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 1

TA:

少鈞: 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: Score will be zero for both persons!!
- Due date => 08:00:00, 2022/10/31 (Mon.)

  No delay. Penalties for late homework:
  - ➢ Up to 7 days late, loss of 50% of the score awarded
  - ➢ After 7 days, the score will be marked as 0.
- You must attend the demonstration, or your score will be 0. The demonstration schedule will be announced on NCKU moodle.
- You must make a GUI and follow the format, or you will get some penalties.
- Upload to => 140.116.154.1 -> Upload/Homework/Hw1_2

  Upload/Homework/Hw1_05
  - ➢ User ID: opencvdl2022 Password: opencvdl2022
- Format
  - ➢ Filename: Hw1_2_StudentID_Name_Version.rar

    Hw1_05_StudentID_Name_Version.rar
    - Ex: Hw1_2_F71234567_林小明_V1.rar
    - If you want to update your file, you should update your version to be V2. Ex: Hw1_2_F71234567_林小明_V2.rar
  - ➢ Content: project folder*(excluding the pictures, only source code)

    *note: remove your “Debug” folder to reduce file size

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

3. Edge Detection (20%)

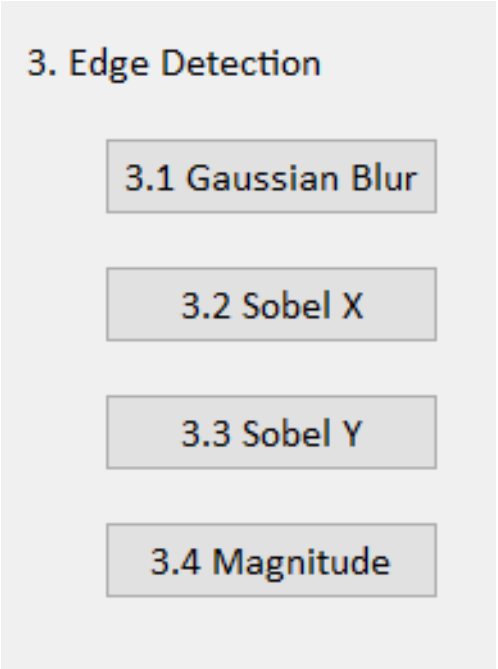
(出題: Chong)

3.1 Gaussian Blur (5%)

3.2 Sobel X (5%)

3.3 Sobel Y (5%)

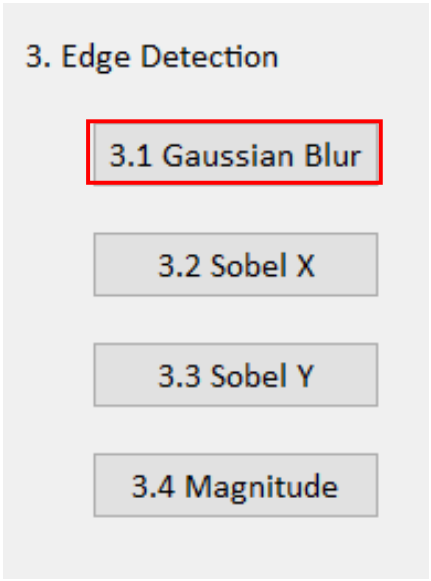
3.4 Magnitude (5%)



3.1 Gaussian Blur (5%)

(出題：Chong)

- Given: a RGB image, “building.jpg”
- Q: 1) **Gaussian Blur**: Convert the RGB image into a grayscale image, then smooth it by your own 3x3 Gaussian smoothing filter (Can not use OpenCV Function, Sobel, GaussianBlur and conv2d). Please show the result.



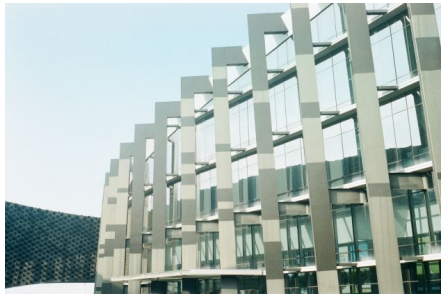
- Hint: Textbook Chapter 5, p.109 ~ p.114

  How to generate Gaussian Filter:

  ① Let $G_{init}(x,y) = \begin{bmatrix} (-1,-1) & (0,-1) & (1,-1) \\ (-1,0) & (0,0) & (1,0) \\ (-1,1) & (0,1) & (1,1) \end{bmatrix}$

  ② Calculate $G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$, $\sigma = \sqrt{0.5}$

  ③ Normalize $G(x,y), G_{norm}(x,y) = \begin{bmatrix} 0.045 & 0.122 & 0.045 \\ 0.122 & 0.332 & 0.122 \\ 0.045 & 0.122 & 0.045 \end{bmatrix}$



building.jpg

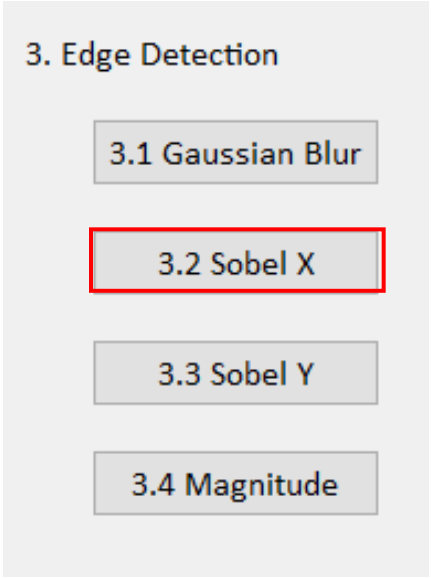
Grayscale

Gaussian Blur

3.2 Sobel X (5%)

(出題：Chong)

- Given: the result of 3.1) Gaussian Blur
- Q: 2)**Sobel X**: Use Sobel edge detection to detect vertical edge by your own 3x3 Sobel X operator (Can not use OpenCV Function, Sobel, GaussianBlur and conv2d). Please show the result.
- Hint: Textbook Chapter 6, p.148 ~ 149



Gaussian Blur

-1	0	1
-2	0	2
-1	0	1

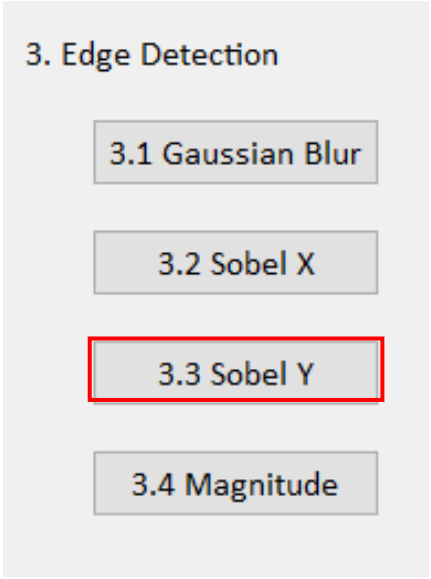
Sobel X Filter

Sobel X

3.3 Sobel Y (5%)

(出題：Chong)

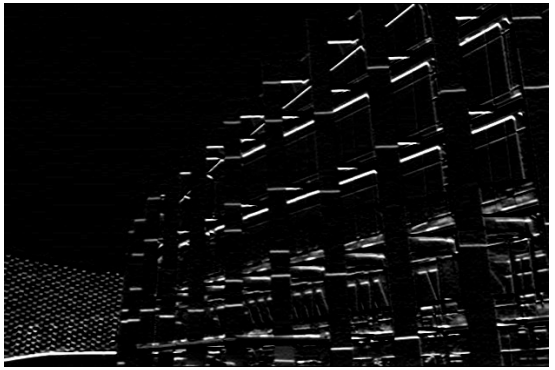
- Given: the result of 3.1) Gaussian Blur
- Q: 3)**Sobel Y**: Use Sobel edge detection to detect horizontal edge by your own 3x3 Sobel Y operator (Can not use OpenCV Function, Sobel, GaussianBlur and conv2d). Please show the result.
- Hint: Textbook Chapter 6, p.148 ~ 149



Gaussian Blur

1	2	1
0	0	0
-1	-2	-1

Sobel Y Filter



Sobel Y

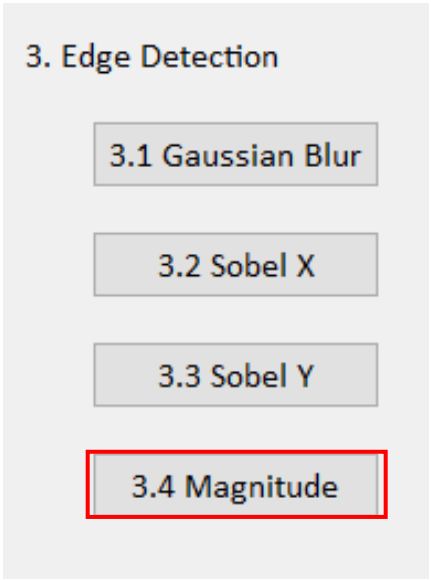
3.4 Magnitude (5%)

(出題：Chong)

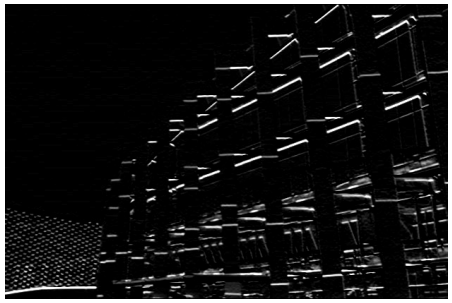
- Given: the result of 3.2) Sobel X and 3.3) Sobel Y
- Q: 4) **Magnitude**: Use the results of 3.2) Sobel X and 3.3) Sobel Y to calculate the magnitude. Please show the result.
- Hint: Textbook Chapter 6, p.148 ~ 149

$$\text{Magnitude} = \sqrt{\|Sobel_X{}^2 + Sobel_Y{}^2\|}$$

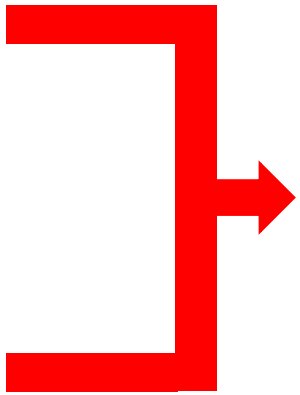
Normalize the result to 0~255.



Sobel X



Sobel Y

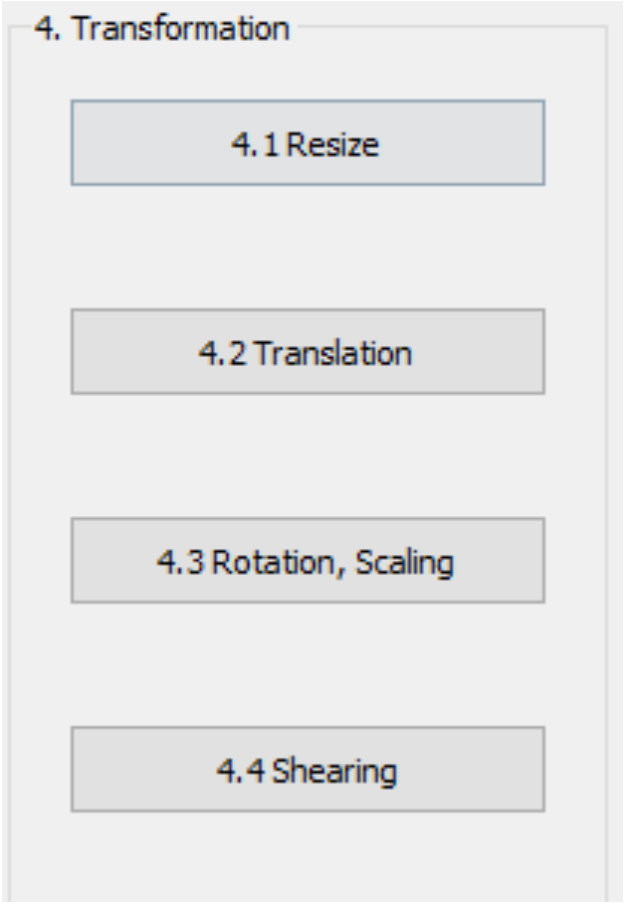


Magnitude

4. Transforms (20%)

(出題: Jeffin)

- 4.1 Resize (5%)
- 4.2 Translation(5%)
- 4.3 Rotation, Scaling (5%)
- 4.4 Shearing (5%)



4.1 Transforms: Resize, Translation, Rotation, Scaling, Shearing(20%)

- Given: "*Microsoft.png*", Image Size (430, 430)

(出題: Jeffin)

- Q:Please resize, translate , rotate, scale and shearing the picture (*Microsoft.png*)

  4.1) Resize:
  From (430,430) to (215,215)
  and cv2.imshow with (430, 430) window (image center: (108, 108)
  top left of window)

  4.2) Image Translation:
  Xnew = Xold + 215 pixels = 108 + 215 = 323
  Ynew = Yold + 215 pixels = 108 + 215 = 323
  Point C (108, 108) is center of resized image
  Point C'(323, 323) is new center of image (bottom right of window)
  (Then overlay with *result image of 4.1)*

  4.3) Rotation, Scaling:
  Center: Center of Image
  Angle = 45° (counter-clockwise)
  Scale = 0.5 , window size (430,430)

  4.4) Shearing:
  Old location: ([[50,50],[200,50],[50,200]])
  New location: ([[10,100],[100,50],[100,250]])
  (Note: Please save your image after each section)

- Hint: Textbook Chapter 12, (p.407 ~ 412)
  python: cv2.warpAffine(),
  Textbook Chapter 3, (p.50 ~ 52)
  cv2.addWeighted()

W:430

H:430

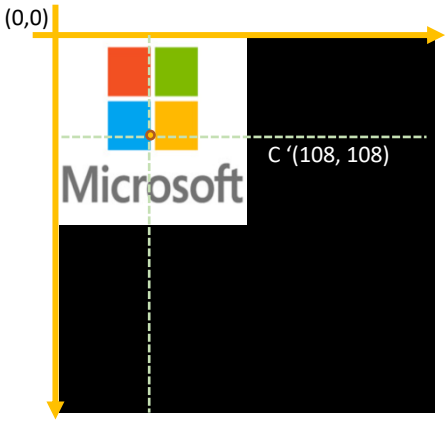
Microsoft.png

4.2 Transforms: Resize, Translation, Rotation, Scaling, Shearing(20%)

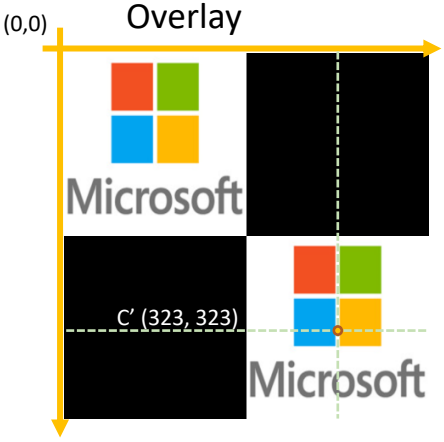
- EX: Given: “Microsoft.*png*”, image size (430, 430) (出題: Jeffin)

Microsoft.png

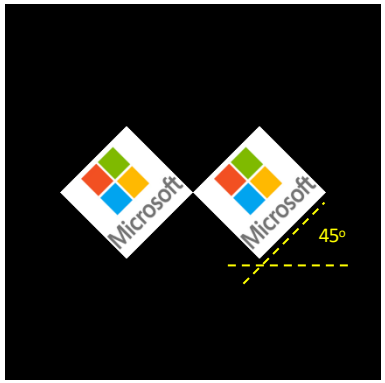
4.1) Resized Image



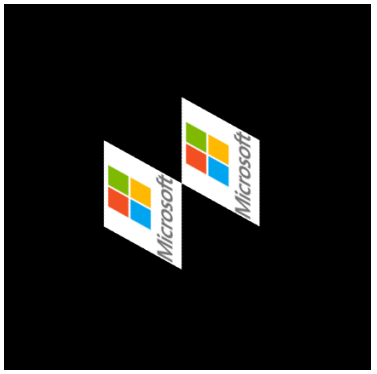
4.2) Translate + Overlay



4.3) Rotate and Scale



4.4) Shearing



- Hint: Textbook Chapter 12, (p.407 ~ 412)
  python: cv2.warpAffine(),
  Textbook Chapter 3, (p.50 ~ 52)
  cv2.addWeighted()

5. **Training Cifar10 Classifier Using VGG19** (20%)

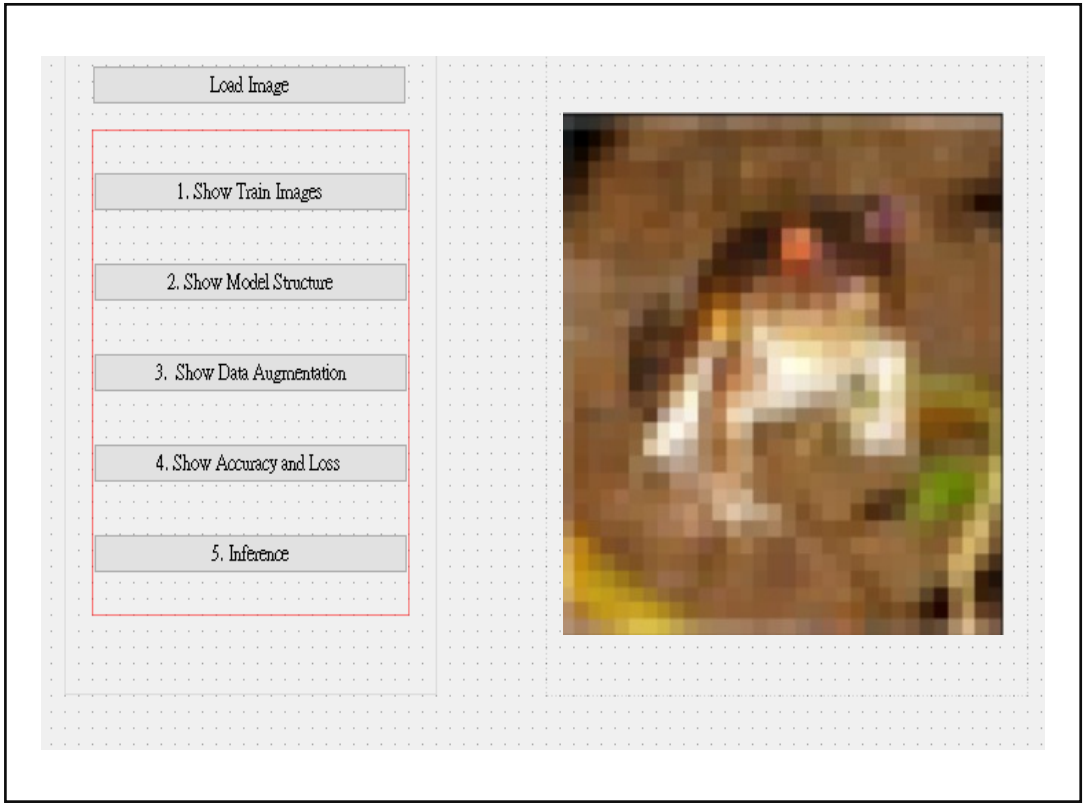
5.1 Load Cifar10 and Random Show 9 Images with Label(4%)

5.2 Load Model and Show Model Structure (4%)

5.3 Show Data Augmentation Result (4%)

5.4 Show Accuracy and Loss (4%)

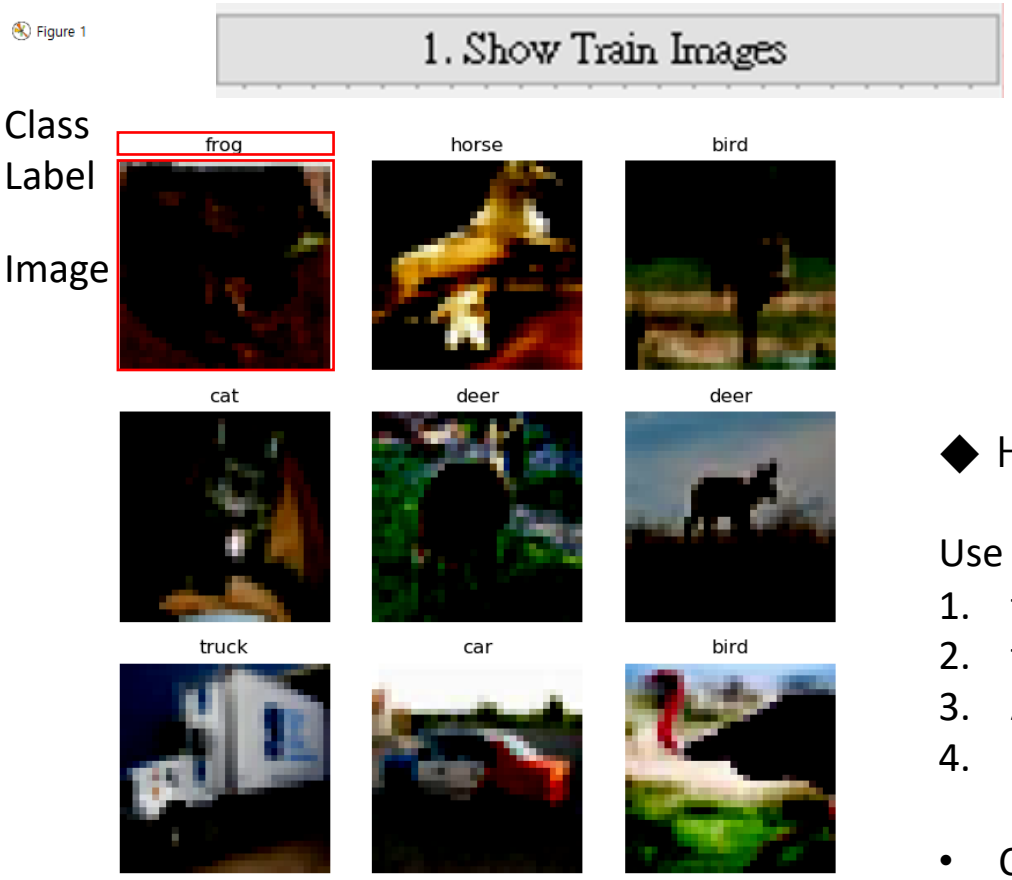
5.5 Inference (4%)



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()

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



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

◆ Hint

Use Matplotlib 4 function

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

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

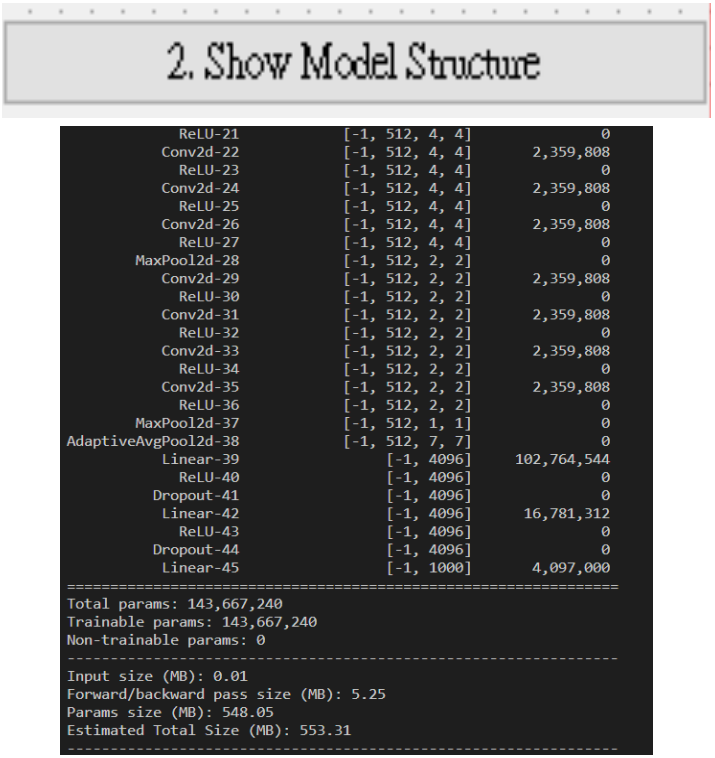
https://matplotlib.org/stable/tutorials/index.html

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.



◆ 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

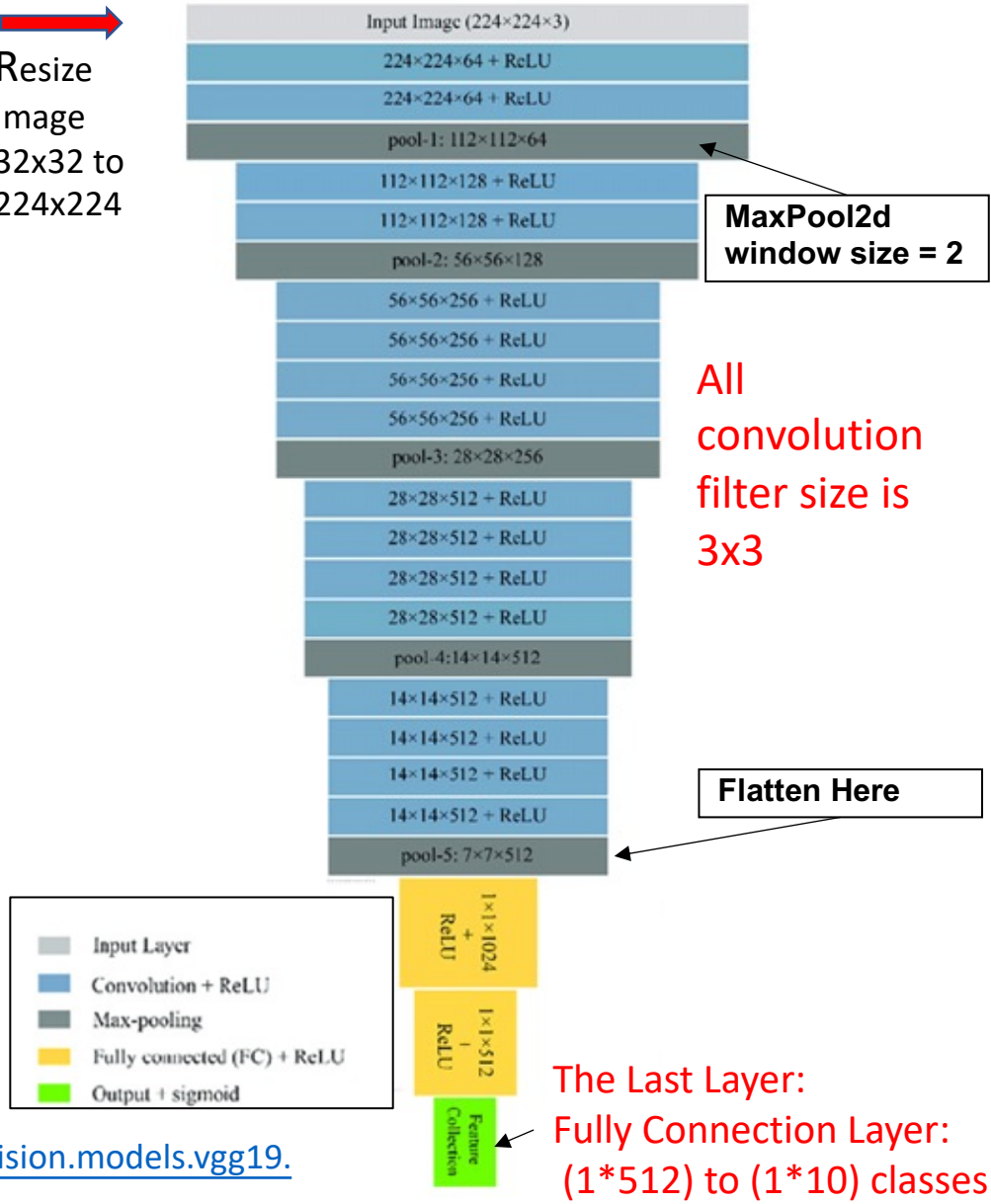
```
              ReLU-21          [-1, 512, 4, 4]               0
            Conv2d-22          [-1, 512, 4, 4]       2,359,808
              ReLU-23          [-1, 512, 4, 4]               0
            Conv2d-24          [-1, 512, 4, 4]       2,359,808
              ReLU-25          [-1, 512, 4, 4]               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



MaxPool2d window size = 2

All convolution filter size is 3x3

Flatten Here

The Last Layer: Fully Connection Layer: (1*512) to (1*10) classes

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)

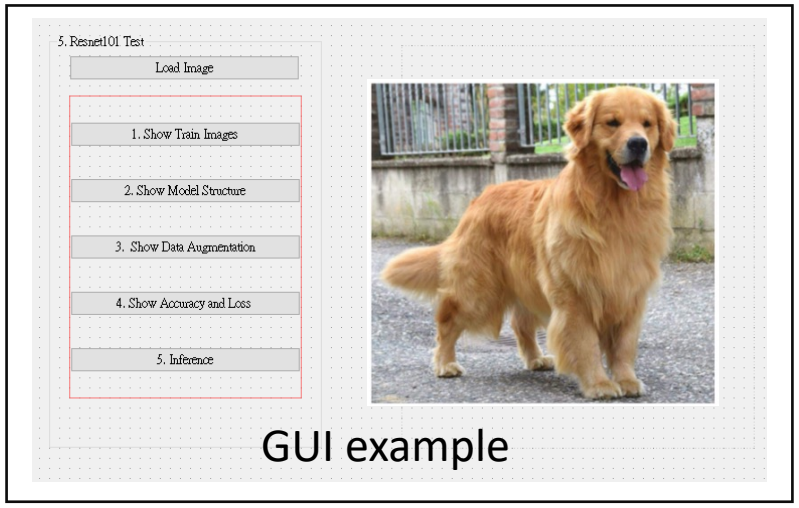
5.3 Show Data Augmentation Result (4%)

- 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.



GUI example

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) tesnsorflow: 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

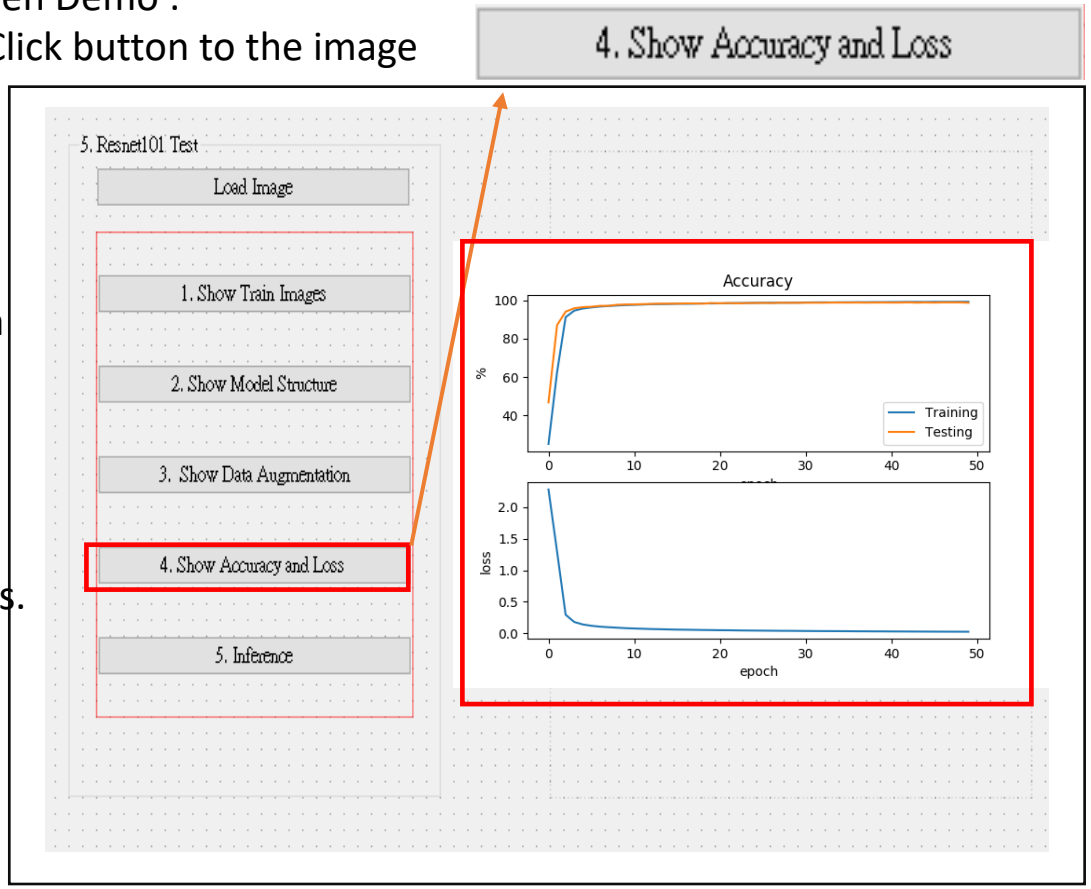
1. model which is called form 5.2.
2. model.fit() returns the accuracy and loss.
3. model.save() to save your model.

Notice

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

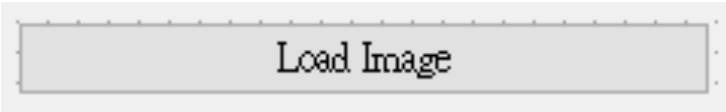
When Demo :

1. Click button to the image

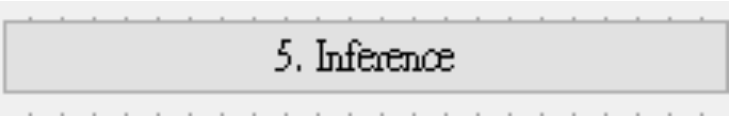


5.5 **Load the image and inference them** with your weighted file (.pth), and show the result image and class (4%)

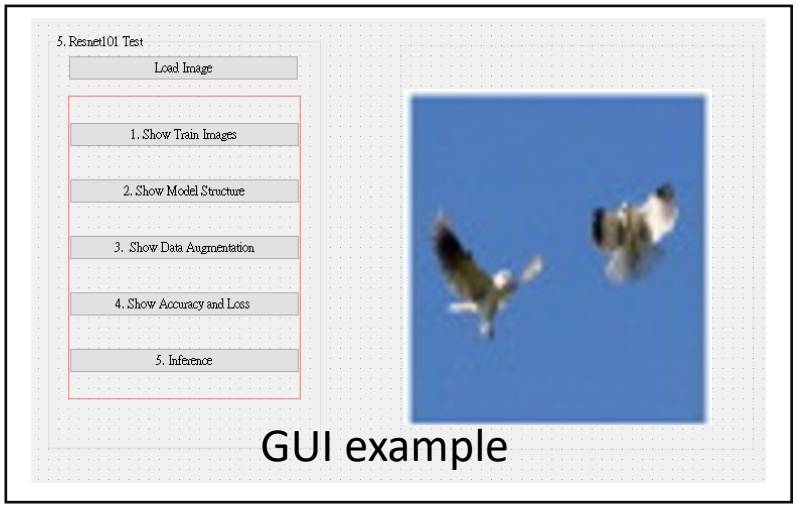
1. Choose the any data

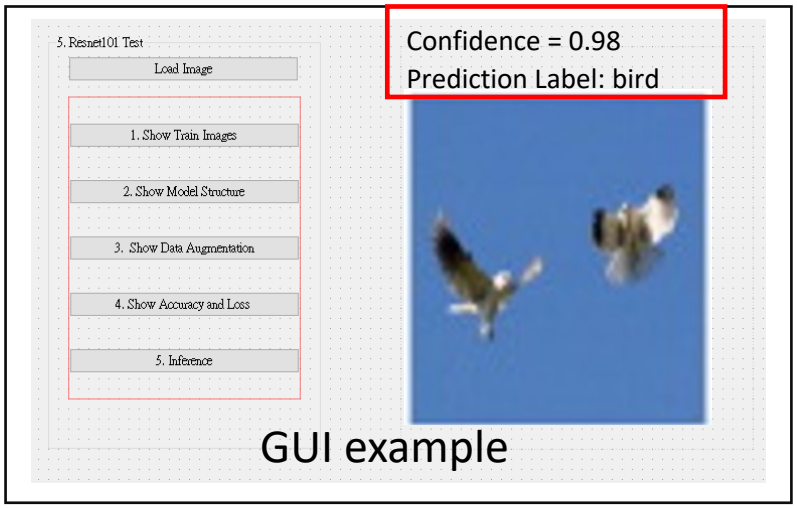


2. Run Inference



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