Institute of Electronics
National Yang Ming Chiao Tung University
Hsinchu, Taiwan

AI Training Course Series

Introduction to OpenCV, PyTorch Dataloader Preprocessing

Lecture 1

Architecture
Development &
Algorithm
Refinement for
Intelligent Computing

Presenter: Cheng-Chia Liao

Advisor: Juinn-Dar Huang, Ph.D.

July 8, 2024

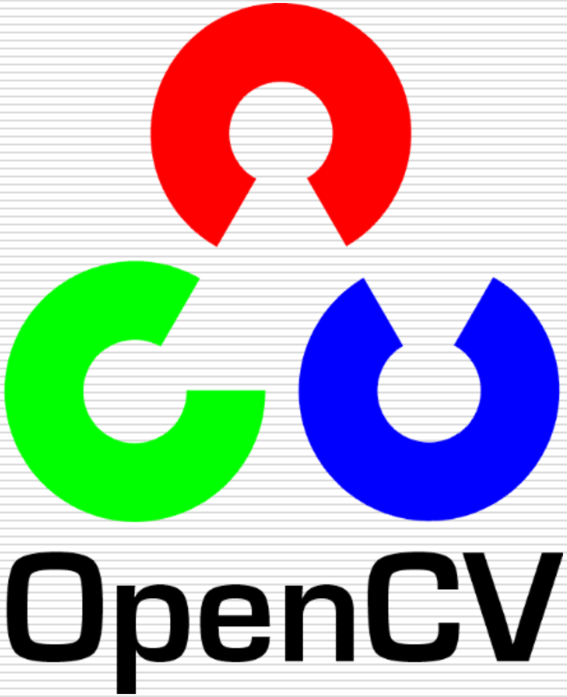
Outline

- Introduction to OpenCV & Installation
- Read / Write Image Files
- Image Processing in OpenCV
- Video Capture in OpenCV
- Python Image Library(PIL)
- PyTorch Load Data
- Preprocessing
- References
- Homework

Introduction to OpenCV & Installation

What is OpenCV

- Open Source Computer Vision Library
- Start with Intel
- Open source and cross-platform



Introduction of OpenCV

- Mainly written by C++
  - Also has Python, Java, MATLAB interfaces
- Support Windows, Linux, Mac OS, Android
- Support CUDA acceleration
- OpenCV supports many image file formats, including: BMP, JPEG, GIF, PNG, TIFF

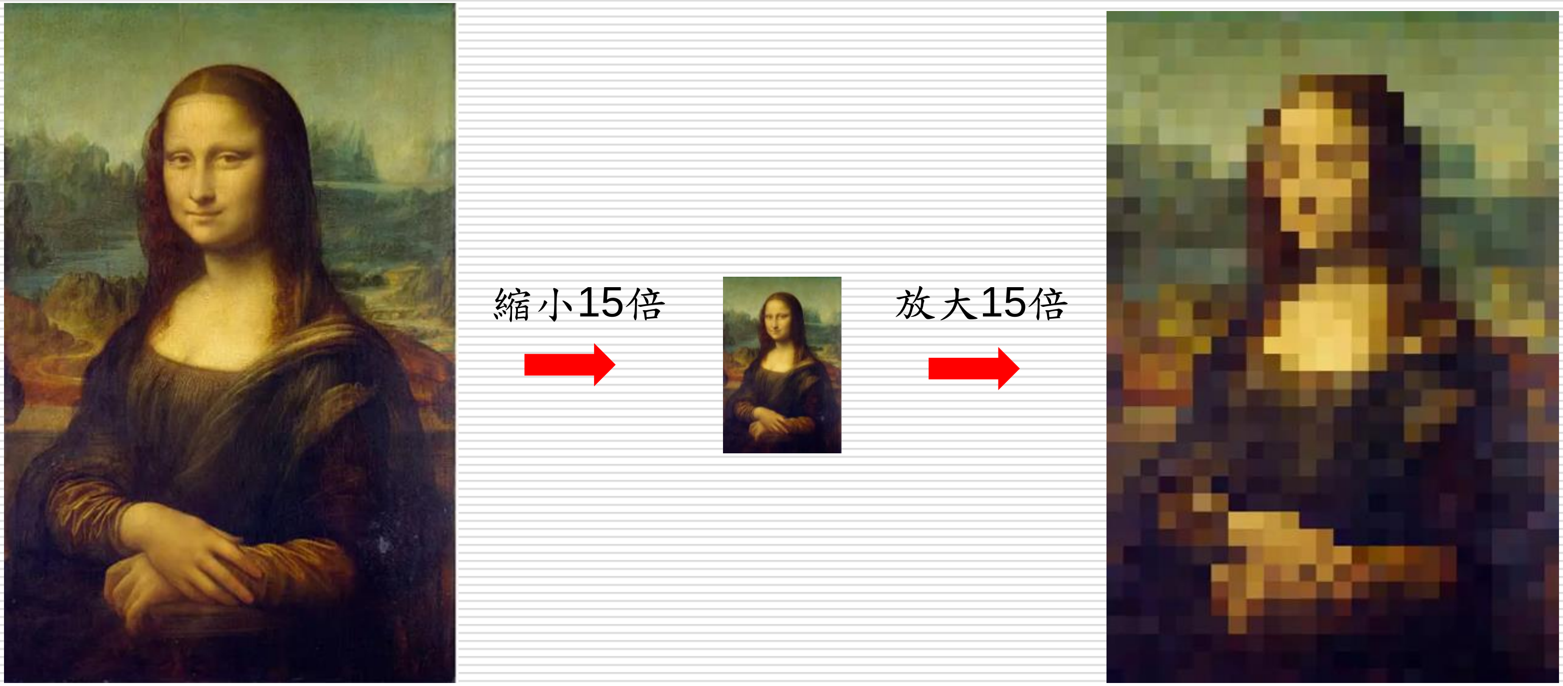
Applications of OpenCV (1/3)

- 影像處理
  - 圖像濾波、邊緣檢測、圖像修復、色彩空間轉換
- 特徵檢測與配對
- 人臉辨別
- 運動分析
- 汽車安全駕駛
- 物體識別
- etc.

Applications of OpenCV (2/3)

- **Mosaic(馬賽克)**

- Reduce the original image and then enlarge it to the original size to get a mosaic image



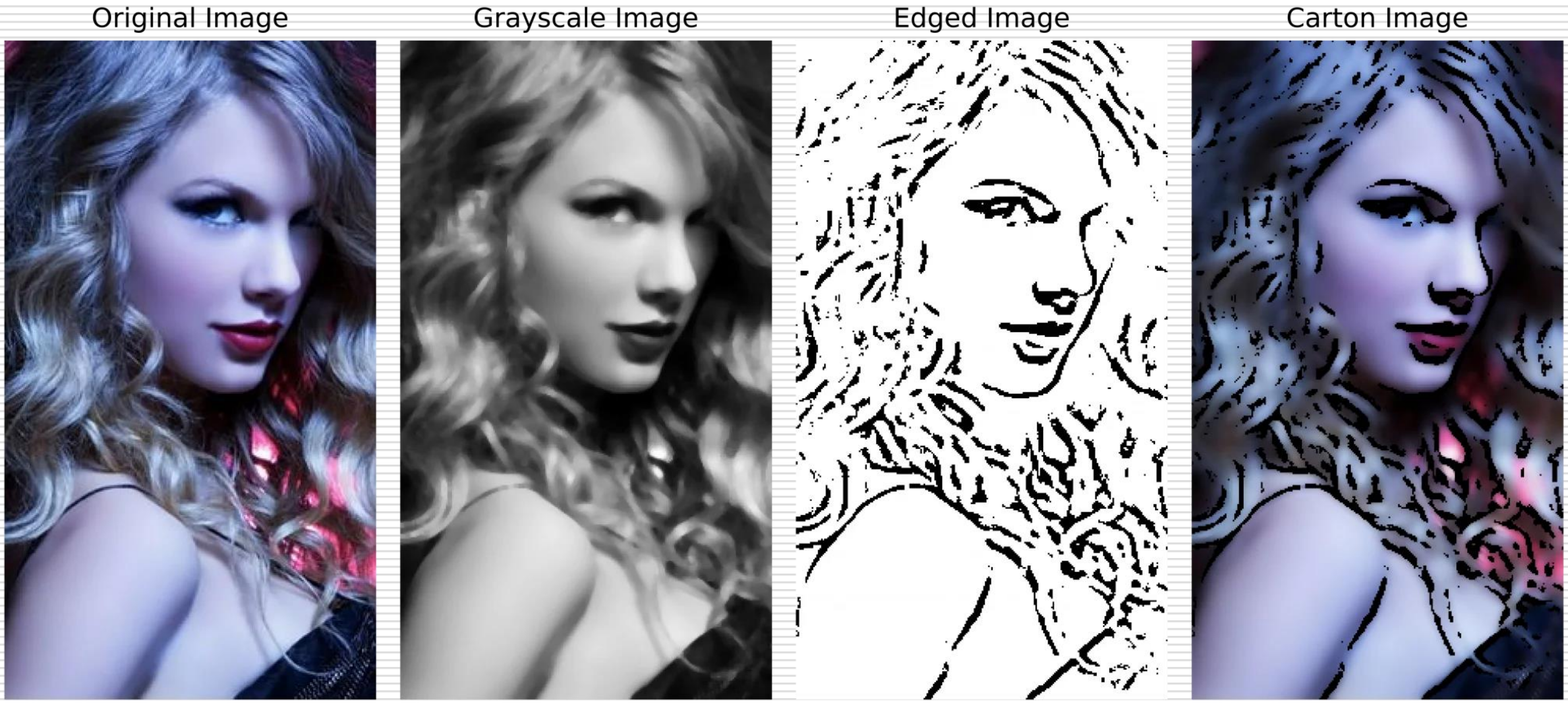
縮小15倍

放大15倍

Applications of OpenCV (3/3)

- **Convert Images into Cartoon using OpenCV**

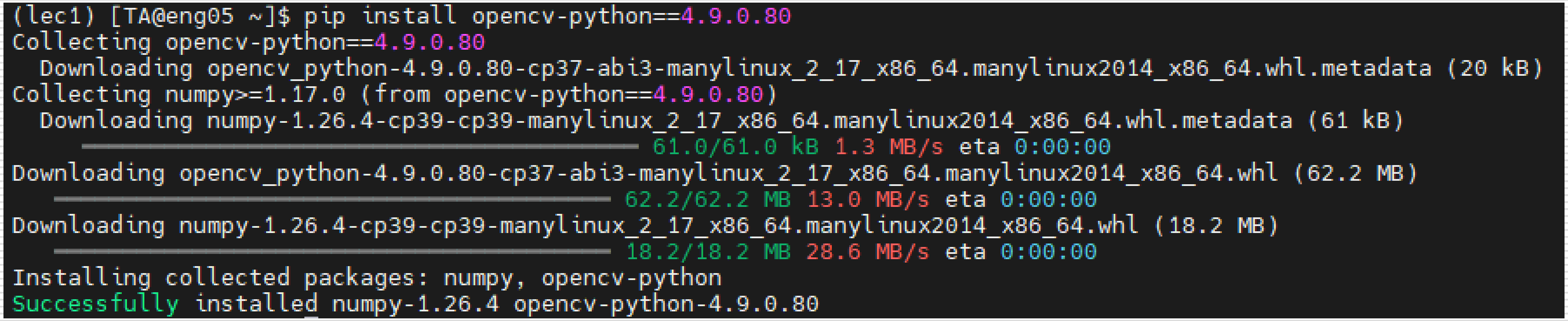
- Edges + Grayscale = cartoon image



Install OpenCV

- **$pip install opencv-python==4.9.0.80**
  - Install in your environment(torch), do NOT install in base

- version 4.9.0.80

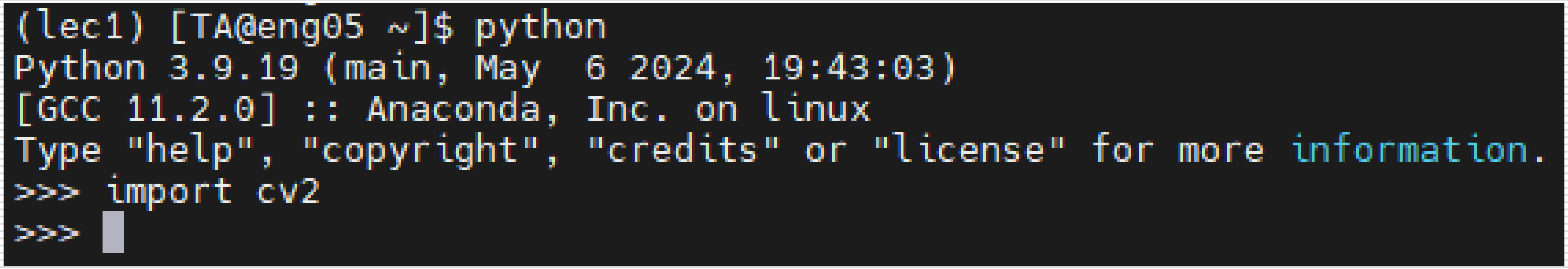


```
(lec1) [TA@eng05 ~]$ pip install opencv-python==4.9.0.80
Collecting opencv-python==4.9.0.80
  Downloading opencv_python-4.9.0.80-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (20 kB)
Collecting numpy>=1.17.0 (from opencv-python==4.9.0.80)
  Downloading numpy-1.26.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (61 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.0/61.0 kB 1.3 MB/s eta 0:00:00
Downloading opencv_python-4.9.0.80-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (62.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.2/62.2 MB 13.0 MB/s eta 0:00:00
Downloading numpy-1.26.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.2/18.2 MB 28.6 MB/s eta 0:00:00
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.26.4 opencv-python-4.9.0.80
```

Check Installation

- **$python**

- **$import cv2**



```
(lec1) [TA@eng05 ~]$ python
Python 3.9.19 (main, May  6 2024, 19:43:03)
[GCC 11.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> 
```

Read / Write Image Files

Read Images (1/5)

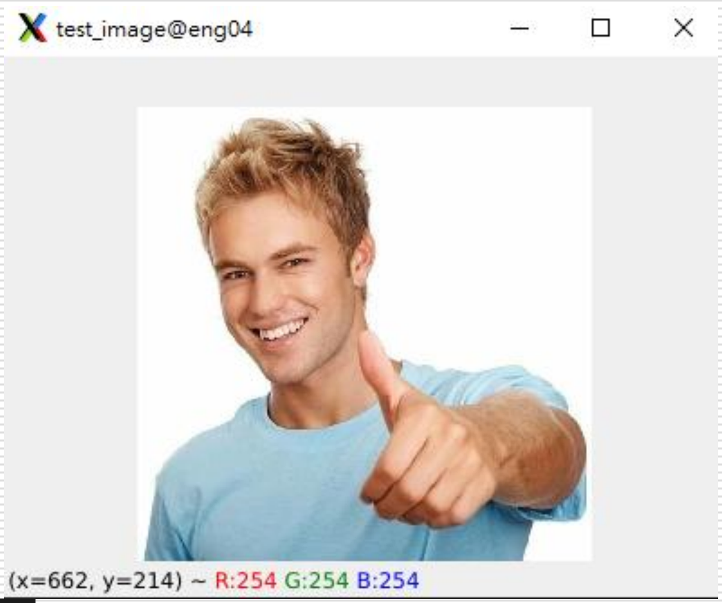
- **cv2.imread()**

- cv2.imread(‘filename’, format)
  - Image = cv2.imread('image.jpg', cv2.IMREAD_COLOR)

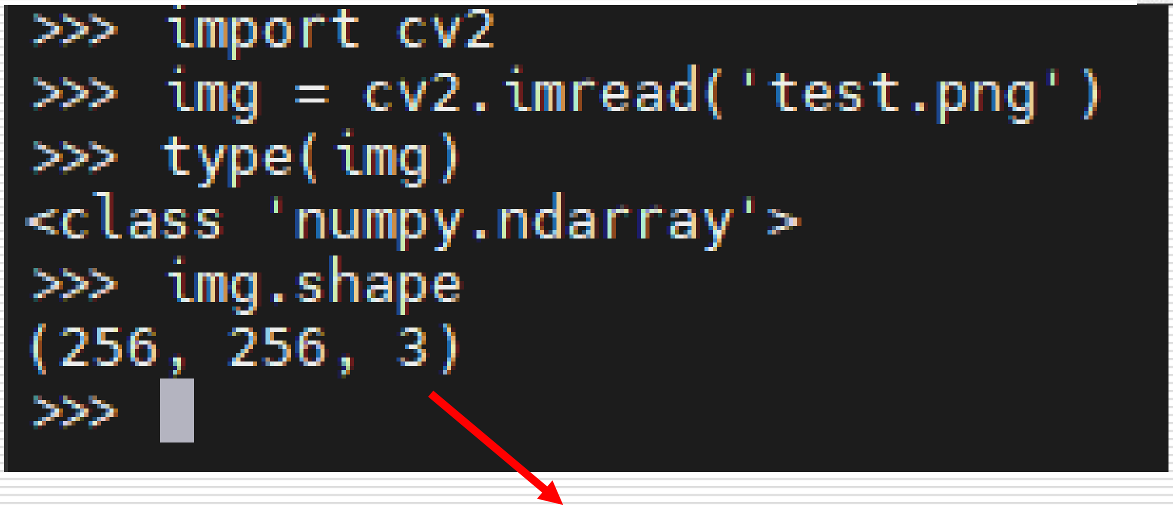
- Format:
  - cv2.IMREAD_COLOR(可用1代替或是不寫)
    - Default, BGR
  - cv2.IMREAD_GRAYSCALE(可用0代替)
  - cv2.IMREAD_UNCHANGED(可用-1代替)
    - 包含透明度的channel

Read Images (2/5)

- $ img = cv2.imread(‘test.png’)
- $ type(img)
  - NumPy
- $ img.shape
  - (Height, Width, Channel)



(x=662, y=214) ~ R:254 G:254 B:254



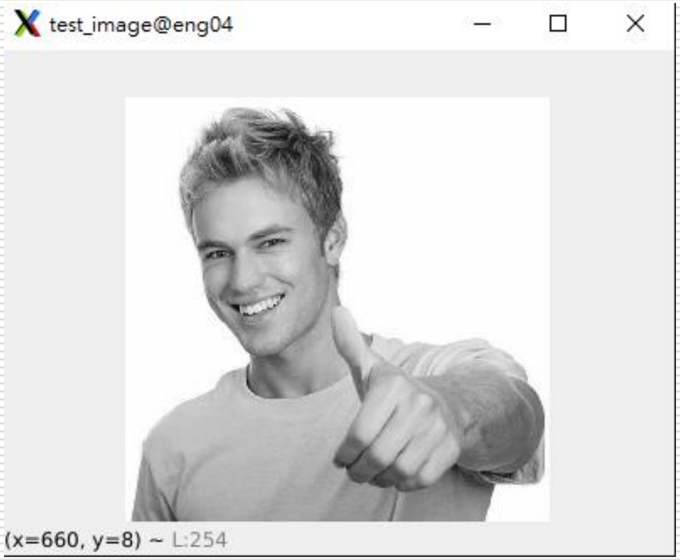
```
>>> import cv2
>>> img = cv2.imread('test.png')
>>> type(img)
<class 'numpy.ndarray'>
>>> img.shape
(256, 256, 3)
>>>
```

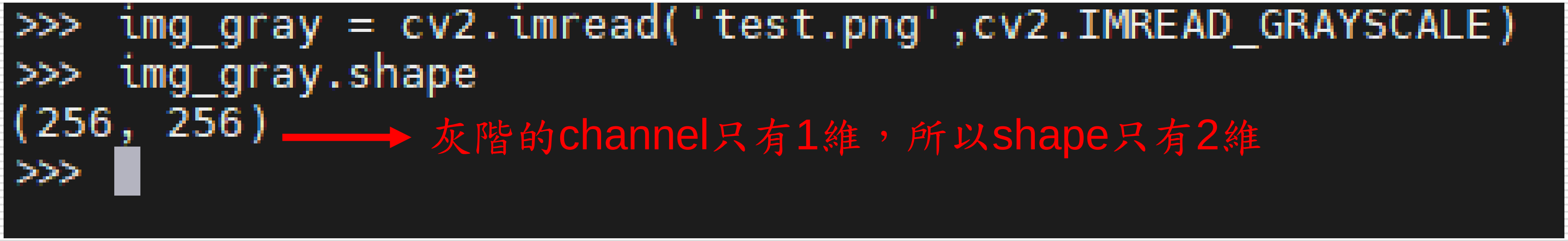
Channel: Blue + Green + Red = 3

Read Images (3/5)

- $ img_gray = cv2.imread(‘test.png’, cv2.IMREAD_GRAYSCALE)

- $ img_gray.shape



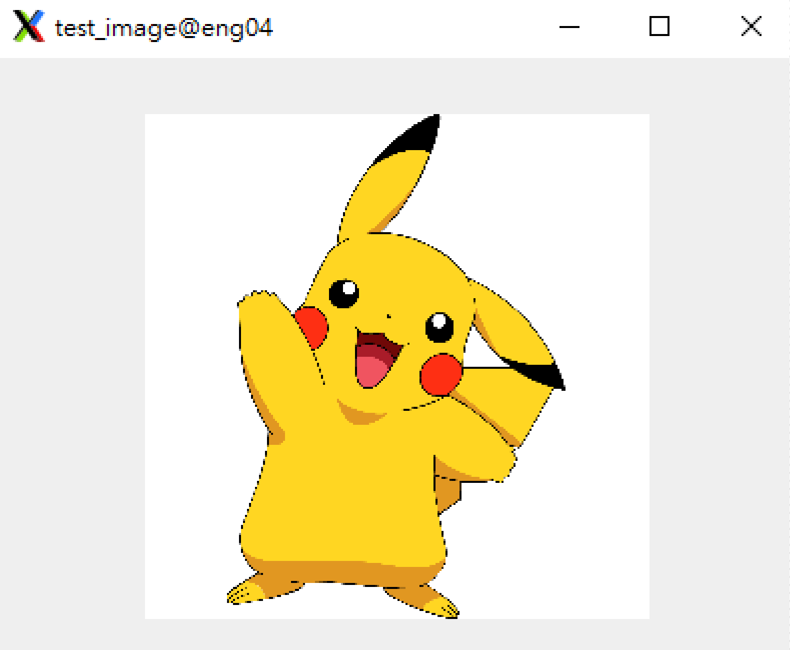


```
>>> img_gray = cv2.imread('test.png',cv2.IMREAD_GRAYSCALE)
>>> img_gray.shape
(256, 256)
>>> 
```

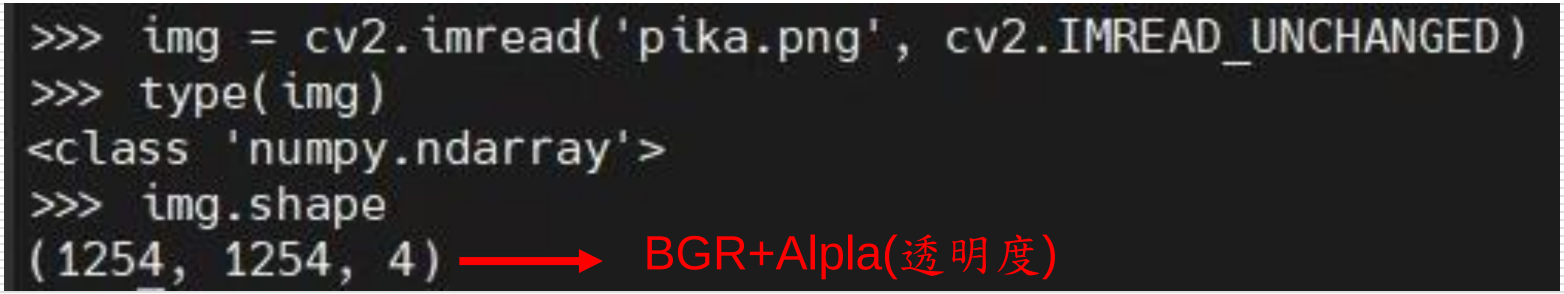
灰階的channel只有1維，所以shape只有2維

Read Images (4/5)

- img = cv2.imread(‘pika.png’, cv2.IMREAD_UNCHANGED)



- Img.shape()

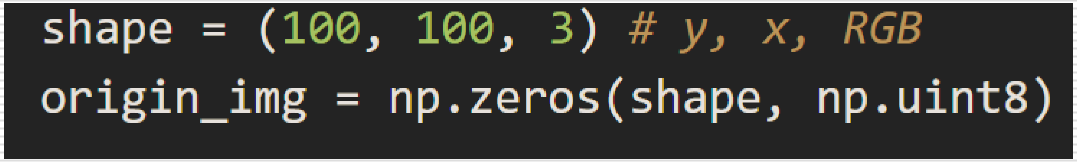


```
>>> img = cv2.imread('pika.png', cv2.IMREAD_UNCHANGED)
>>> type(img)
<class 'numpy.ndarray'>
>>> img.shape
(1254, 1254, 4)
```

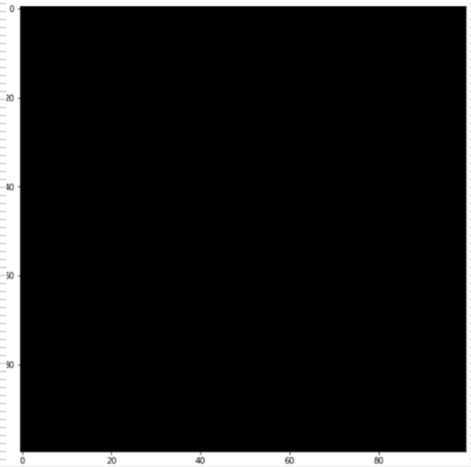
BGR+Alpla(透明度)

Read Images (5/5)

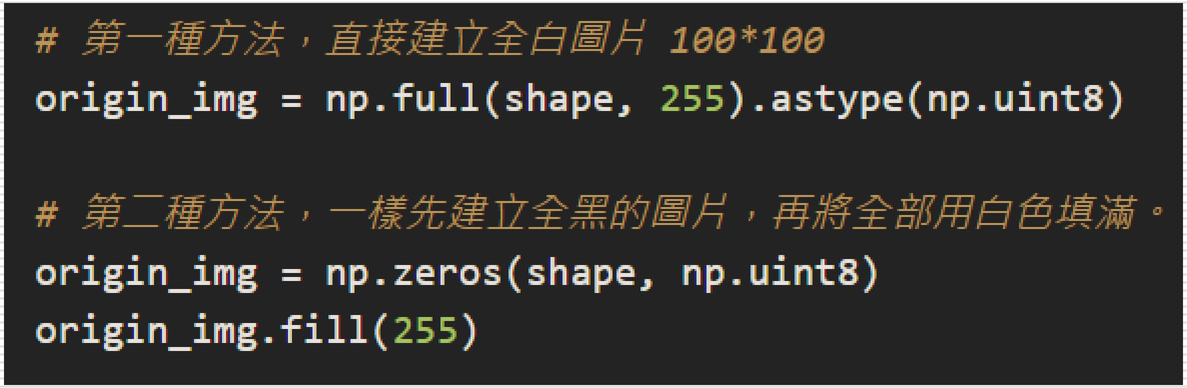
- 建立全黑的圖片 (100 x 100)



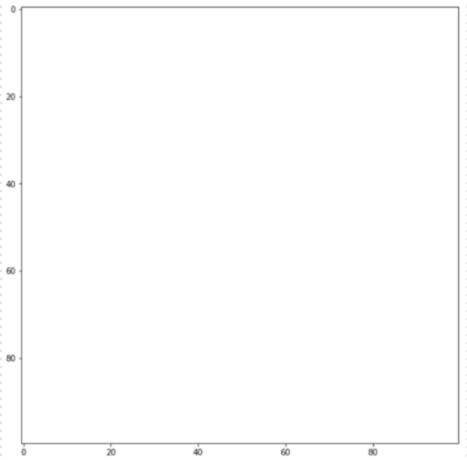
```
shape = (100, 100, 3) # y, x, RGB
origin_img = np.zeros(shape, np.uint8)
```



- 建立全白的圖片



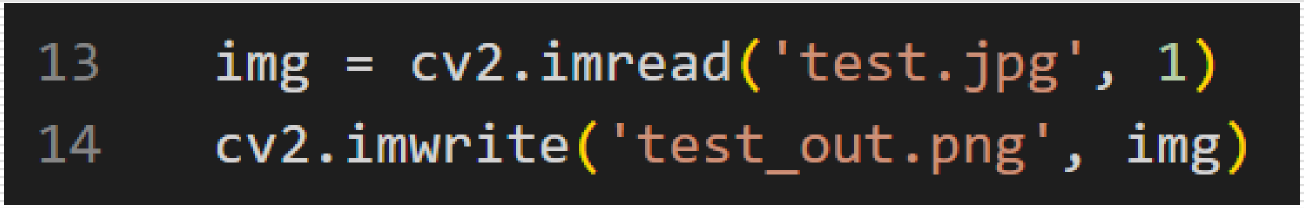
```
# 第一種方法，直接建立全白圖片 100*100
origin_img = np.full(shape, 255).astype(np.uint8)

# 第二種方法，一樣先建立全黑的圖片，再將全部用白色填滿。
origin_img = np.zeros(shape, np.uint8)
origin_img.fill(255)
```



Write Images (1/2)

- **cv2.imwrite()**

- **$ cv2.imwrite(‘file_name’, image_name)**
  - Determine the format with the filename extension

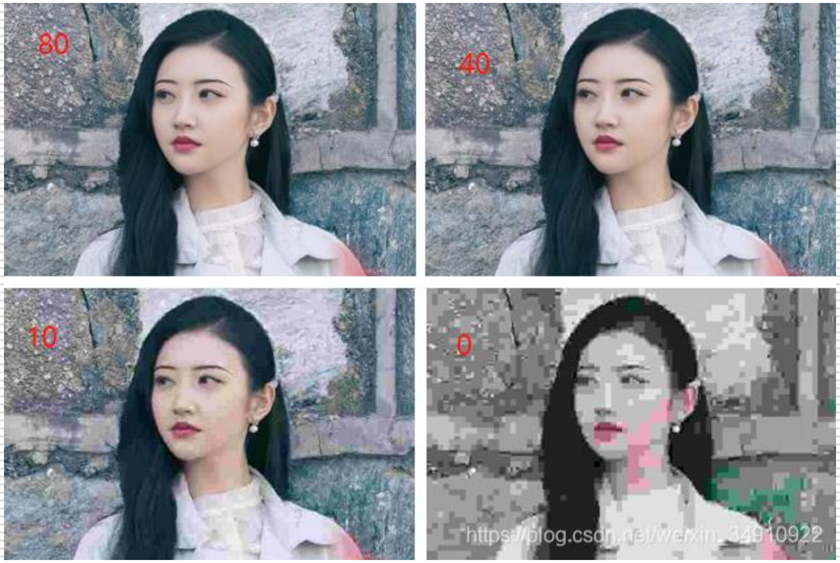


```
img = cv2.imread('test.jpg', 1)
cv2.imwrite('test_out.png', img)
```

Write Images (2/2)

- Set quality (0~100)
  - **$ cv2.imwrite(‘test.jpg', img, [cv2.IMWRITE_JPEG_QUALITY, 80])**

- Set compression level (0~9)
  - **$ cv2.imwrite(‘test.png', img, [cv2.IMWRITE_PNG_COMPRESSION, 6])**



jpg set quality

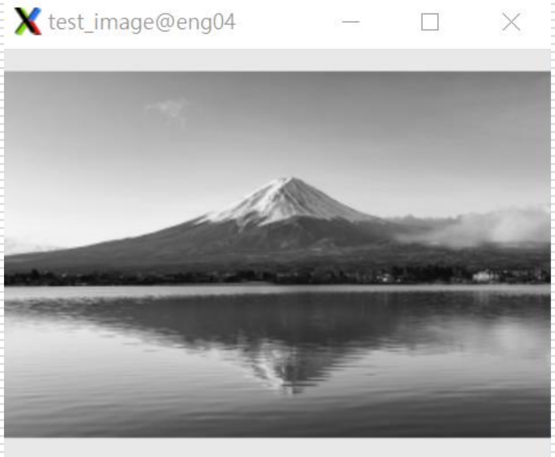
png set compression level

Show Image Files

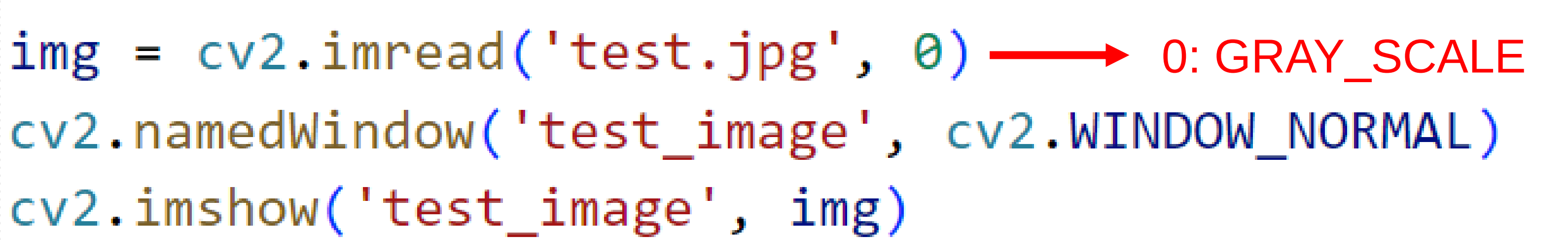
Show Images (1/3)

- **cv2.namedWindow()**
- **cv2.imshow()**

- Create a scalable window and show an image
  - cv2.namedWindow(‘window_name’, cv2.WINDOW_NORMAL)
  - cv2.imshow(‘window_name’, img_name)



- Use namedWindow() before imshow()

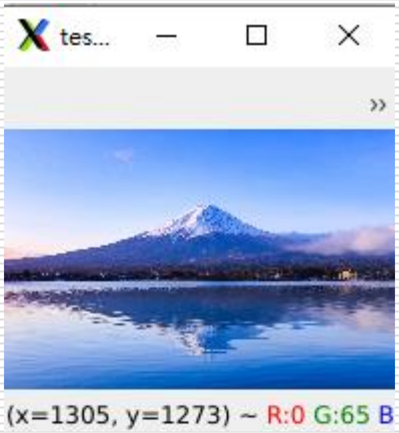


```
img = cv2.imread('test.jpg', 0)
cv2.namedWindow('test_image', cv2.WINDOW_NORMAL)
cv2.imshow('test_image', img)
```

0: GRAY_SCALE

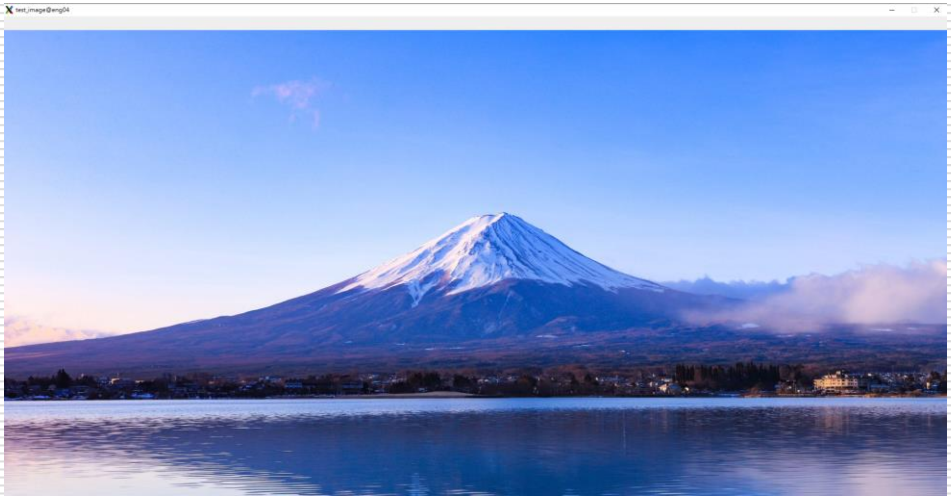
Show Images (2/3)

- cv2.namedWindow(‘window_name’, **cv2.WINDOW_NORMAL**)
  - 視窗大小可以改變



(x=1305, y=1273) ~ R:0 G:65 B

- cv2.namedWindow(‘window _name’, **cv2.WINDOW_AUTOSIZE**)
  - 視窗大小不可改變

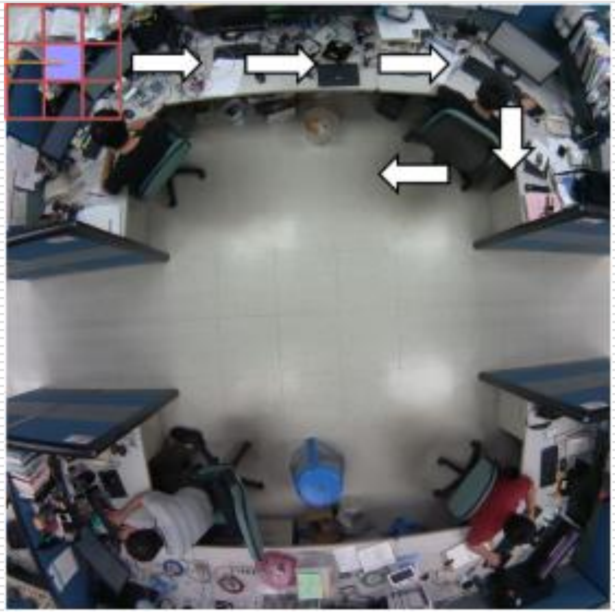


Show Images (3/3)

- **cv2.waitKey()** sets the waiting time
  - 不加waitKey圖像視窗會很快顯示並關閉
  - ()內填入等待時間，單位:ms
  - ()內填入0，視窗會持續顯示，按任意按鍵可關閉視窗
- **cv2.destroyAllWindows()**
  - closes all windows
- **cv2.DestroyWindow('window_name')**
  - closes the specific window
  - 不加destroyAllWindows可能會使程式無法正常結束

Image Processing in OpenCV

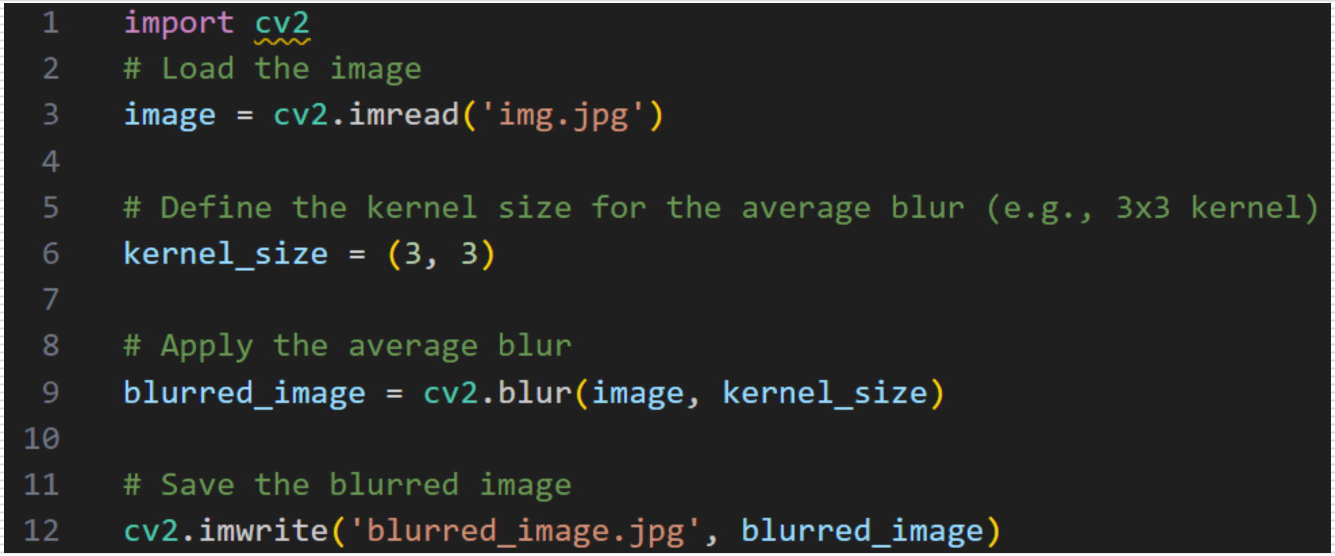
Smoothing



- Average blur
  - cv2.blur(img, kernel_size)
- Median blur
  - cv2.medianBlur(img, kernel_size)
- Gaussian blur
  - cv2.GaussianBlur(img, kernel_size, variance)

Average Blur

- cv2.blur(img, kernel_size)



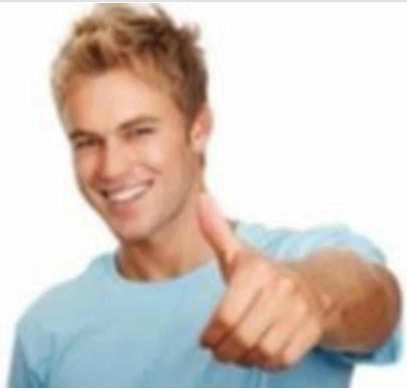
```python
import cv2
# Load the image
image = cv2.imread('img.jpg')

# Define the kernel size for the average blur (e.g., 3x3 kernel)
kernel_size = (3, 3)

# Apply the average blur
blurred_image = cv2.blur(image, kernel_size)

# Save the blurred image
cv2.imwrite('blurred_image.jpg', blurred_image)
```

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

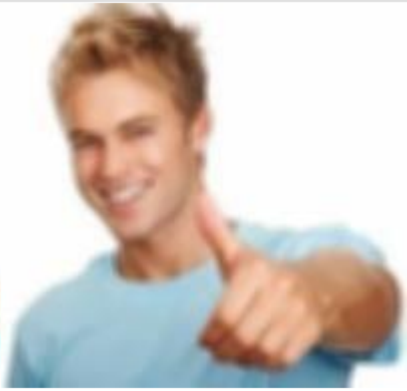
Average blurring kernel

Original

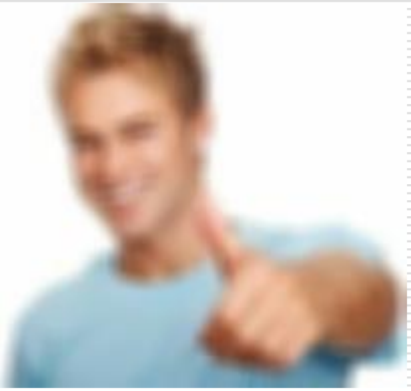
kernel size:3x3



kernel size:5x5



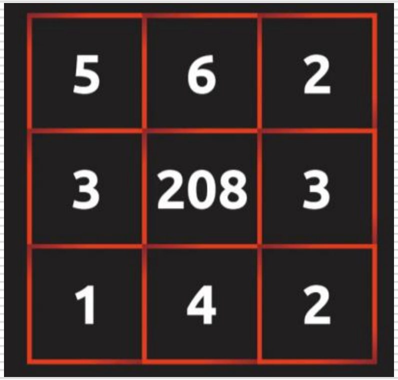
kernel size:9x9



Median Blur (1/2)

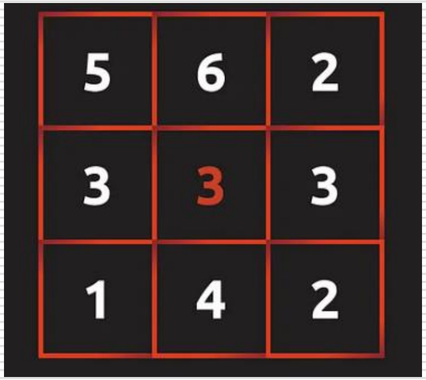
- cv2.medianBlur(img, kernel_size)
- 計算 kernel 視窗內所有 pixel 的中位數，再取代 kernel 中間的數值
- 常用於濾除雜訊

Original



5	6	2
3	208	3
1	4	2

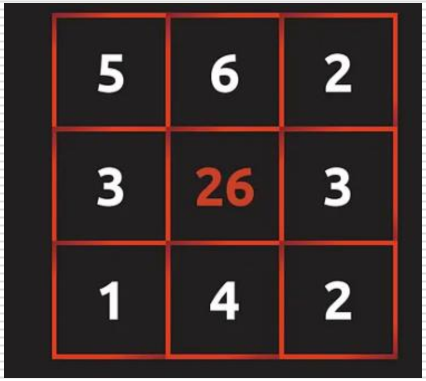
Median Blur Result



5	6	2
3	3	3
1	4	2

由小到大的排序：
1, 2, 2, 3, 「**3**」, 4, 5, 6, 208

Average Blur Result

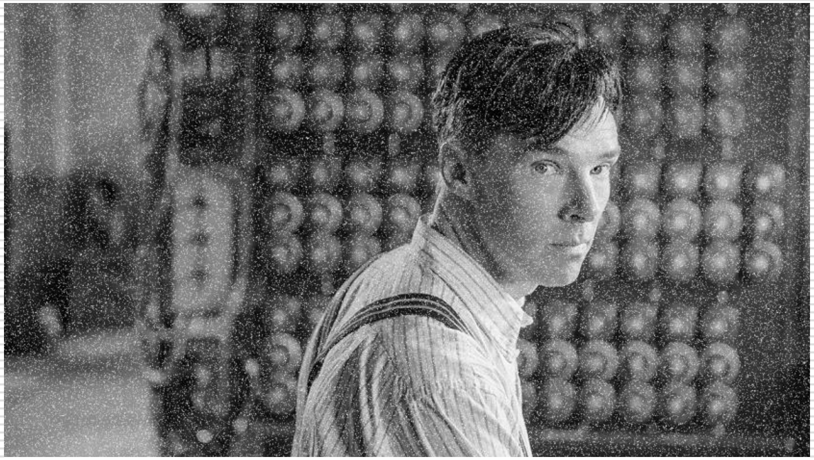


5	6	2
3	26	3
1	4	2

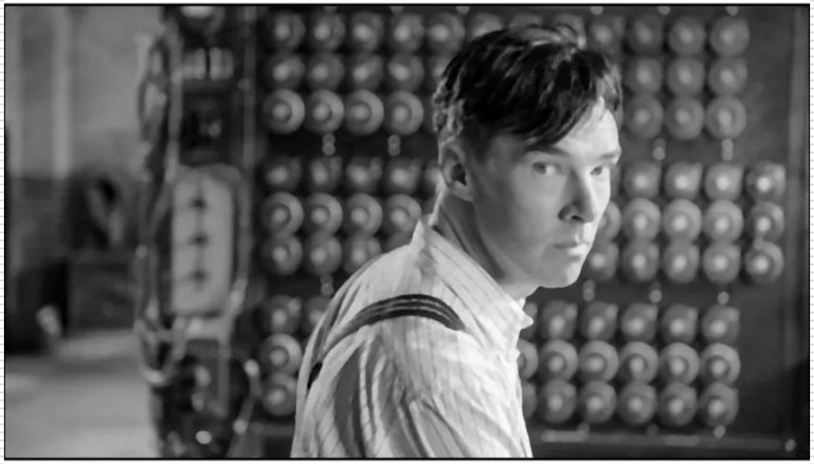
（5+6+2+3+208+3+1+4+2）/ 9
= 26

Median Blur (2/2)

Original image



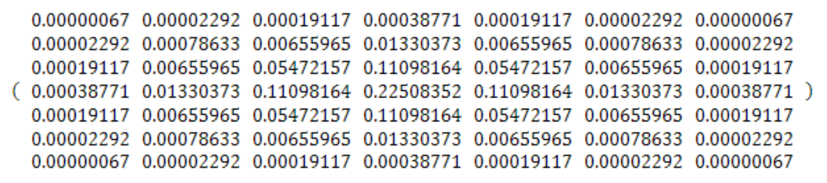
Median Blur Image, mask size: 7



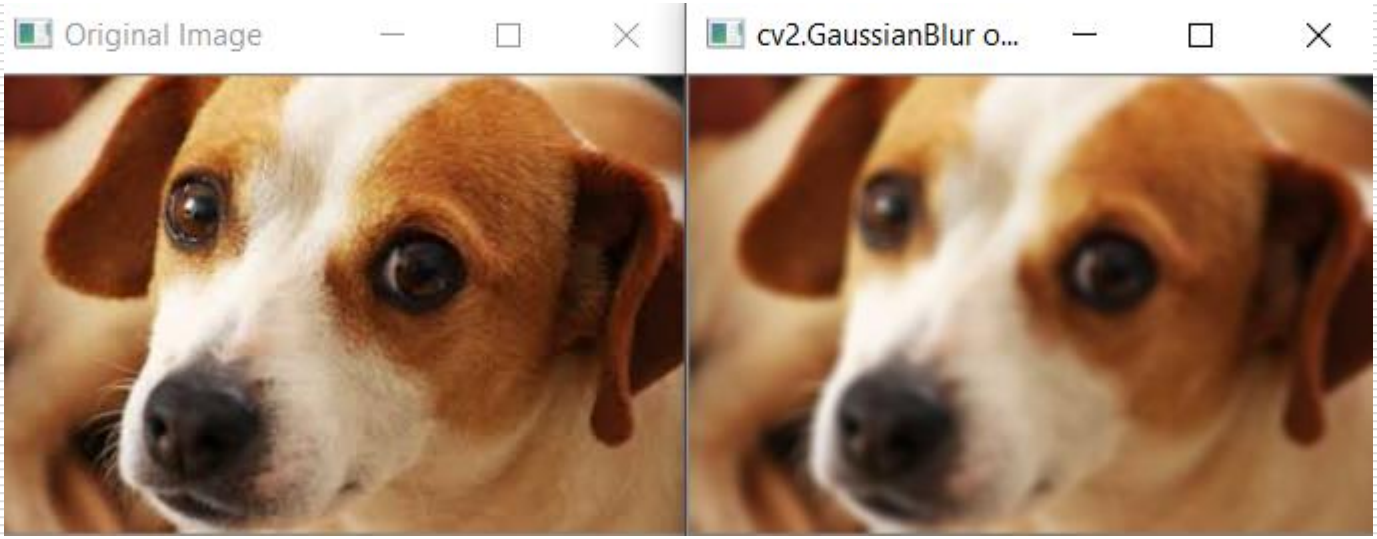
Gaussian Blur

- cv2.GaussianBlur(img, kernel_size, variance)
  - 利用高斯函數來計算每個像素的加權平均值
  - Variance越大，模糊程度越高
- 使圖像變得模糊，從而達到降噪、減少細節的效果

Gaussian kernel matrix example



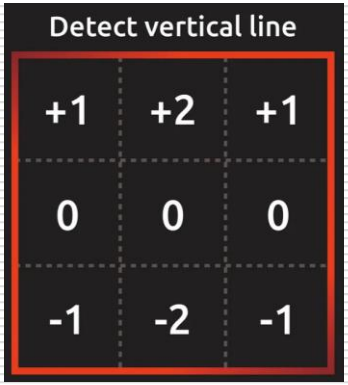
$$\begin{pmatrix}
0.00000067 & 0.00002292 & 0.00019117 & 0.00038771 & 0.00019117 & 0.00002292 & 0.00000067 \\
0.00002292 & 0.00078633 & 0.00655965 & 0.01330373 & 0.00655965 & 0.00078633 & 0.00002292 \\
0.00019117 & 0.00655965 & 0.05472157 & 0.11098164 & 0.05472157 & 0.00655965 & 0.00019117 \\
0.00038771 & 0.01330373 & 0.11098164 & 0.22508352 & 0.11098164 & 0.01330373 & 0.00038771 \\
0.00019117 & 0.00655965 & 0.05472157 & 0.11098164 & 0.05472157 & 0.00655965 & 0.00019117 \\
0.00002292 & 0.00078633 & 0.00655965 & 0.01330373 & 0.00655965 & 0.00078633 & 0.00002292 \\
0.00000067 & 0.00002292 & 0.00019117 & 0.00038771 & 0.00019117 & 0.00002292 & 0.00000067
\end{pmatrix}$$



Canny Edge Detection (1/4)

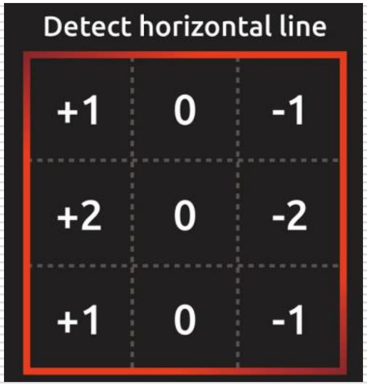
- **edges = cv2.canny(img_gray, low_threshold, high_threshold)**
  - **canny的input必須是gray scale image**

- Use Sobel kernel to detect edge



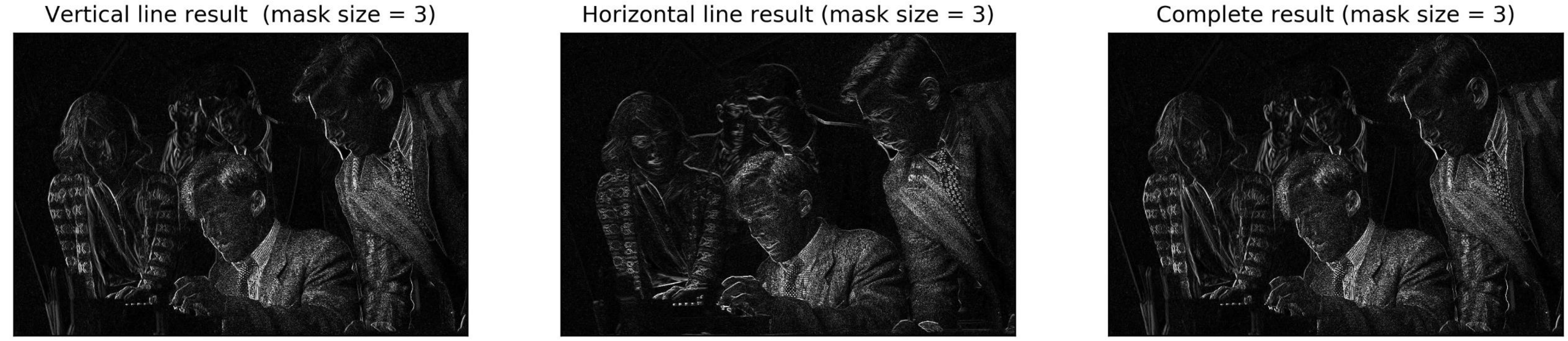
Detect vertical line

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



Detect horizontal line

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



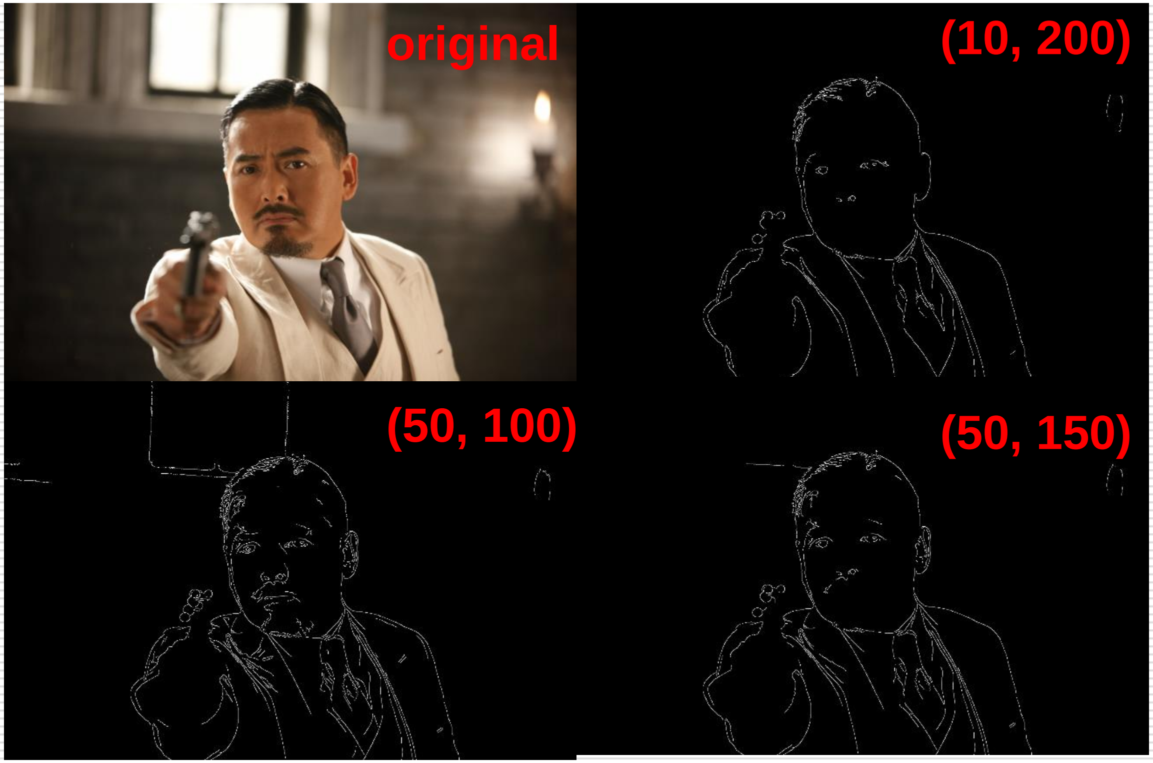
Vertical line result (mask size = 3)

Horizontal line result (mask size = 3)

Complete result (mask size = 3)

Canny Edge Detection (2/4)

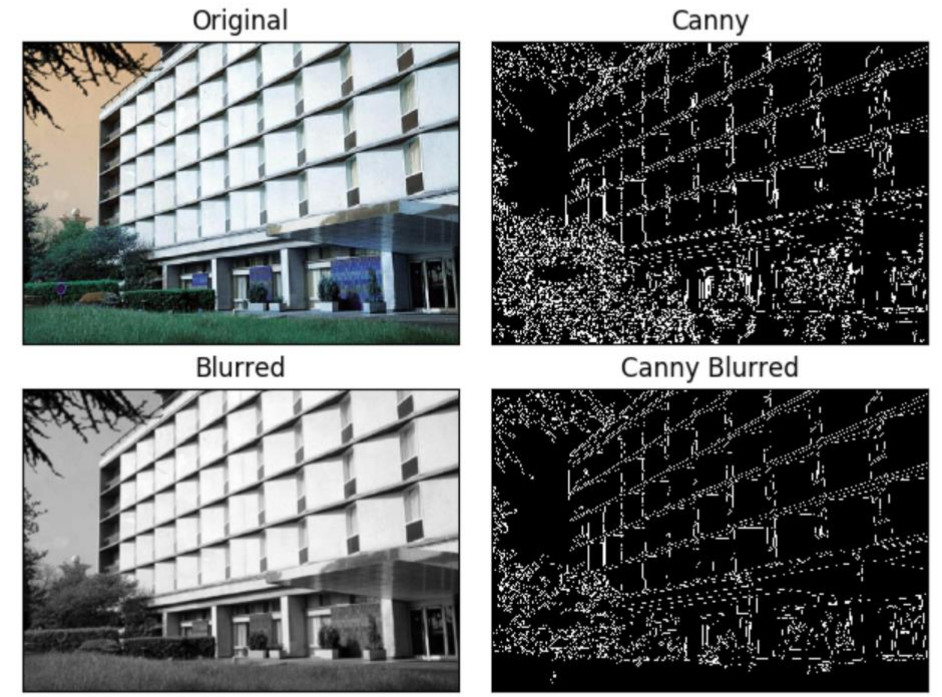
- **edges = cv2.canny(img_gray, low_threshold, high_threshold)**
- low_threshold connect the non-continuous edge
- high_threshold is the main threshold
  - John Canny演算法作者建議低到高比例為1：2或1：3



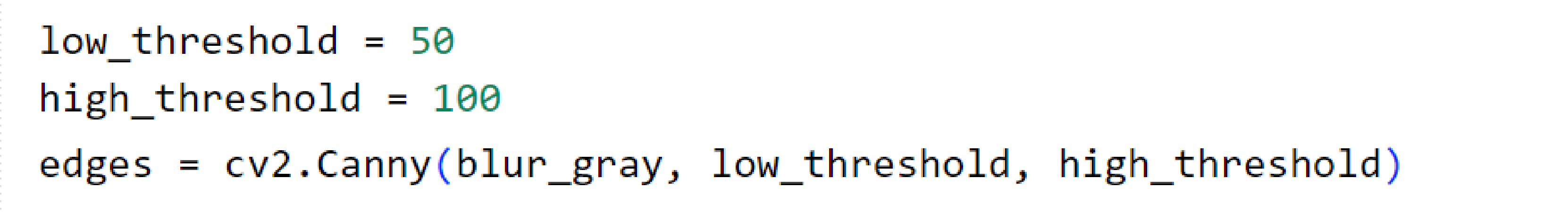
Canny Edge Detection (3/4)

- gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  - 將圖片從BGR轉成灰階

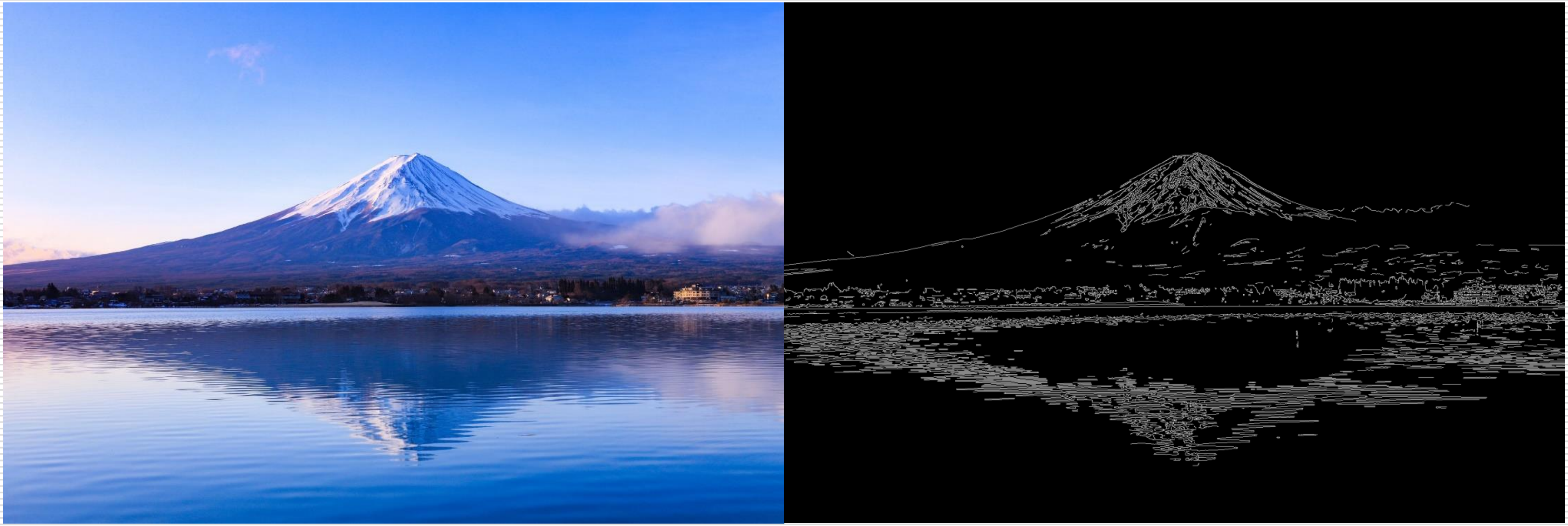
- blur_gray = cv2.GaussianBlur(gray, (3, 3), 0)
  - 在做邊緣偵測時，通常會先做平滑化來降低雜訊



Canny Edge Detection (4/4)



```
low_threshold = 50
high_threshold = 100
edges = cv2.Canny(blur_gray, low_threshold, high_threshold)
```



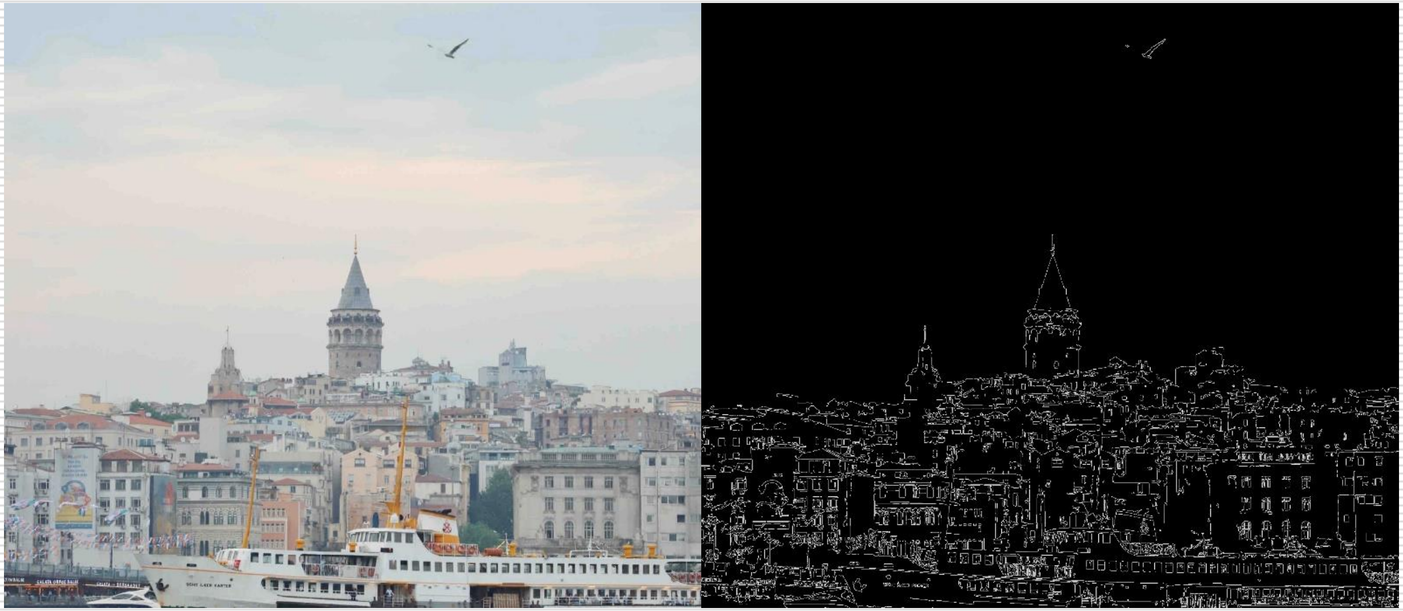
More Image Processing in OpenCV

- OpenCV Tutorials
  - https://docs.opencv.org/3.4/d2/d96/tutorial_py_table_of_contents_imgproc.html

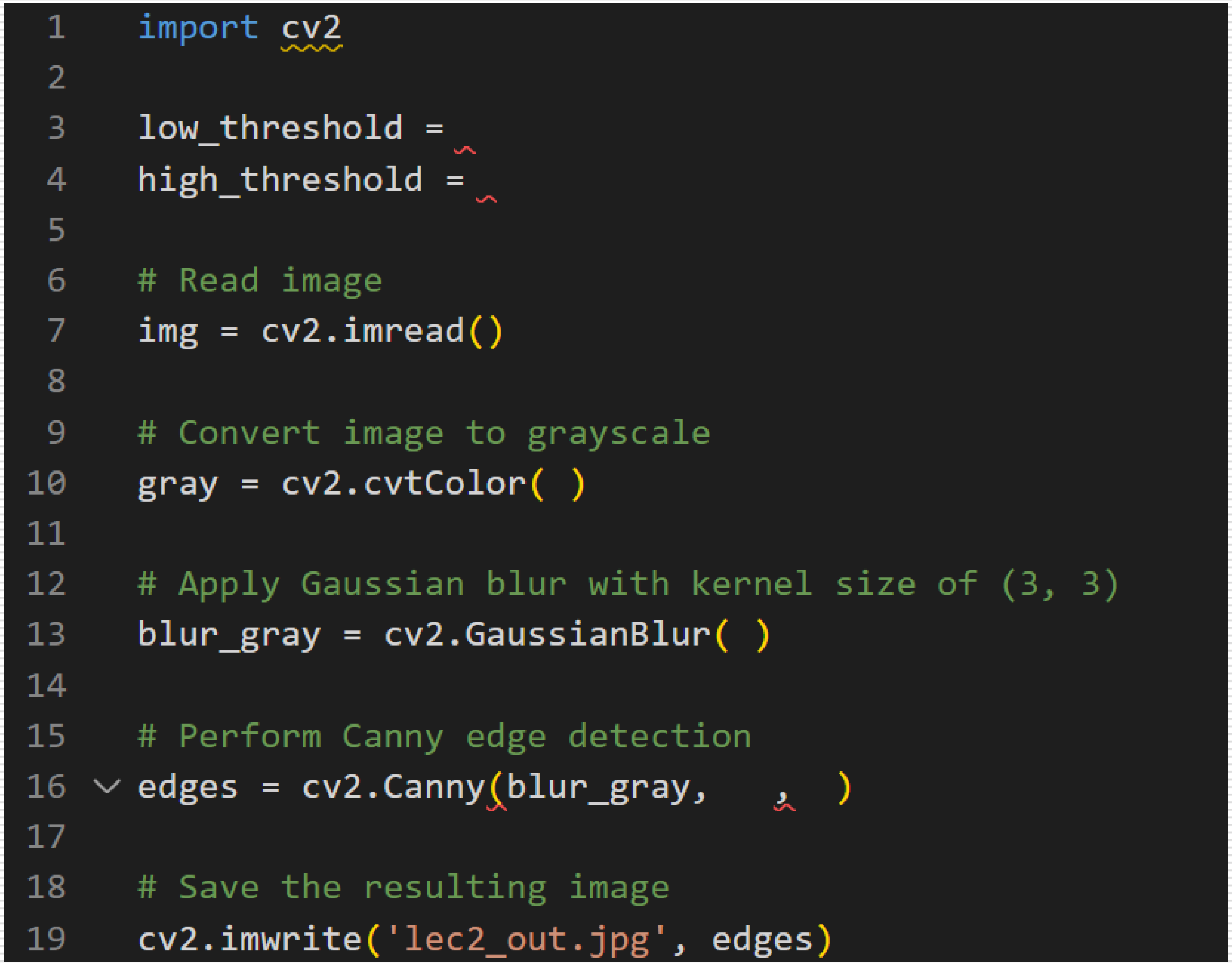
- Image Processing Using OpenCV – With Practical Examples
  - https://www.analyticsvidhya.com/blog/2021/05/image-processing-using-opencv-with-practical-examples/

Exercise (1/2)

- Canny Edge Detection實作
  - 將input image過完canny edge detection，得到output image
- 於eng05 server(140.113.225.245)上執行
  - $ tar -xvf /DATA/AI_training_HW_2024/lec1.tar



Exercise (2/2)



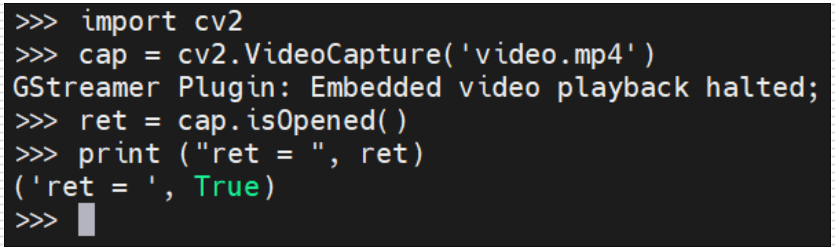
```python
import cv2

low_threshold = 
high_threshold = 

# Read image
img = cv2.imread()

# Convert image to grayscale
gray = cv2.cvtColor( )

# Apply Gaussian blur with kernel size of (3, 3)
blur_gray = cv2.GaussianBlur( )

# Perform Canny edge detection
edges = cv2.Canny(blur_gray,    ,   )

# Save the resulting image
cv2.imwrite('lec2_out.jpg', edges)
```

Video Capture in OpenCV

Video Capture (1/2)

- **cap = cv2.VideoCapture(filename)**
  - creates a VideoCapture object “cap” and initializes it to read frames from the input video file

- **ret = cap.isOpened( )**
  - Used to check whether the initialization was successful
    - Success: return True
    - Fail: return False



```
>>> import cv2
>>> cap = cv2.VideoCapture('video.mp4')
GStreamer Plugin: Embedded video playback halted;
>>> ret = cap.isOpened()
>>> print ("ret = ", ret)
('ret = ', True)
>>> 
```

- If initialization fails, it can be opened with

ret = cv2.VideoCapture.open(filename)

Video Capture (2/2)

- **cv2.VideoCapture.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)**
  - 設定frame的寬為1280

- **cv2.VideoCapture.set(cv2.CAP_PROP_FRAME_HEIGHT, 960)**
  - 設定frame的高為960

- **ret, frame = cap.read()**
  - 存取frame並檢查 ret 的值，以確定frame的讀取是否成功

Video Writer (1/4)

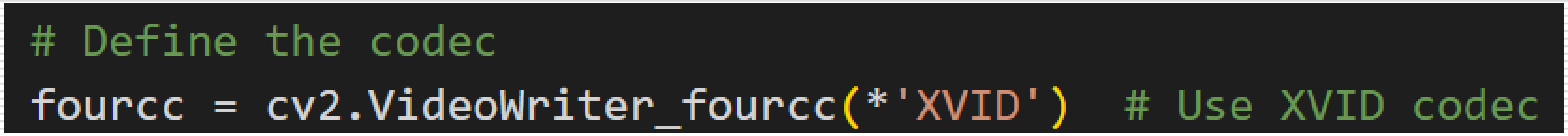
- The **cv2.VideoWriter()** function can convert pictures and images read by the camera into video files

- Modify the properties of the video and the conversion of the video type

- The steps to save a video include creating an object, writing a video, and releasing an object

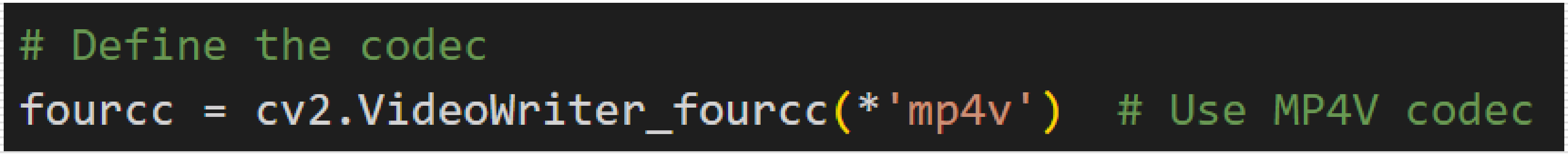
Video Writer (2/4)

- **fourcc = cv2.VideoWriter_fourcc(*'XVID’)**
  - 表示 XVID 編碼格示，副檔名為 .avi



```
# Define the codec
fourcc = cv2.VideoWriter_fourcc(*'XVID')  # Use XVID codec
```

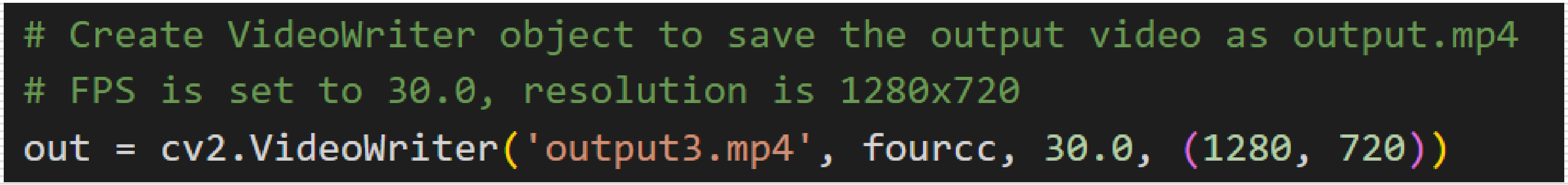
- **fourcc = cv2.VideoWriter_fourcc(*'MP4V’)**
  - 表示 MP4 編碼格示，副檔名為 .mp4



```
# Define the codec
fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Use MP4V codec
```

Video Writer (3/4)

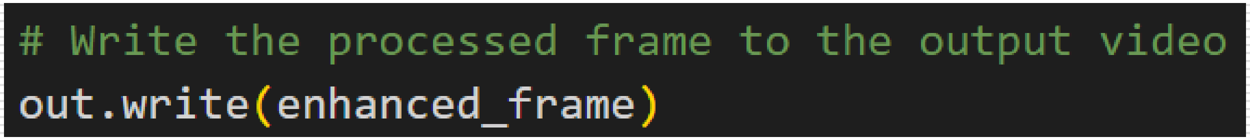
- **out = cv2.VideoWriter(filename, fourcc, fps, frameSize)**
  - filename 為輸出的影片名稱
  - fourcc 為影片編碼與解碼的格式
  - fps 為影片的幀率
  - frameSize 為影片的長度與寬度



```python
# Create VideoWriter object to save the output video as output.mp4
# FPS is set to 30.0, resolution is 1280x720
out = cv2.VideoWriter('output3.mp4', fourcc, 30.0, (1280, 720))
```

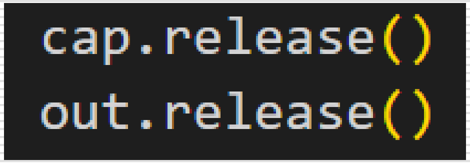
Video Writer (4/4)

- **cv2.VideoWriter.write(frame)**
  - 讀取的 frame 寫入影片



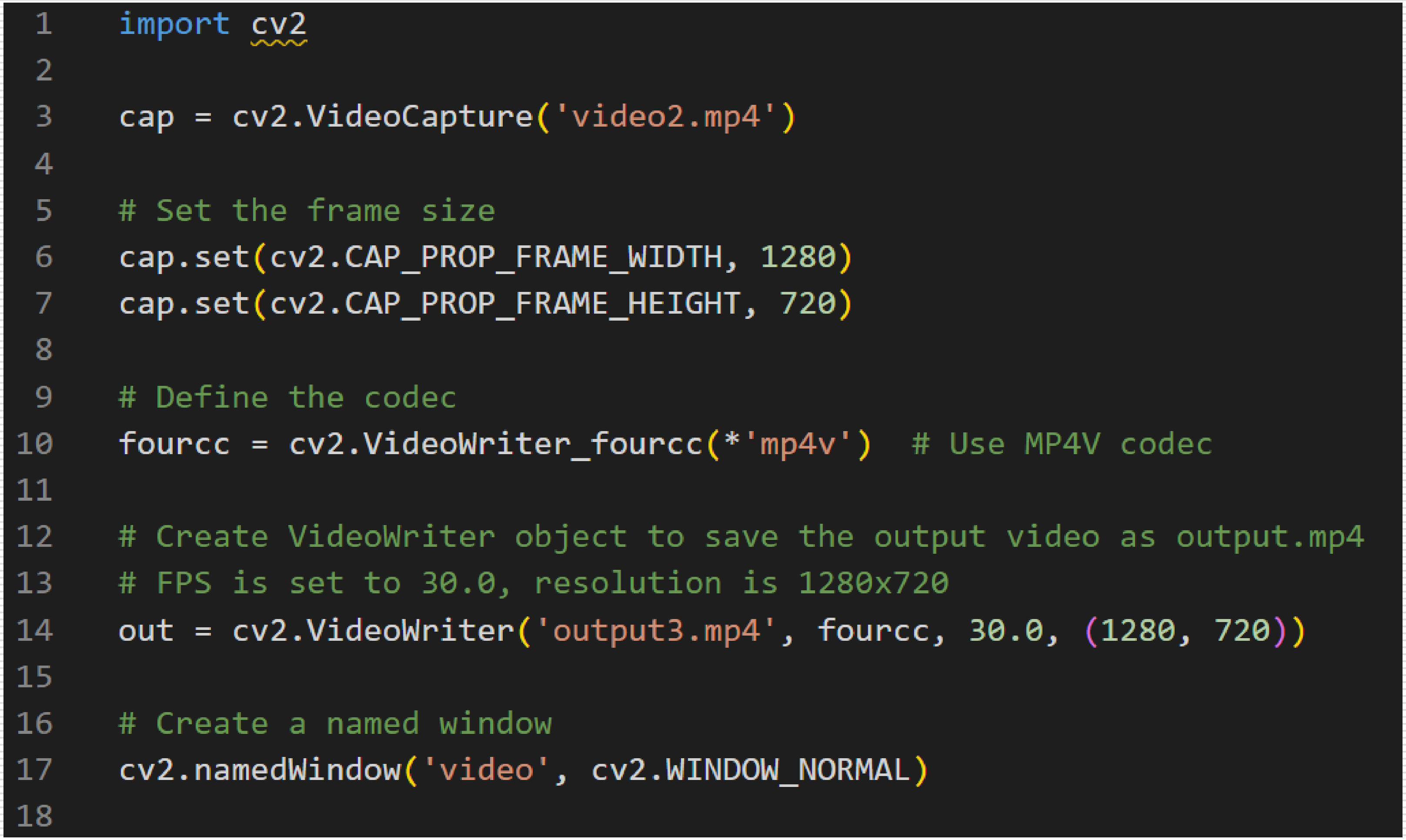
```
# Write the processed frame to the output video
out.write(enhanced_frame)
```

- **cv2.VideoWriter.release()**
  - 不需要 cv2.VideoWriter 類別物件時，要將其釋放



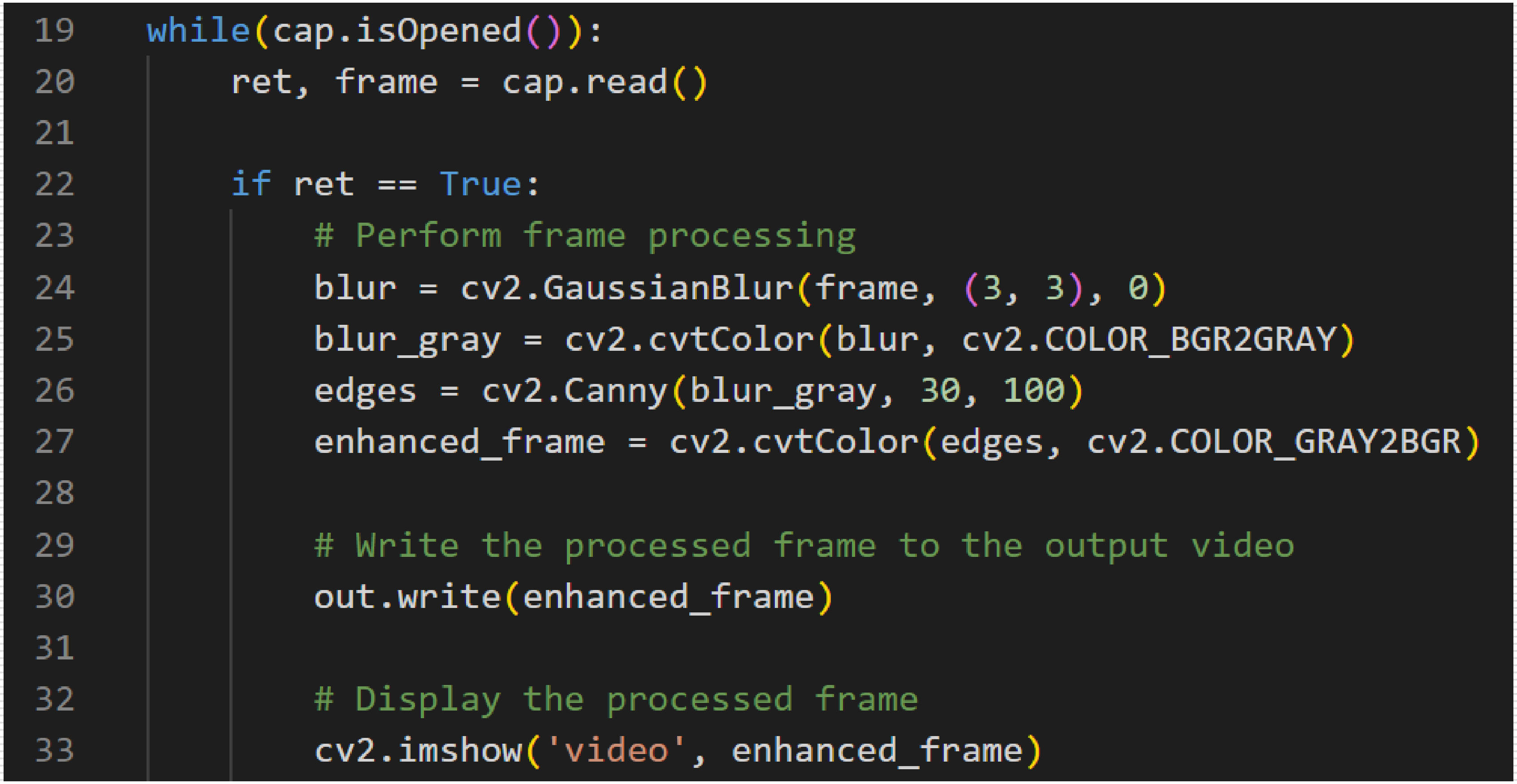
```
cap.release()
out.release()
```

Example (1/4)



```python
import cv2

cap = cv2.VideoCapture('video2.mp4')

# Set the frame size
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

# Define the codec
fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Use MP4V codec

# Create VideoWriter object to save the output video as output.mp4
# FPS is set to 30.0, resolution is 1280x720
out = cv2.VideoWriter('output3.mp4', fourcc, 30.0, (1280, 720))

# Create a named window
cv2.namedWindow('video', cv2.WINDOW_NORMAL)

```

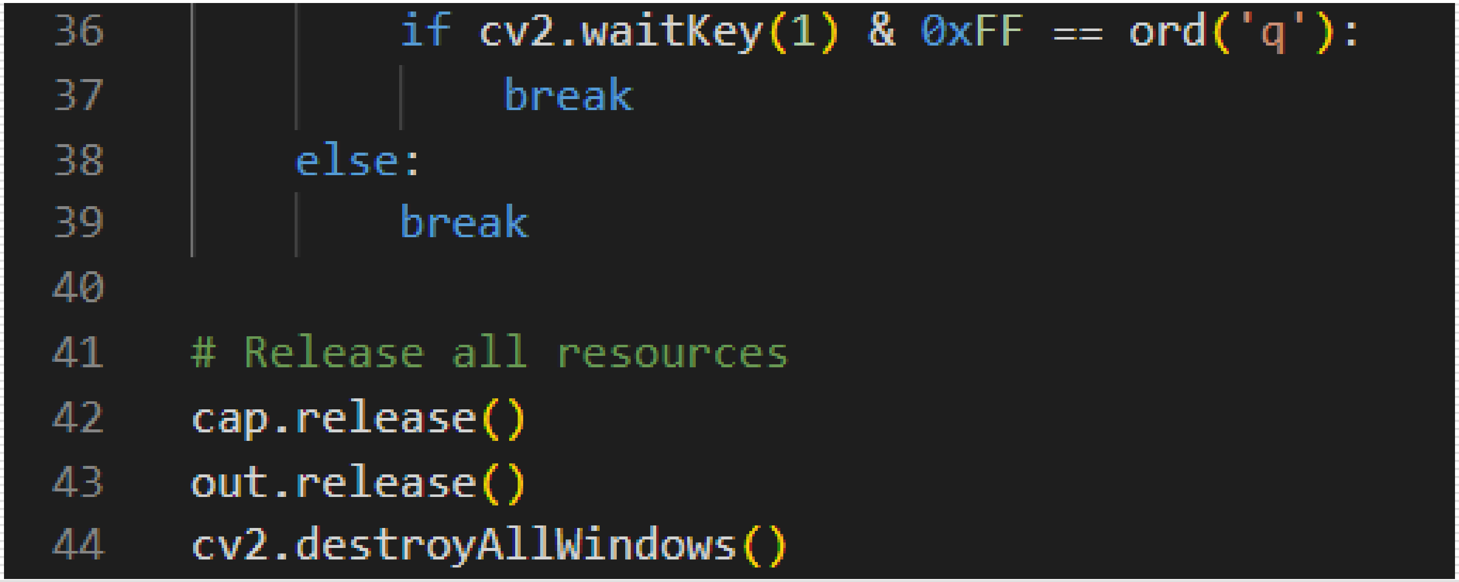
Example (2/4)



```python
    while(cap.isOpened()):
        ret, frame = cap.read()

        if ret == True:
            # Perform frame processing
            blur = cv2.GaussianBlur(frame, (3, 3), 0)
            blur_gray = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)
            edges = cv2.Canny(blur_gray, 30, 100)
            enhanced_frame = cv2.cvtColor(edges, cv2.COLOR_GRAY2BGR)

            # Write the processed frame to the output video
            out.write(enhanced_frame)

            # Display the processed frame
            cv2.imshow('video', enhanced_frame)
```

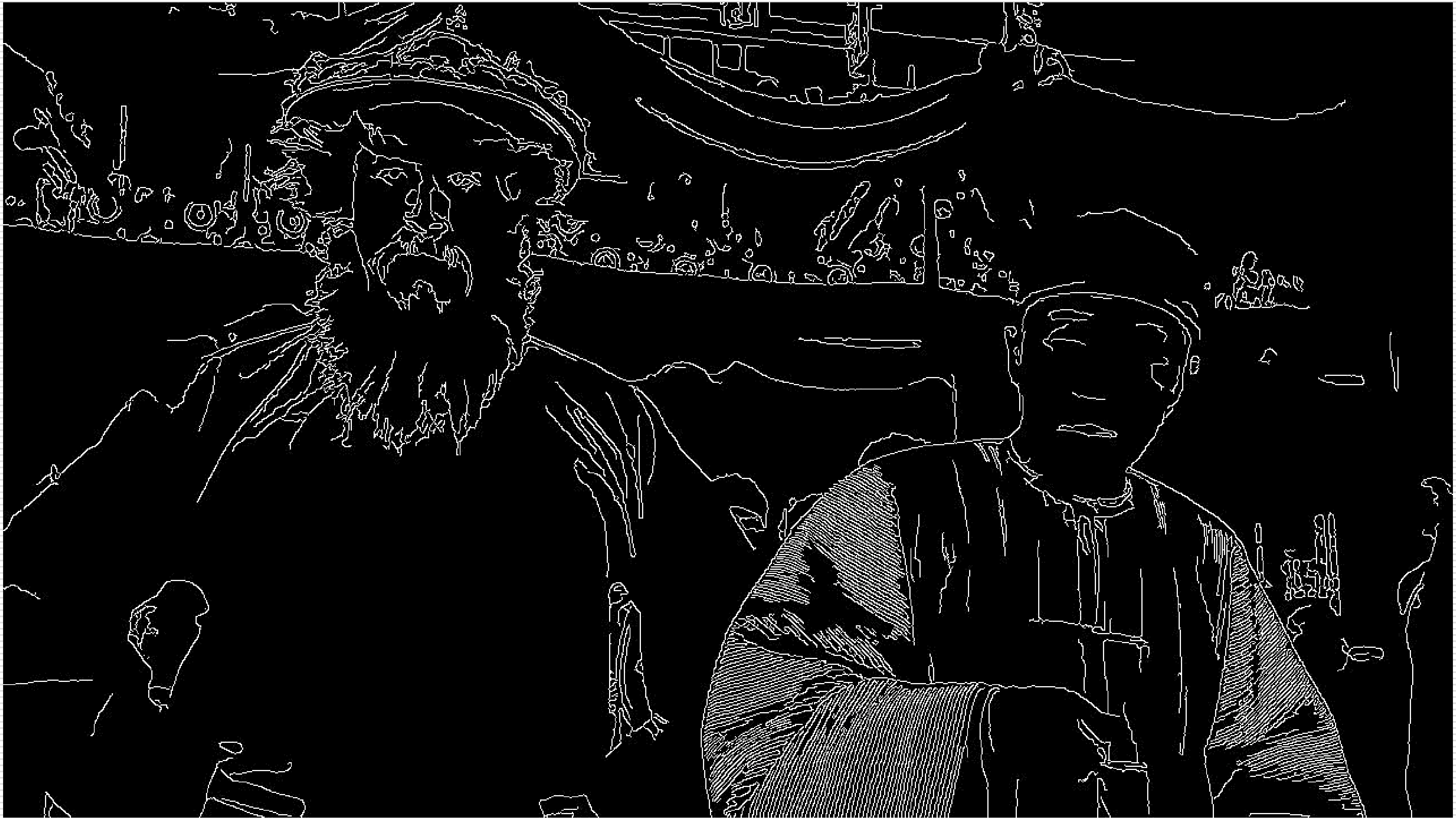
Example (3/4)

如果按下 'q' 鍵，條件式為 True
可以在該條件下跳出迴圈



```
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

# Release all resources
cap.release()
out.release()
cv2.destroyAllWindows()
```

Example (4/4)



Python Image Library(PIL)

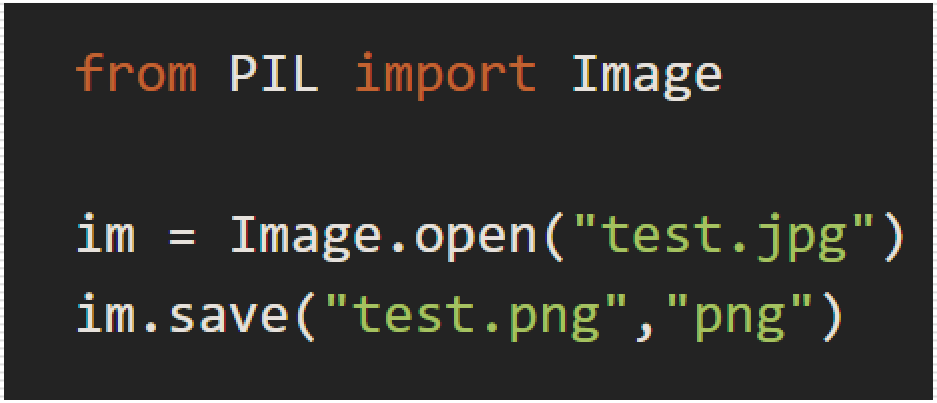
PIL (Pillow)

- PIL: a python image library, RGB
  - (cv2.IMREAD_COLOR : BGR)
- **$conda install Pillow**
- **From PIL import Image**
- **img = Image.open(“test.jpg”)**
- **img.save(“test.png”,”png”)**

PIL vs. Pillow

- PIL: Python Imaging Library
  - PIL是一個方便的python圖像處理庫，功能非常強大，曾經一度被認為是python平台事實上的圖像處理標準庫，不過Python 2.7以後不再支持

- **Pillow**
  - Pillow是基於PIL模塊fork的一個派生分支
  - 可以用來轉檔、調色、濾鏡、浮水印等等的功能
  - 雖然是pillow，但是import寫法依然是from PIL



```
from PIL import Image

im = Image.open("test.jpg")
im.save("test.png","png")
```

OpenCV vs. PIL (1/2)

- **讀取圖像的通道順序區別**
  - OpenCV讀取圖像，通道順序是：BGR
  - Pillow讀取圖像，通道順序是：RGB

- 獲得圖像shape區別
  - OpenCV.shape是(height, width, channel)
  - Pillow.size是(width, height)

OpenCV vs. PIL (2/2)

	優點	缺點
OpenCV	由C和C++編寫，跨平台，處理時間較快	對顯示中文支持較差
Pillow	跨平台，對顯示中文字體有著很好的支持，可以讀取多種格式的圖像	處理時間較慢

NumPy

- NumPy : a python data analysis library

- **$ import numpy as np**

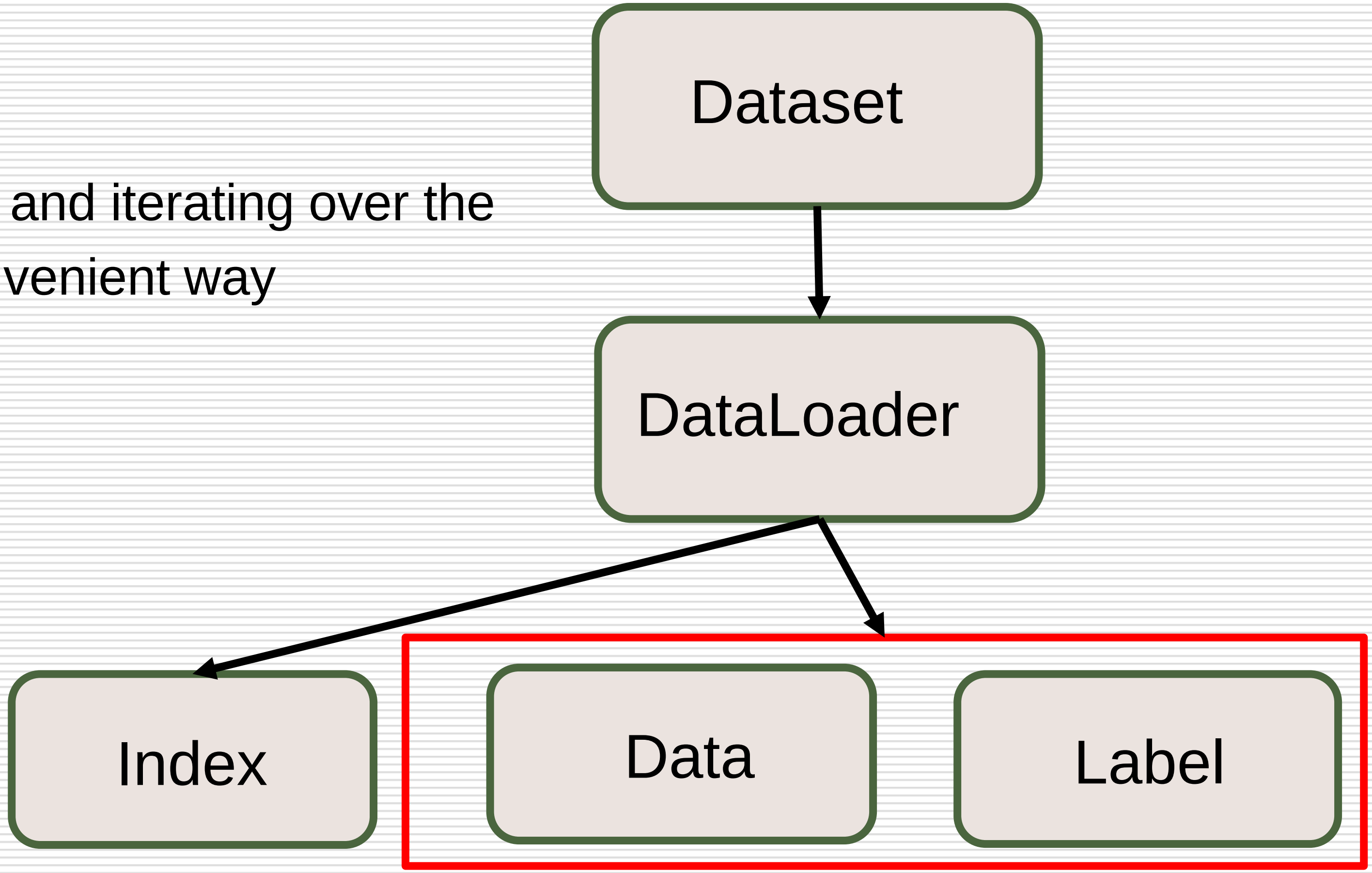
- **$ np.loadtxt (‘data.txt’)**

- **$ image = Image.open(‘IMAGE.jpg')**

PyTorch Load Data

Data Load in PyTorch

- Dataset
  - It provides an interface to access and manipulate the data in a uniform way
- DataLoader
  - helps in loading and iterating over the the dataset in a convenient way



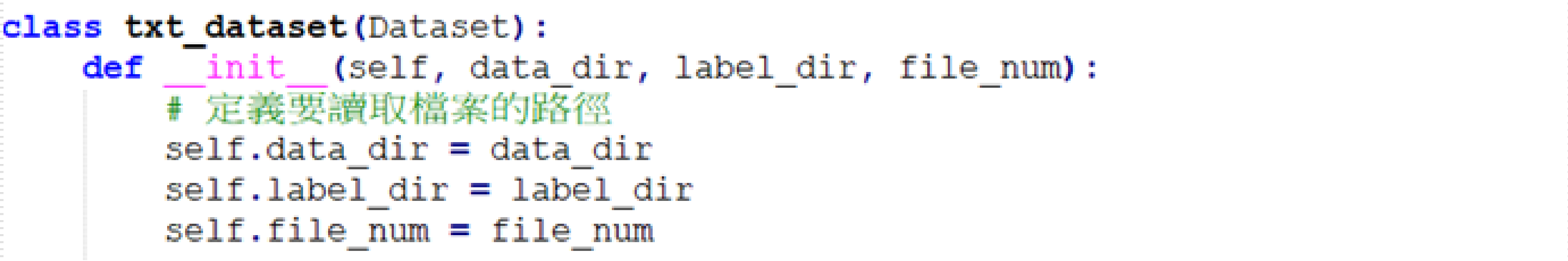
Dataset (1/4)

- **$ from torch.utils.data import Dataset, DataLoader**

- torch.utils.data.Dataset
  - An abstract class representing a Dataset
  - All the user-defined dataset must inherit the class
  - All subclass should override **“__getitem__()”** and **“__len__()”**
  - Often override **“__init__()”**

Dataset (2/4)

- __init__() :
  - Define objects in class

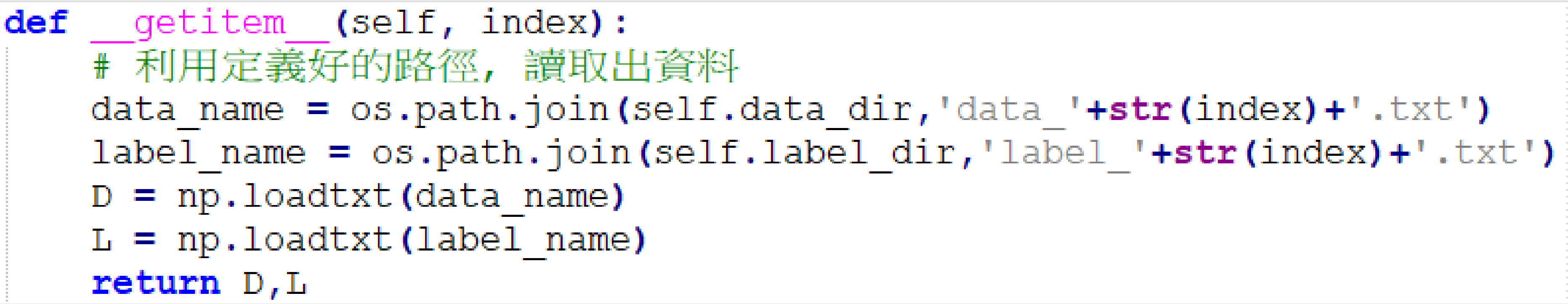


```python
class txt_dataset(Dataset):
    def __init__(self, data_dir, label_dir, file_num):
        # 定義要讀取檔案的路徑
        self.data_dir = data_dir
        self.label_dir = label_dir
        self.file_num = file_num
```

Dataset (3/4)

- __getitem__()
  - Define the way to get items in dataset

- Example for image dataset with (image, label)

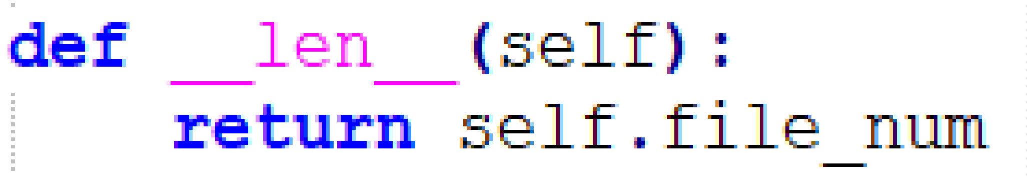


```python
def __getitem__(self, index):
    # 利用定義好的路徑，讀取出資料
    data_name = os.path.join(self.data_dir,'data_'+str(index)+'.txt')
    label_name = os.path.join(self.label_dir,'label_'+str(index)+'.txt')
    D = np.loadtxt(data_name)
    L = np.loadtxt(label_name)
    return D,L
```

Dataset (4/4)

- __len__() : determine the number of samples in the dataset

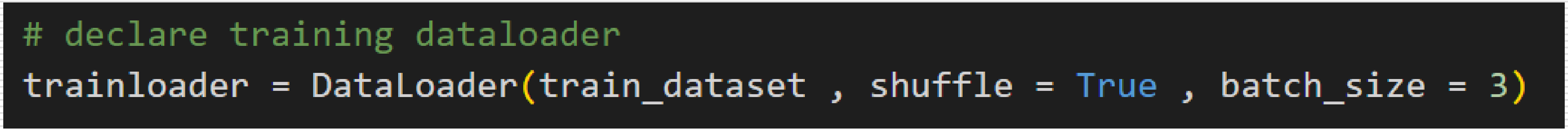
- Example :



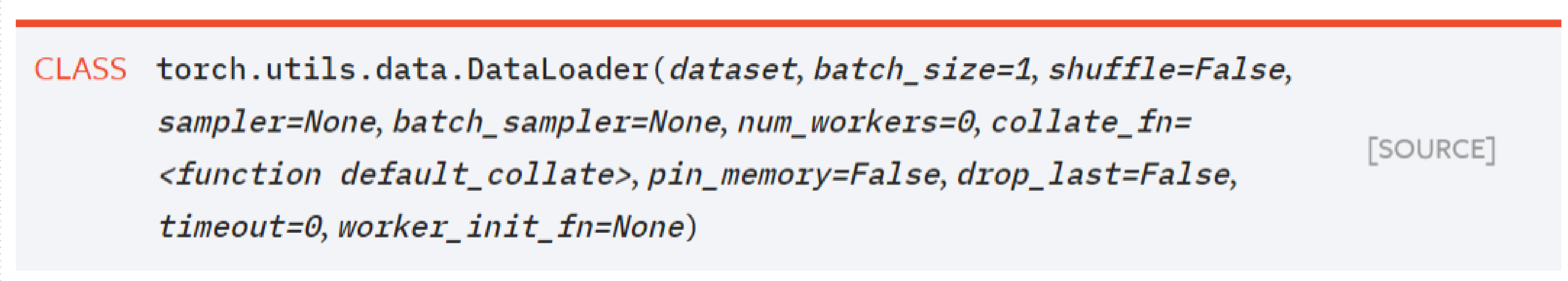
```python
def __len__(self):
    return self.file_num
```

DataLoader

- The container of dataset
  - dataset
  - batch_size
    - 指定了在每次模型更新參數時一次性處理的資料樣本數量
  - Shuffle(洗牌)
    - 每個訓練迭代中樣本的呈現順序將是不同的



```
# declare training dataloader
trainloader = DataLoader(train_dataset , shuffle = True , batch_size = 3)
```



CLASS torch.utils.data.DataLoader(*dataset*, *batch_size=1*, *shuffle=False*, *sampler=None*, *batch_sampler=None*, *num_workers=0*, *collate_fn=<function default_collate>*, *pin_memory=False*, *drop_last=False*, *timeout=0*, *worker_init_fn=None*) [SOURCE]

Use of DataLoader

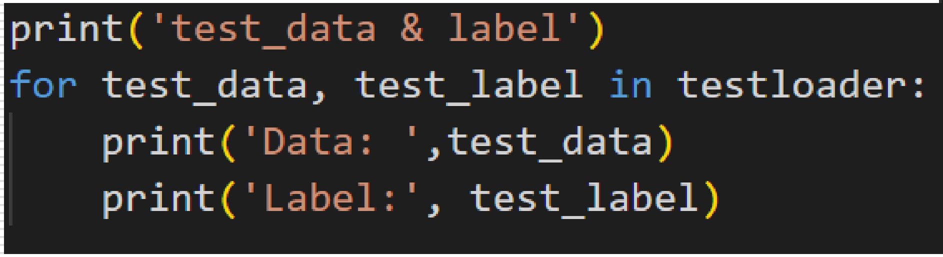
- for (data, label) in DataLoader :

  …

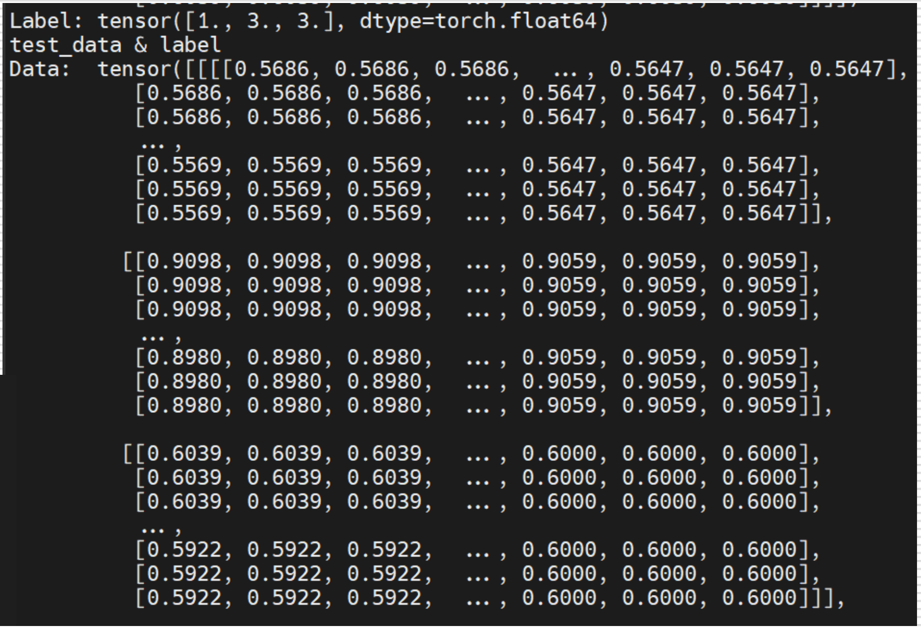
- for i, (data, label) in enumerate(DataLoader) :

  …

  # i is the index of data/batch

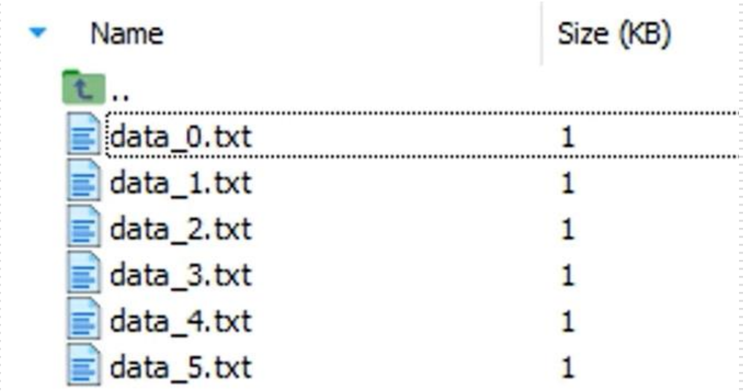


```
print('test_data & label')
for test_data, test_label in testloader:
    print('Data: ',test_data)
    print('Label:', test_label)
```

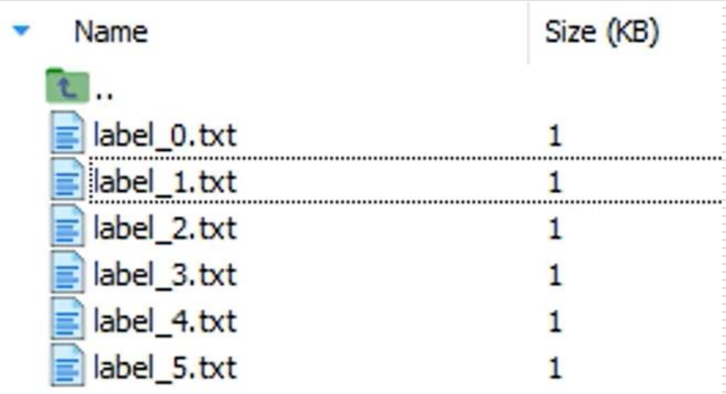


```
Label: tensor([1., 3., 3.], dtype=torch.float64)
test_data & label
Data:  tensor([[[[0.5686, 0.5686, 0.5686,  ..., 0.5647, 0.5647, 0.5647],
          [0.5686, 0.5686, 0.5686,  ..., 0.5647, 0.5647, 0.5647],
          [0.5686, 0.5686, 0.5686,  ..., 0.5647, 0.5647, 0.5647],
          ...,
          [0.5569, 0.5569, 0.5569,  ..., 0.5647, 0.5647, 0.5647],
          [0.5569, 0.5569, 0.5569,  ..., 0.5647, 0.5647, 0.5647],
          [0.5569, 0.5569, 0.5569,  ..., 0.5647, 0.5647, 0.5647]],

         [[0.9098, 0.9098, 0.9098,  ..., 0.9059, 0.9059, 0.9059],
          [0.9098, 0.9098, 0.9098,  ..., 0.9059, 0.9059, 0.9059],
          [0.9098, 0.9098, 0.9098,  ..., 0.9059, 0.9059, 0.9059],
          ...,
          [0.8980, 0.8980, 0.8980,  ..., 0.9059, 0.9059, 0.9059],
          [0.8980, 0.8980, 0.8980,  ..., 0.9059, 0.9059, 0.9059],
          [0.8980, 0.8980, 0.8980,  ..., 0.9059, 0.9059, 0.9059]],

         [[0.6039, 0.6039, 0.6039,  ..., 0.6000, 0.6000, 0.6000],
          [0.6039, 0.6039, 0.6039,  ..., 0.6000, 0.6000, 0.6000],
          [0.6039, 0.6039, 0.6039,  ..., 0.6000, 0.6000, 0.6000],
          ...,
          [0.5922, 0.5922, 0.5922,  ..., 0.6000, 0.6000, 0.6000],
          [0.5922, 0.5922, 0.5922,  ..., 0.6000, 0.6000, 0.6000],
          [0.5922, 0.5922, 0.5922,  ..., 0.6000, 0.6000, 0.6000]]],
```

Example (1/3)



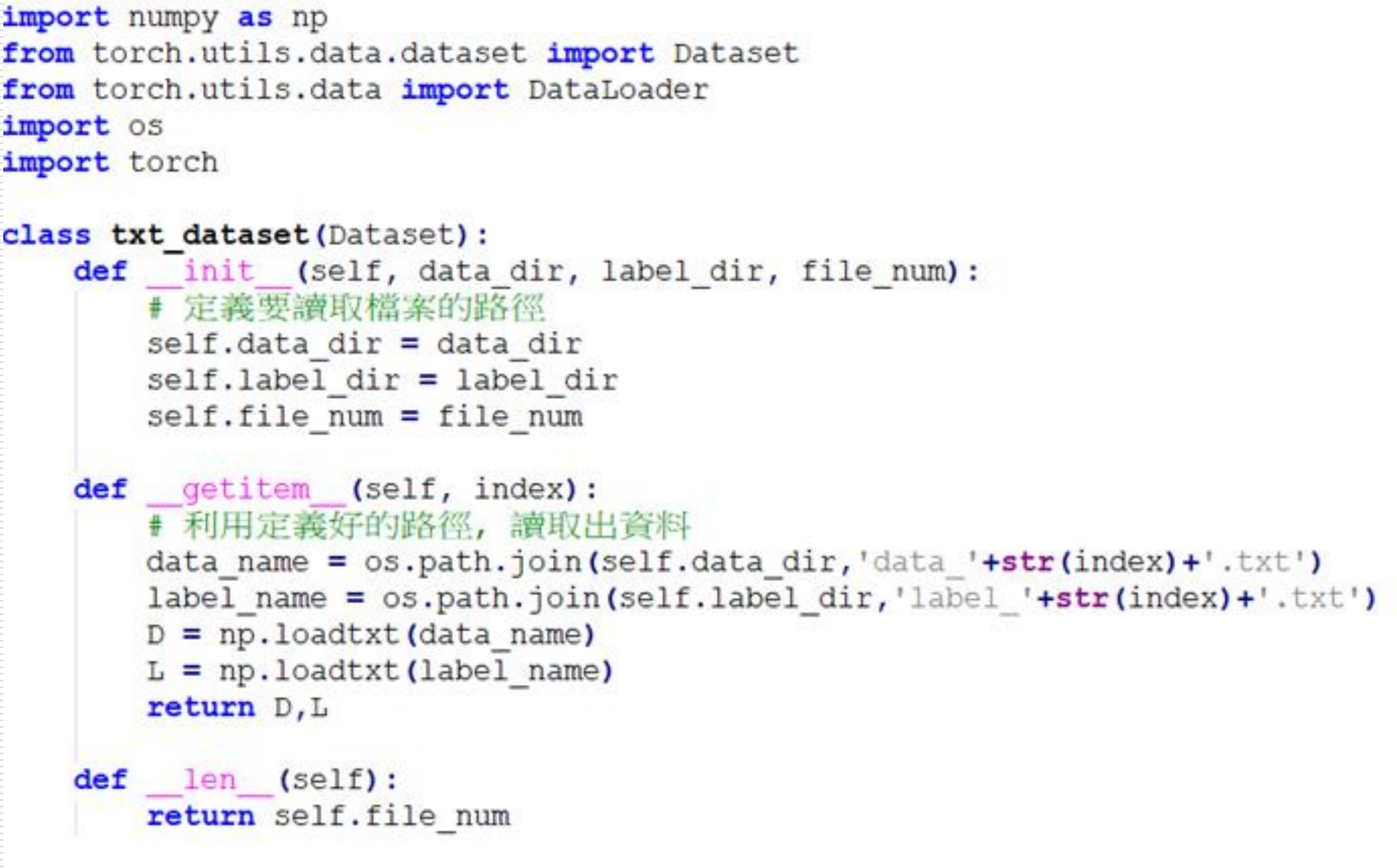
Name	Size (KB)
..	
data_0.txt	1
data_1.txt	1
data_2.txt	1
data_3.txt	1
data_4.txt	1
data_5.txt	1



Name	Size (KB)
..	
label_0.txt	1
label_1.txt	1
label_2.txt	1
label_3.txt	1
label_4.txt	1
label_5.txt	1

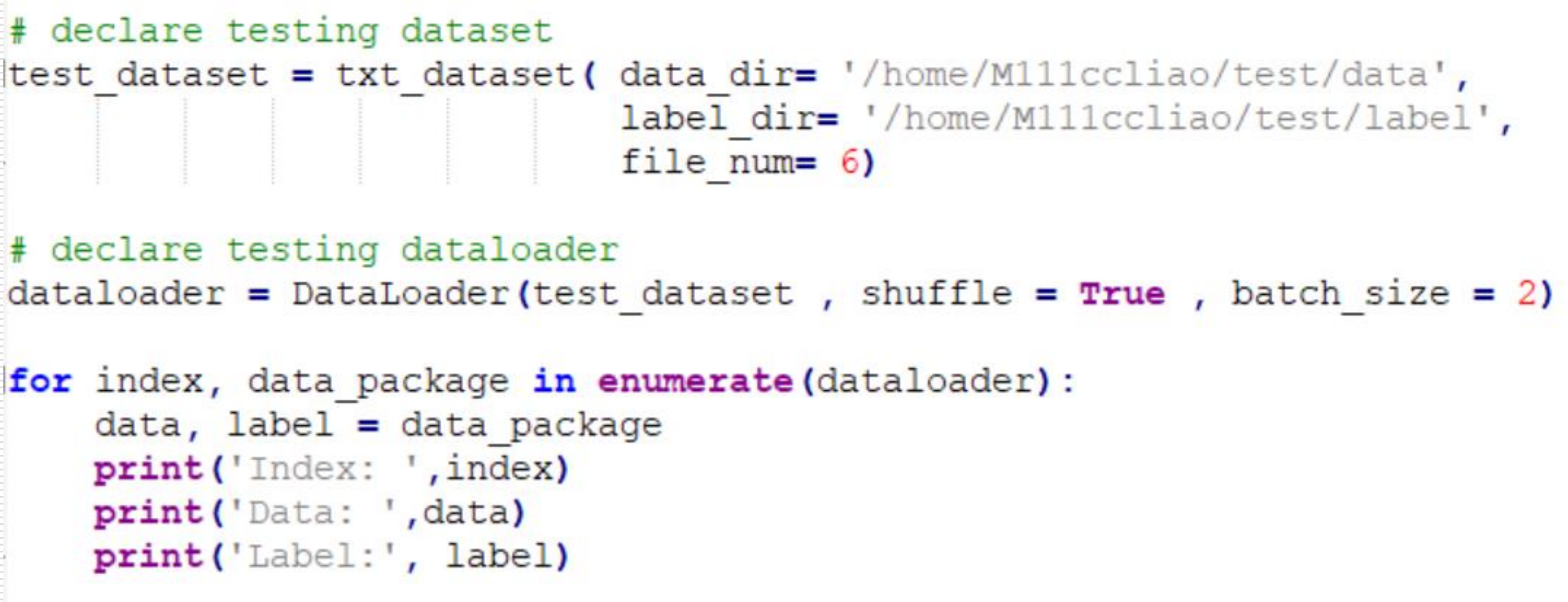
Data	1	2	3	4	5	6
Label	7	8	9	10	11	12

Example (2/3)



```
import numpy as np
from torch.utils.data.dataset import Dataset
from torch.utils.data import DataLoader
import os
import torch

class txt_dataset(Dataset):
    def __init__(self, data_dir, label_dir, file_num):
        # 定義要讀取檔案的路徑
        self.data_dir = data_dir
        self.label_dir = label_dir
        self.file_num = file_num

    def __getitem__(self, index):
        # 利用定義好的路徑，讀取出資料
        data_name = os.path.join(self.data_dir,'data_'+str(index)+'.txt')
        label_name = os.path.join(self.label_dir,'label_'+str(index)+'.txt')
        D = np.loadtxt(data_name)
        L = np.loadtxt(label_name)
        return D,L

    def __len__(self):
        return self.file_num
```

Example (3/3)



```
# declare testing dataset
test_dataset = txt_dataset( data_dir= '/home/M111ccliao/test/data',
                            label_dir= '/home/M111ccliao/test/label',
                            file_num= 6)

# declare testing dataloader
dataloader = DataLoader(test_dataset , shuffle = True , batch_size = 2)

for index, data_package in enumerate(dataloader):
    data, label = data_package
    print('Index: ',index)
    print('Data: ',data)
    print('Label:', label)
```

Preprocessing

Torchvision

- Often use **torchvision.transforms** to preprocess dataset
  - torchvision：pytorch提供好用的圖片處理工具

- $ **from torchvision import datasets, transforms**

- Model use tensor to accelerate on GPU
- Transforms take PIL(python image library) image as input

- Read image in PIL
- … preprocess …
- Transform into tensor

Transforms of Torchvision

- Introduce 4 kinds of transforms
  - Transforms on PIL image
  - Transforms on tensor
  - Conversion transform
  - Compose

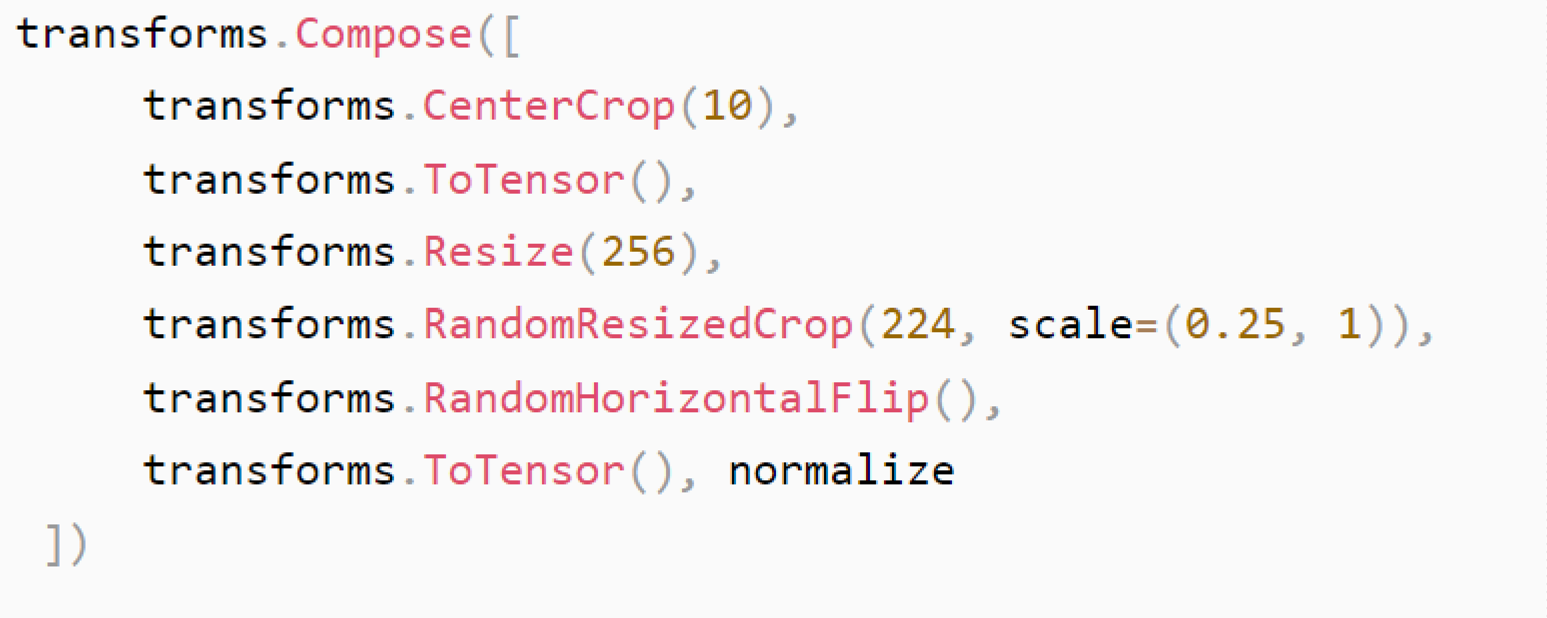
Transforms on PIL Image

- Data augmentation with torchvision.transforms
- transforms.Resize(size, interpolate)
  - Often use in match model input
- transforms.CenterCrop(size)
- transforms.RandomCrop(size, padding, …)
- transforms.Pad(padding, fill=0,padding_mode="constant")
- transforms.RandomHorizontalFlip(p=0.5)

Compose Transforms

- **torchvision.transforms.Compose(transforms)**

- Compose transforms (in sequence) into one



```
transforms.Compose([
     transforms.CenterCrop(10),
     transforms.ToTensor(),
     transforms.Resize(256),
     transforms.RandomResizedCrop(224, scale=(0.25, 1)),
     transforms.RandomHorizontalFlip(),
     transforms.ToTensor(), normalize
 ])
```

Image Resize

- **transforms.Resize(size, interpolation)**
  - **Often used in data preprocessing**
- size: the height and width of the image
  - ex: size = 100, represent height = width = 100
  - ex: size = (160,80) = (height, width)
- Interpolation
  - used to estimate the pixel values for the new image size
  - ex: PIL.Image.BILINEAR(default), PIL.Image.BICUBIC
- Ex: transforms.Resize(160, 80)

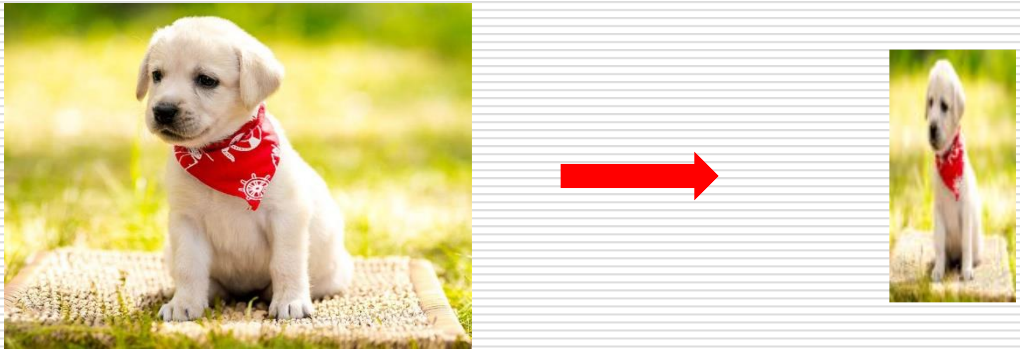


Image CenterCrop

- **transforms.CenterCrop(size)**

- Image cutting is performed by extending the set size range from the center point of the Image.

- Ex: transforms.CenterCrop(300)

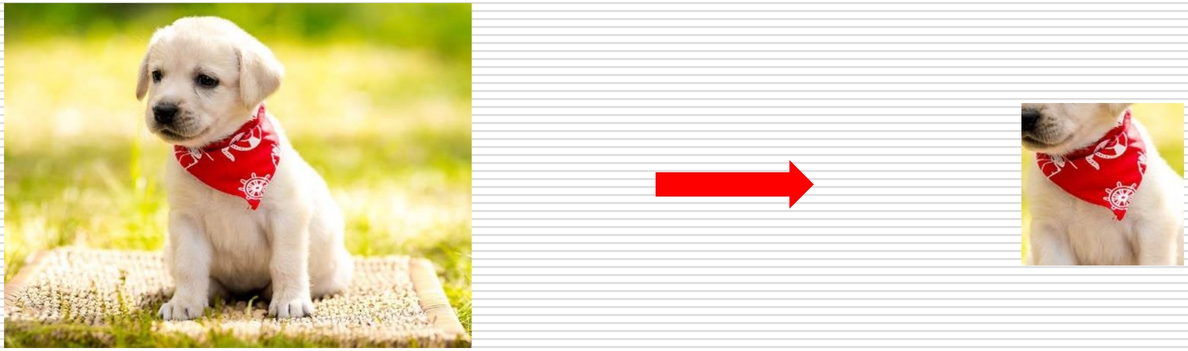


Image RandomCrop

- **transforms.RandomCrop(size, padding, …)**
- Randomly cut out a piece of image
- Ex: transforms.RandomCrop(300, 500)

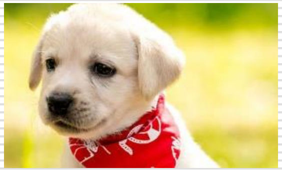
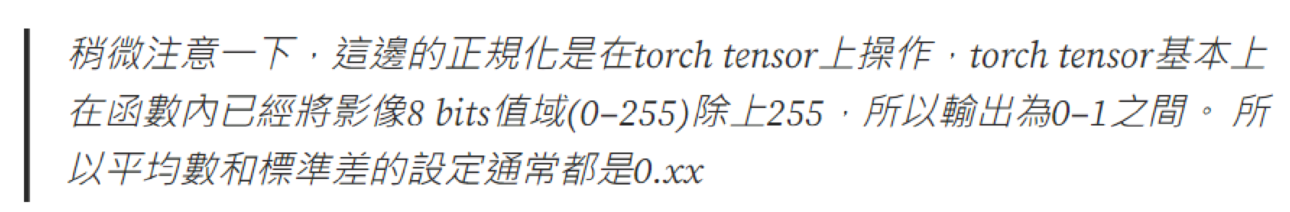


Image Normalize

- **transforms.Normalize(mean, std)**
  - mean : average per channel
  - std: standard deviation per channel

- Ex: transforms.Normalize([0.5, 0.5, 0.5], [0.1, 0.1, 0.1])



> 稍微注意一下，這邊的正規化是在torch tensor上操作，torch tensor基本上在函數內已經將影像8 bits值域(0–255)除上255，所以輸出為0–1之間。 所以平均數和標準差的設定通常都是0.xx

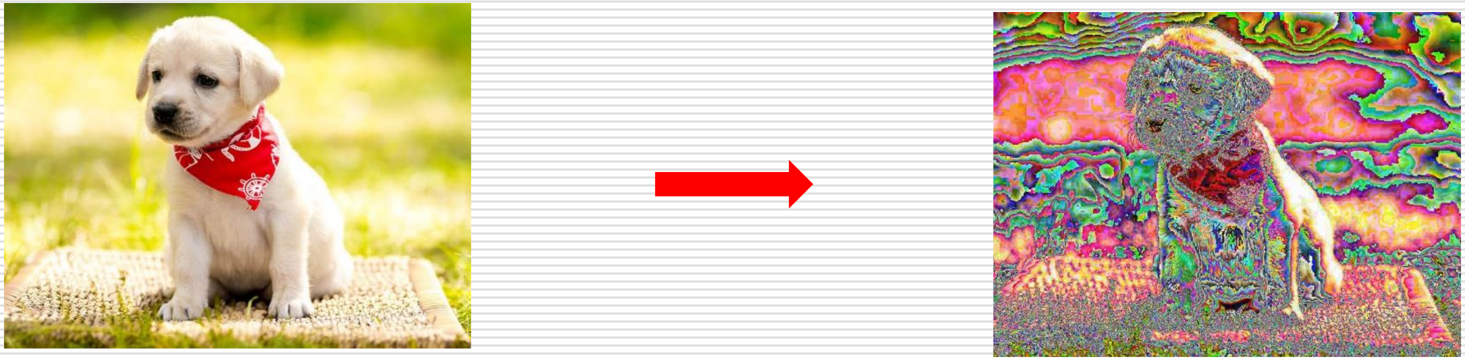
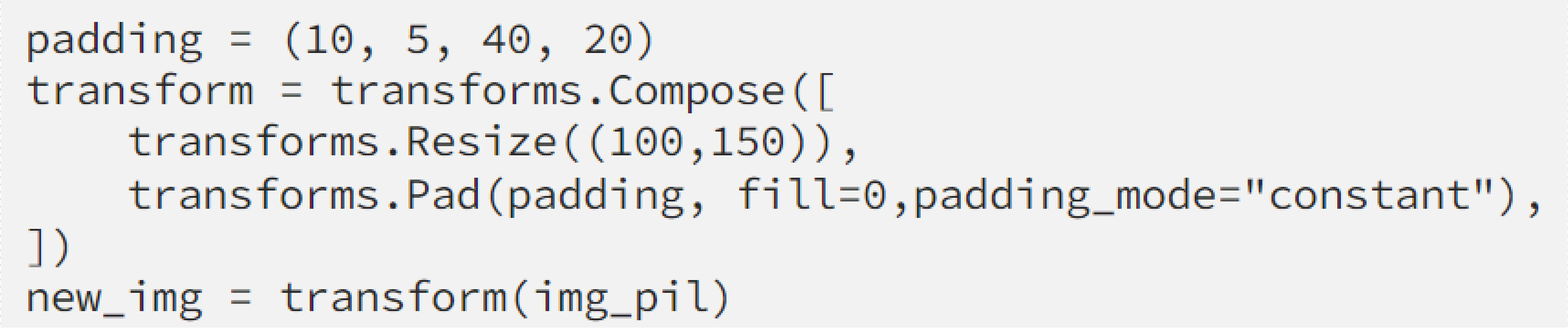


Image Pad (1/3)

- **transforms.Pad(padding, fill=0,padding_mode="constant")**
- The padding width and height are extended from the outside of the image, and the padding value is the pad value.
- Parameter:
  - padding: padding width and height, can set each side separately
    - (left, up, right, down)
  - fill: filled value, only required when padding_mode is constant
    - ex: fill=0, pad with black
  - padding_mode:
    - constant
    - edge
    - reflect
    - symmetric

Image Pad (2/3)



```
padding = (10, 5, 40, 20)
transform = transforms.Compose([
    transforms.Resize((100,150)),
    transforms.Pad(padding, fill=0,padding_mode="constant"),
])
new_img = transform(img_pil)
```

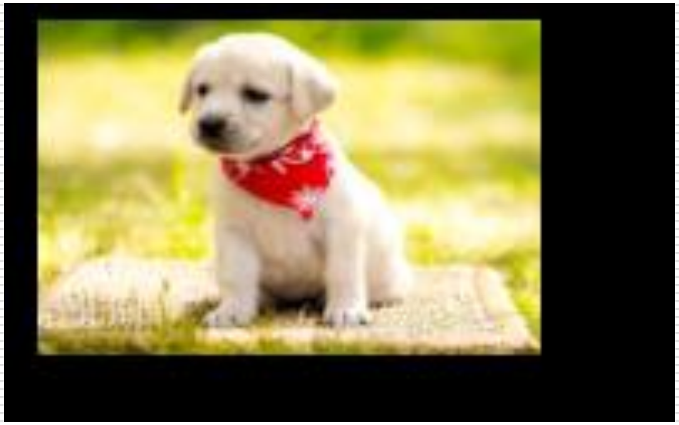
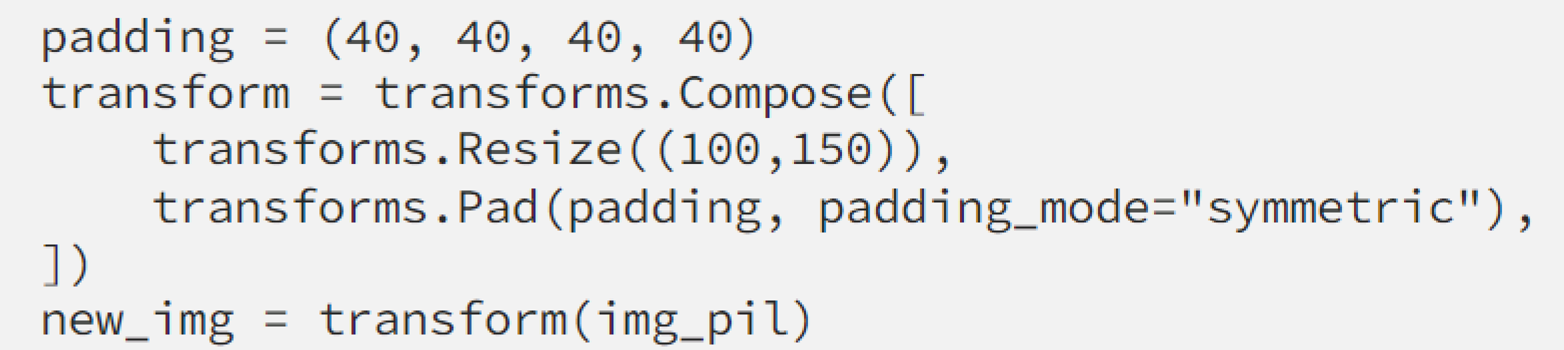
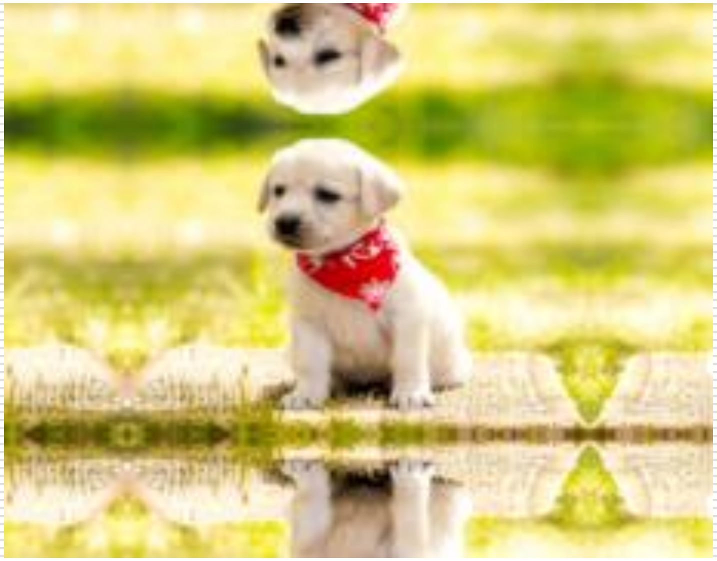


Image Pad (3/3)

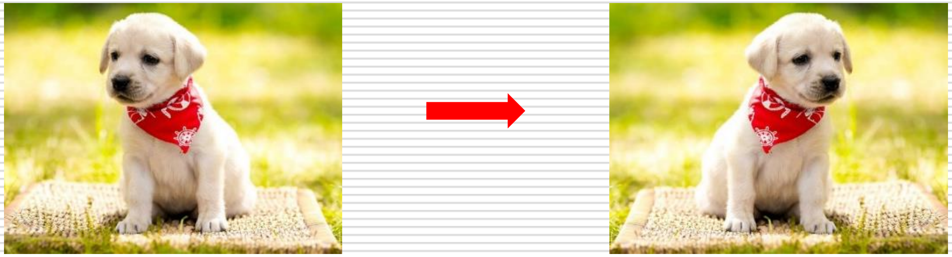


```
padding = (40, 40, 40, 40)
transform = transforms.Compose([
    transforms.Resize((100,150)),
    transforms.Pad(padding, padding_mode="symmetric"),
])
new_img = transform(img_pil)
```

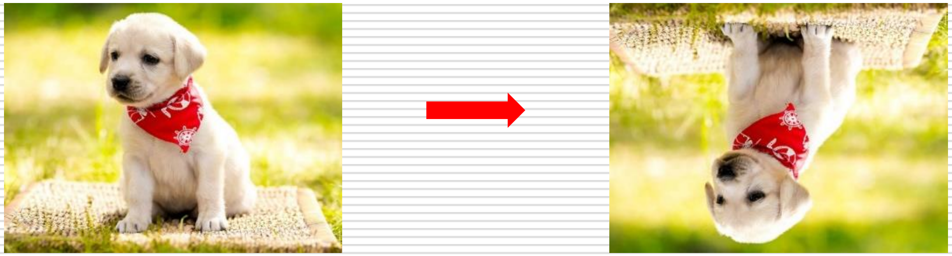


RandomFlip

- **transforms.RandomHorizontalFlip(p=0.7)**
  - Parameter: p(float) probability of the image being horizontal flipped.
    - Default value is 0.5



- **transforms.RandomVerticalFlip(p=0.7)**
  - Parameter: p(float) probability of the image being vertical flipped.
    - Default value is 0.5



Conversion Between PIL and Tensor

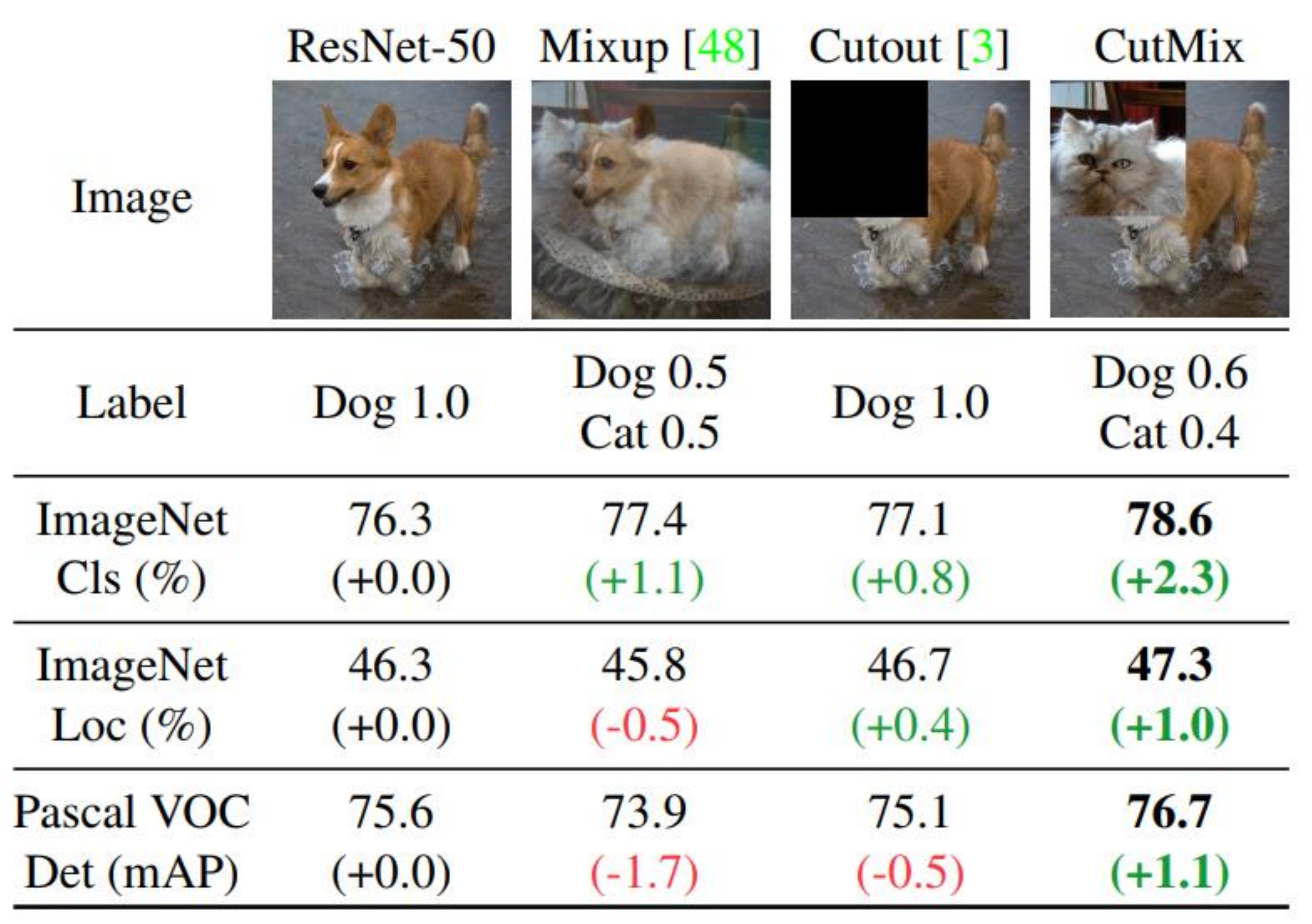
- **transforms.ToPILImage(mode=None)**
  - Convert tensor or ndarray into PIL image

- **transforms.ToTensor()**
  - Convert PIL image into tensor

Advanced Data Augmentation Methods

- Mixup
  - Generates a weighted combinations of random image pairs from the training data
- Cutout
  - Randomly masks out square regions of input during training
- CutMix
  - Replace the removed regions with a patch from another image

Advanced Data Augmentation Methods



	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	**78.6** **(+2.3)**
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	**47.3** **(+1.0)**
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	**76.7** **(+1.1)**

classification →

localization →

detection →

References (1/3)

- Basic operations (read/write/svae) Python and OpenCV
  - https://blog.gtwang.org/programming/opencv-basic-image-read-and-write-tutorial/
- OpenCV Tutorials
  - https://docs.opencv.org/master/d9/df8/tutorial_root.html
- OpenCV 擷取網路攝影機串流影像，處理並寫入影片檔案教學
  - https://blog.gtwang.org/programming/opencv-webcam-video-capture-and-file-write-tutorial/
- Torch.utils.data master documentation
  - https://pytorch.org/docs/stable/data.html

References (2/3)

- Torchvision.transforms master documentation
  - https://pytorch.org/vision/0.9/transforms.html#

- Pytorch提供之torchvision data augmentation技巧
  - https://chih-sheng-huang821.medium.com/03-pytorch-dataaug-a712a7a7f55e

- PyTorch 怎麼讀取資料? Dataset and DataLoader
  - https://ithelp.ithome.com.tw/articles/10277163

References (3/3)

- Mixup
  - https://arxiv.org/abs/1710.09412v2
  - https://github.com/facebookresearch/mixup-cifar10

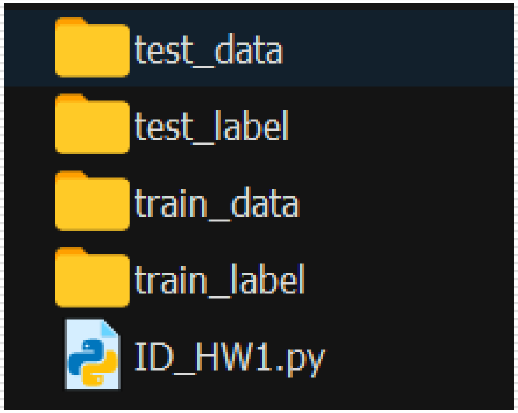
- Cutout
  - https://arxiv.org/abs/1708.04552
  - https://github.com/uoguelph-mlrg/Cutout

- CutMix
  - https://arxiv.org/abs/1905.04899
  - https://github.com/clovaai/CutMix-PyTorch

Homework

Homework (1/3)

- 題目: Dataset and Dataloader
- 在HW1中有train/test dataset
  - data格式: jpg
  - label格式: txt

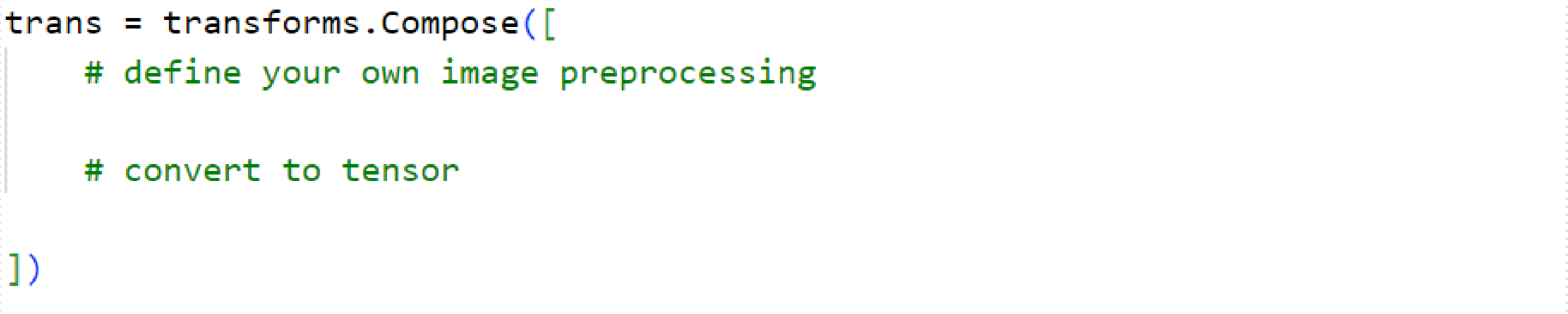


test_data
test_label
train_data
train_label
ID_HW1.py

- 請各位完成ID_HW1.py挖空的部分
  - Image preprocessing
    - 方式不限
  - Dataset
    - **Datasets路徑請用相對路徑 ('./train_data')**
  - DataLoader

Homework (2/3)

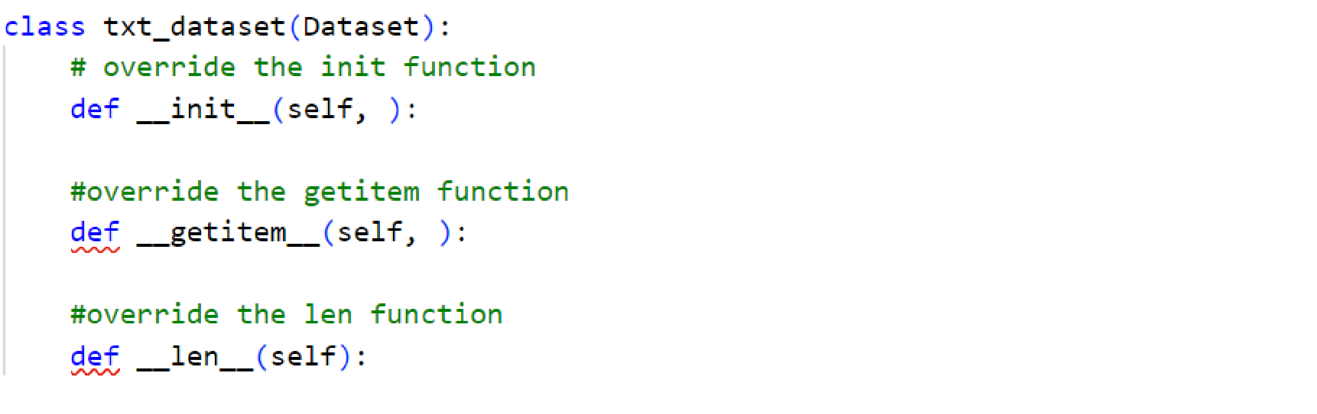
- 於eng05 server(140.113.225.245)上執行
  - $ tar -xvf /DATA/AI_training_HW_2024/HW1.tar
  - 完成的作業請上傳至FB群組公告中的google表單
  - 繳交python檔就好，不要交dataset
  - 檔案名稱格式：[帳號]_HW1.py
    - Ex. M111CCLIAO_HW1.py
  - 繳交期限為 **2024/7/12 (五) 23:59**
- Image preprocessing:



```
trans = transforms.Compose([
    # define your own image preprocessing

    # convert to tensor

])
```

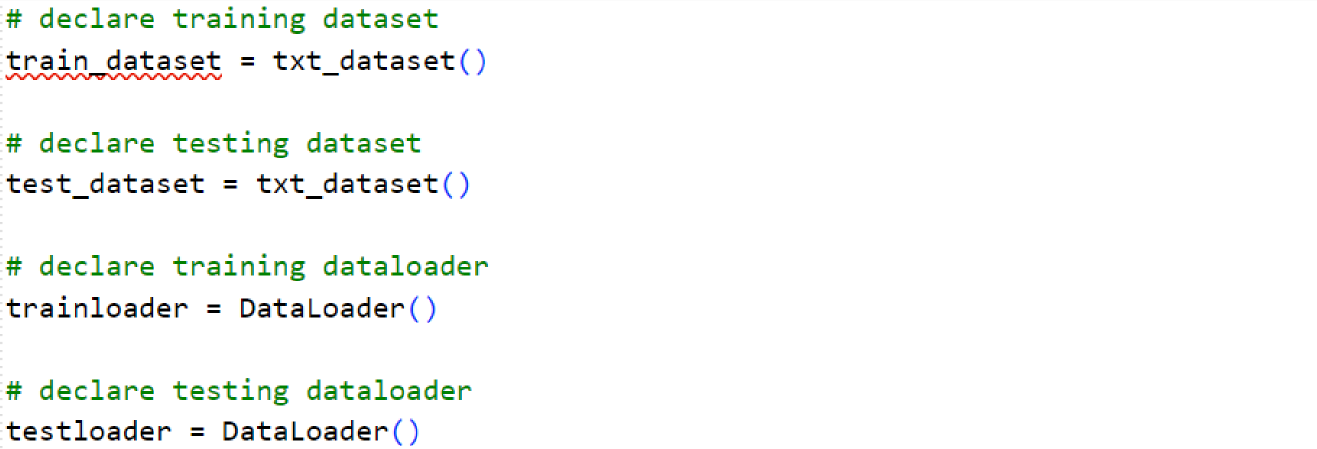
Homework (3/3)

- Dataset:



```
class txt_dataset(Dataset):
    # override the init function
    def __init__(self, ):

    #override the getitem function
    def __getitem__(self, ):

    #override the len function
    def __len__(self):
```

- DataLoader:



```
# declare training dataset
train_dataset = txt_dataset()

# declare testing dataset
test_dataset = txt_dataset()

# declare training dataloader
trainloader = DataLoader()

# declare testing dataloader
testloader = DataLoader()
```

Thank you