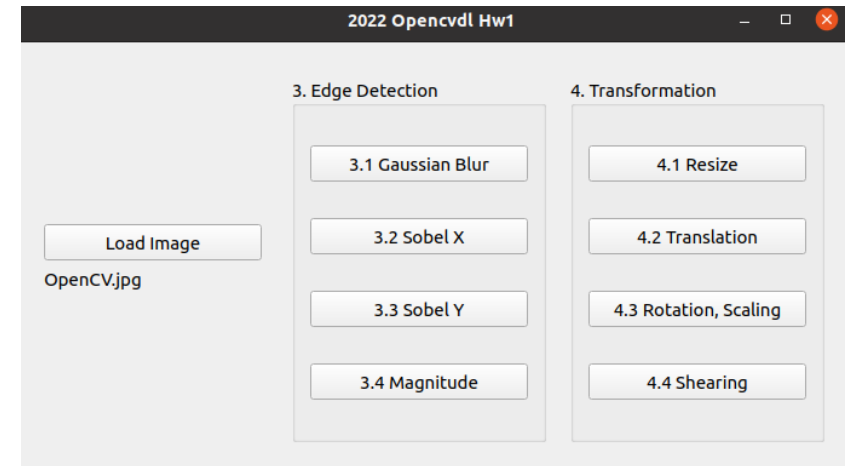


Notice (2/2)

- Python
 - Python 3.7 (<https://www.python.org/downloads/>)
 - opencv-contrib-python (3.4.2.17)
 - Matplotlib 3.1.1
 - UI framework: pyqt5 (5.15.1)
 - Pytorch
 - Tensorflow

Assignment scoring (Total: 100%)

1. (20%) Image Processing (出題 : Sam)
 - 1.1 (5%) Color Separation
 - 1.2 (5%) Color Transformation
 - 1.3 (5%) Color Detection
 - 1.4 (5%) Blending
2. (20%) Image Smoothing (出題 : Jack)
 - 2.1 (6%) Gaussian blur
 - 2.2 (7%) Bilateral filter
 - 2.3 (7%) Median filter
3. (20%) Edge Detection (出題 : Chong)
 - 3.1 (5%) Gaussian Blur
 - 3.2 (5%) Sobel X
 - 3.3 (5%) Sobel Y
 - 3.4 (5%) Magnitude
4. (20%) Transforms (出題 : Jeffin)
 - 4.1 (5%) Resize
 - 4.2 (5%) Translation
 - 4.3 (5%) Rotation, Scaling
 - 4.4 (5%) Shearing
5. (20%) Training Cifar10 Classifier Using VGG19 (出題 : Wen)
 - 5.1 (4%) Load Cifar10 and Random Show 9 Images with Label
 - 5.2 (4%) Load Model and Show Model Structure
 - 5.3 (4%) Show Data Augmentation Result
 - 5.4 (4%) Show Accuracy and Loss
 - 5.5 (4%) Inference



5. Training Cifar10 Classifier Using VGG19 (20%)

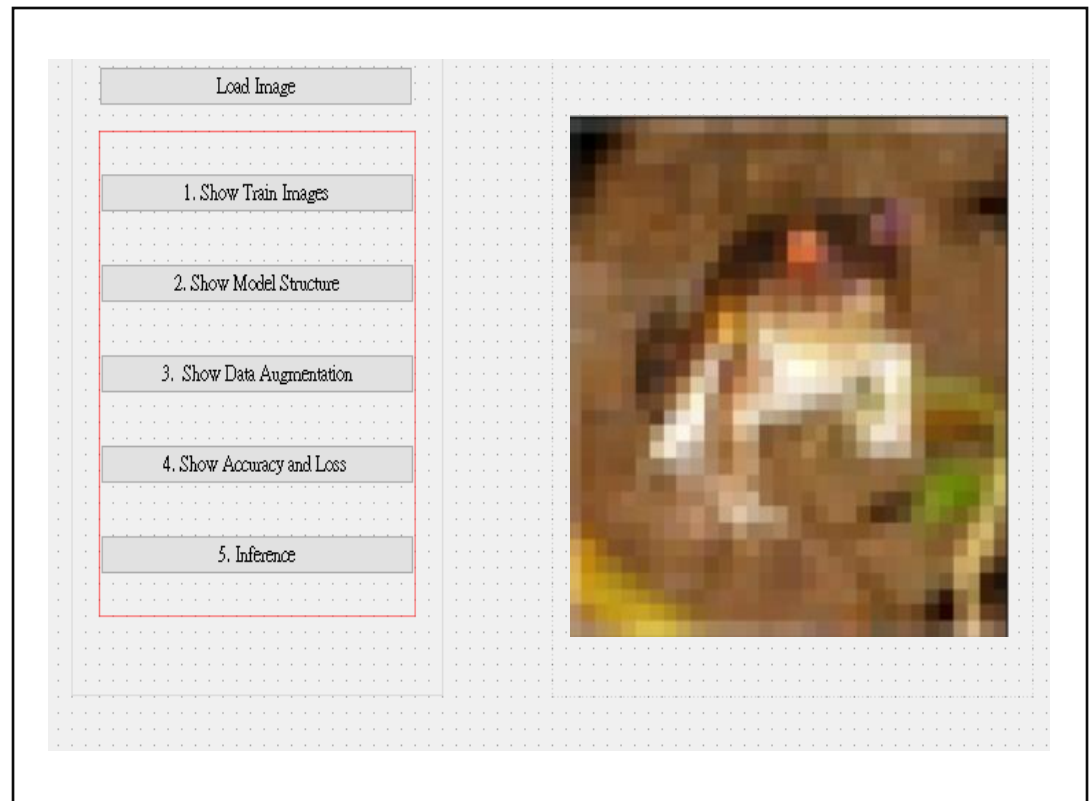
5.1 Load Cifar10 and Random Show 9 Images with Label(4%)

5.2 Load Model and Show Model Structure (4%)

5.3 Show Data Augmentation Result (4%)

5.4 Show Accuracy and Loss (4%)

5.5 Inference (4%)



5.1 Load Cifar10 **training dataset**, and then **Randomly show 9 Images** and **Labels** respectively (4%)

1. Load Cifar10.

1)Tensorflow: `tf.keras.datasets.cifar10.load_data()`

10 Class of CIFAR 10 Dataset

0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

2. Click Button to Random show 9 Images with Labels.



Figure 1

Class
Label
Image



◆ Hint

Use Matplotlib 4 function

1. `figure()`
2. `title()`
3. `Axis()`
4. `imshow()`

- Can refer by tutorial at the Matplotlib library official web-site

<https://matplotlib.org/stable/tutorials/index.html>

5.2 Load Model and Show Model Structure(4%)

1. Load Model

When Training:

1)Tensorflow: `tf.keras.applications.VGG19()`

When Demo:

1)Tensorflow: `tf.keras.models.load_model(model_name)`

◆ Hint(when call model)

1. Input shape should set to 32 x 32.
2. Classes should to set 10.

2. Click Button to Show Model Structure on terminal.



```
ReLU-21      [-1, 512, 4, 4]      0
Conv2d-22     [-1, 512, 4, 4]      2,359,808
ReLU-23      [-1, 512, 4, 4]      0
Conv2d-24     [-1, 512, 4, 4]      2,359,808
ReLU-25      [-1, 512, 4, 4]      0
Conv2d-26     [-1, 512, 4, 4]      2,359,808
ReLU-27      [-1, 512, 4, 4]      0
MaxPool2d-28  [-1, 512, 2, 2]      0
Conv2d-29     [-1, 512, 2, 2]      2,359,808
ReLU-30      [-1, 512, 2, 2]      0
Conv2d-31     [-1, 512, 2, 2]      2,359,808
ReLU-32      [-1, 512, 2, 2]      0
Conv2d-33     [-1, 512, 2, 2]      2,359,808
ReLU-34      [-1, 512, 2, 2]      0
Conv2d-35     [-1, 512, 2, 2]      2,359,808
ReLU-36      [-1, 512, 2, 2]      0
MaxPool2d-37  [-1, 512, 1, 1]      0
AdaptiveAvgPool2d-38 [-1, 512, 7, 7]      0
Linear-39     [-1, 4096]      102,764,544
ReLU-40      [-1, 4096]      0
Dropout-41    [-1, 4096]      0
Linear-42     [-1, 4096]      16,781,312
ReLU-43      [-1, 4096]      0
Dropout-44    [-1, 4096]      0
Linear-45     [-1, 1000]     4,097,000
=====
Total params: 143,667,240
Trainable params: 143,667,240
Non-trainable params: 0
-----
Input size (MB): 0.01
Forward/backward pass size (MB): 5.25
Params size (MB): 548.05
Estimated Total Size (MB): 553.31
-----
```

◆ Hint

Pytorch API

Use the two option

1) Summary function

from torchsummary import summary
summary(Model name, (Input Channel,
Input Width, Input Height))

2) Print function

From torchvision import models
model = torchvision.models.vgg19()
Print(Model)

- Can refer this web-site <https://pypi.org/project/pytorch-model-summary/>

5.2 Load Model and Show Model Structure(4%)

Batch size, channel, H,W

ReLU-21	[-1, 512, 4, 4]	0
Conv2d-22	[-1, 512, 4, 4]	2,359,808
ReLU-23	[-1, 512, 4, 4]	0
Conv2d-24	[-1, 512, 4, 4]	2,359,808
ReLU-25	[-1, 512, 4, 4]	0
Conv2d-26	[-1, 512, 4, 4]	2,359,808
ReLU-27	[-1, 512, 4, 4]	0
MaxPool2d-28	[-1, 512, 2, 2]	0
Conv2d-29	[-1, 512, 2, 2]	2,359,808
ReLU-30	[-1, 512, 2, 2]	0
Conv2d-31	[-1, 512, 2, 2]	2,359,808
ReLU-32	[-1, 512, 2, 2]	0
Conv2d-33	[-1, 512, 2, 2]	2,359,808
ReLU-34	[-1, 512, 2, 2]	0
Conv2d-35	[-1, 512, 2, 2]	2,359,808
ReLU-36	[-1, 512, 2, 2]	0
MaxPool2d-37	[-1, 512, 1, 1]	0
AdaptiveAvgPool2d-38	[-1, 512, 7, 7]	0
Linear-39	[-1, 4096]	102,764,544
ReLU-40	[-1, 4096]	0
Dropout-41	[-1, 4096]	0
Linear-42	[-1, 4096]	16,781,312
ReLU-43	[-1, 4096]	0
Dropout-44	[-1, 4096]	0
Linear-45	[-1, 1000]	4,097,000
=====		
Total params:	143,667,240	
Trainable params:	143,667,240	
Non-trainable params:	0	

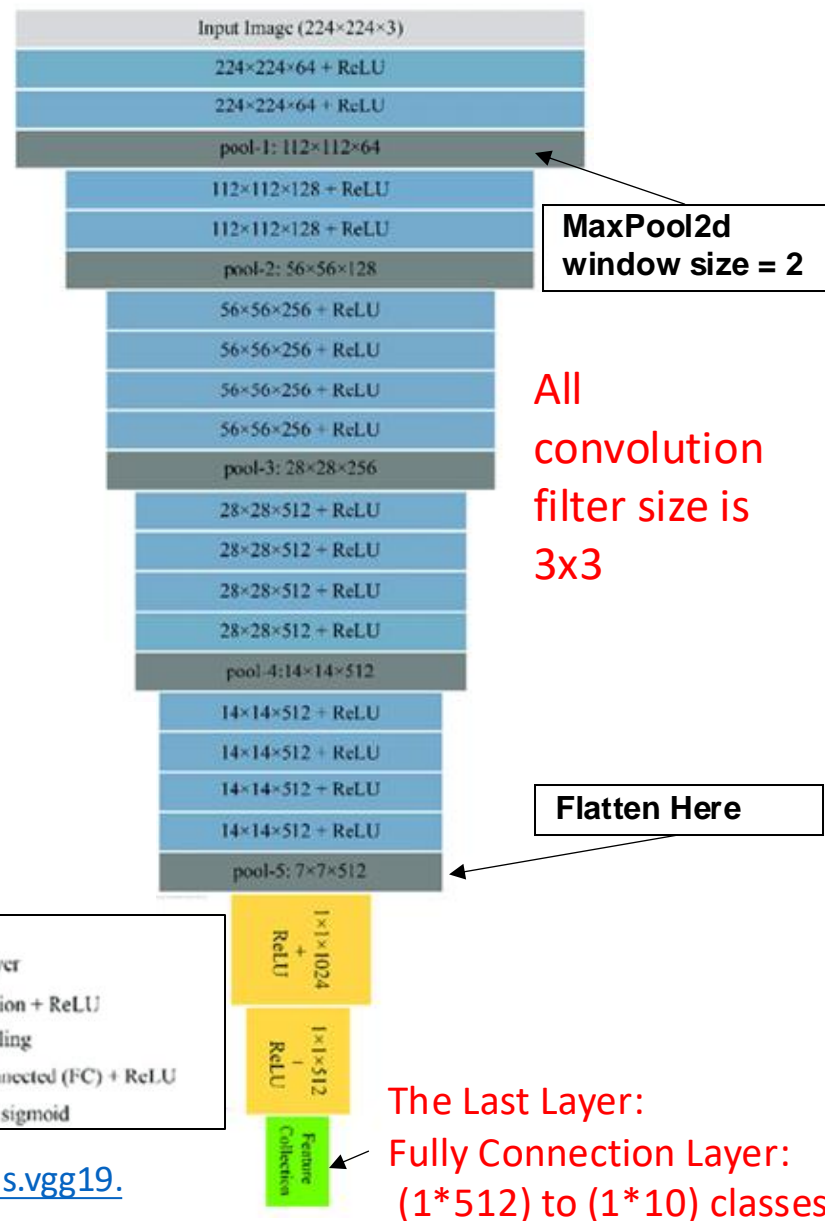
Input size (MB):	0.01	
Forward/backward pass size (MB):	5.25	
Params size (MB):	548.05	
Estimated Total Size (MB):	553.31	

Layers List of Model

After processing each layer
Change of input data type

Number of trainable parameters

Resize Image
32x32 to 224x224



VGG19 framework

1. Reference

- 1) <https://pytorch.org/vision/0.12/generated/torchvision.models.vgg19.html>(Source Code)
- 2) <https://www.cs.toronto.edu/~kriz/cifar.html> (Cifar10 Dataset)

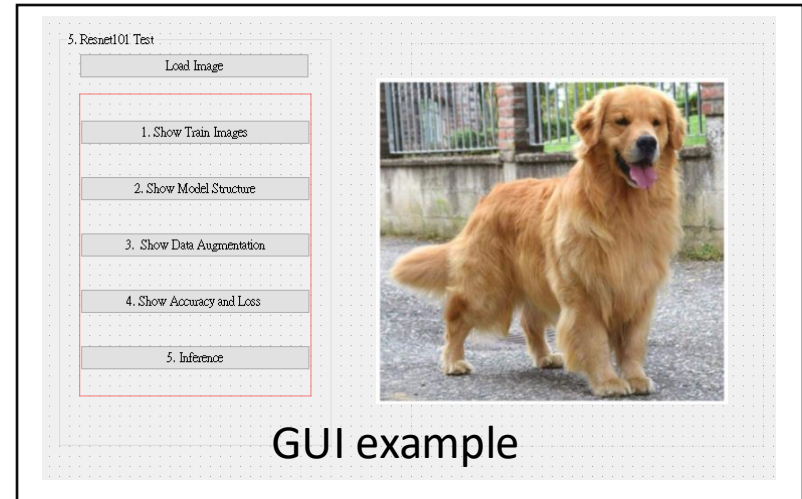
5.3 Show Data Augmentation Result (4%)

- When conducting deep learning training, we often need **massive amounts of data to ensure that there is no over-fitting** during training.
- **Data augmentation** is to modify and deform the existing pictures in the dataset to create more pictures for machine learning to **make up for the lack of data**.
- In this section, we want you to try 3 kinds of augmentation methods.

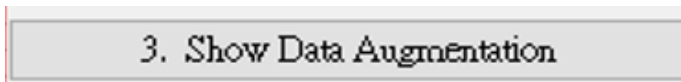
❑ The methods of transforms on PIL Image and torch.*Tensor	
CenterCrop (size)	Crops the given image at the center.
ColorJitter ([brightness, contrast, ...])	Randomly change the brightness, contrast, saturation and hue of an image.
FiveCrop (size)	Crop the given image into four corners and the central crop.
Grayscale ([num_output_channels])	Convert image to grayscale.
Pad (padding[, fill, padding_mode])	Pad the given image on all sides with the given “pad” value.
RandomAffine (degrees[, translate, scale, ...])	Random affine transformation of the image keeping center invariant.
RandomApply (transforms[, p])	Apply randomly a list of transformations with a given probability.
RandomCrop (size[, padding, pad_if_needed, ...])	Crop the given image at a random location.
RandomGrayscale ([p])	Randomly convert image to grayscale with a probability of p (default 0.1).
RandomHorizontalFlip ([p])	Horizontally flip the given image randomly with a given probability.
RandomPerspective ([distortion_scale, p, ...])	Performs a random perspective transformation of the given image with a given probability.

5.3 Show Data Augmentation Result (4%)

1. Load Image to select an image file.



2. Click Button to Show Augmentation Data. Concatenate 3 figures of the augmentation results and show it.



Transforms on PIL Image and torch.*Tensor



Function: [RandomRotation\(\)](#) [RandomResizedCrop\(\)](#) [RandomHorizontalFlip\(\)](#)

- Can refer by tutorial at the Matplotlib library official web-site
<https://matplotlib.org/stable/tutorials/index.html>

5.4 Training your model at least **30 epochs at home** and **record the accuracy and loss in each epoch**. Show the Figure **of your training loss and accuracy when demo time**. (4%)

Do in your home:

1) tensorflow: model.fit()

You should set the parameter:

1) Epoch to 30

2) Loss : CrossEntropyLoss

3) Optimizer: Adam

Show the accuracy and loss via plt and get a screen shoot to store this image.

◆ Hint

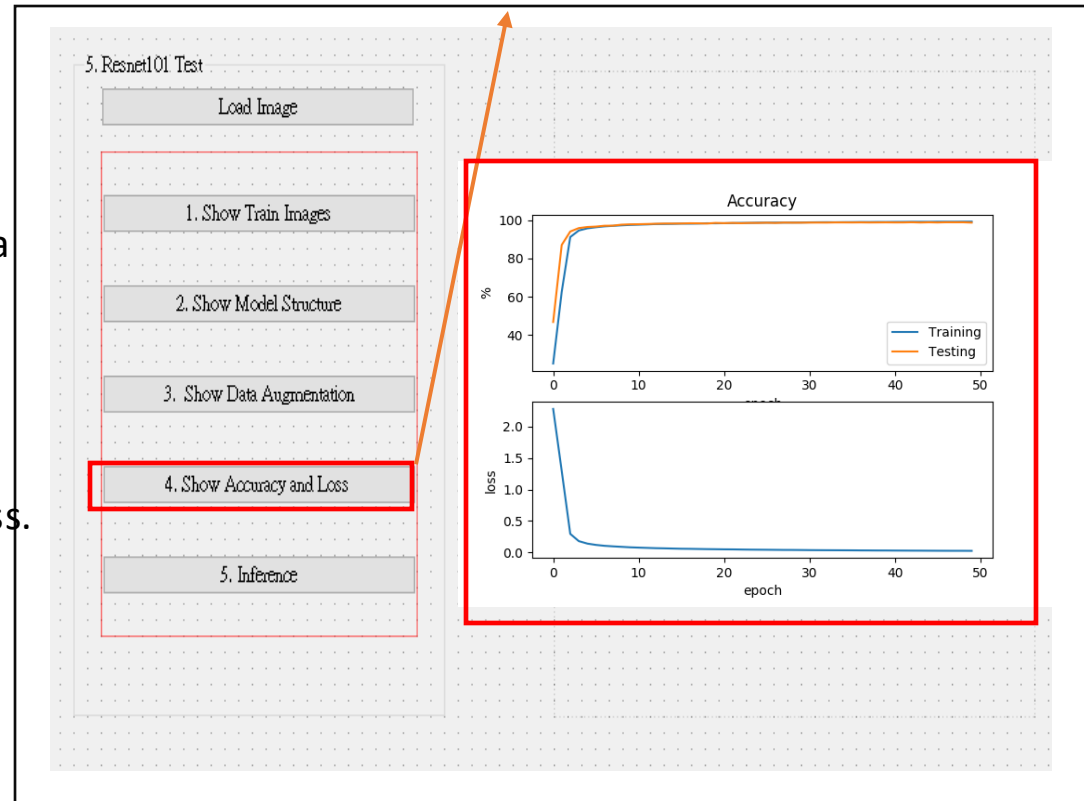
1. model which is called form 5.2.
2. model.fit() returns the accuracy and loss.
3. model.save() to save your model.

1. **Please save your weighted file during training**
2. **Record each training result (accuracy, loss) and draw it as the picture on the right**

When Demo :

1. Click button to the image

4. Show Accuracy and Loss

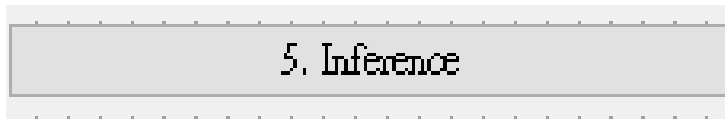


5.5 Load the image and inference them with your weighted file (.pth), and show the result image and class (4%)

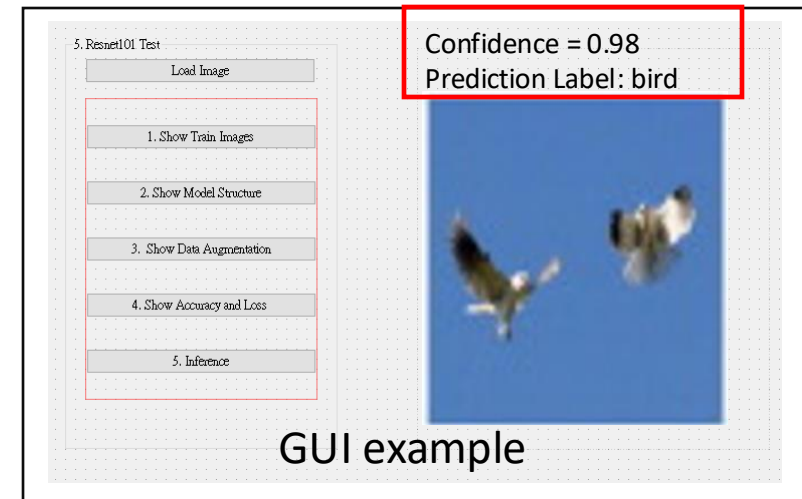
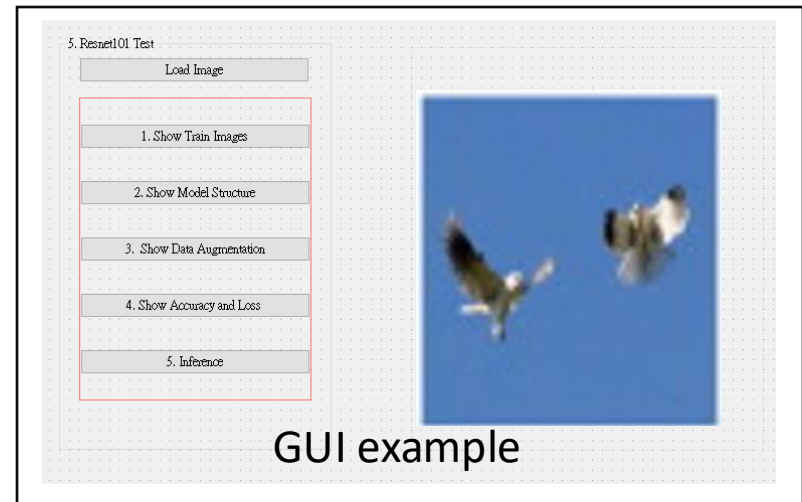
1. Choose the any data



2. Run Inference



1. Using the model.predict() to classification
2. Show the result on the UI
3. Show the prediction label, confidence score



Hint

1. <https://towardsdatascience.com/understanding-pytorch-with-an-example-a-step-by-step-tutorial-81fc5f8c4e8e#5017>
2. <https://pytorch.org/tutorials/>
3. <https://yanwei-liu.medium.com/pytorch-with-grad-cam-6a92a54bfaad>