CNN Bonus HW

(Add 4% in the HW)

AOI Defect Detection

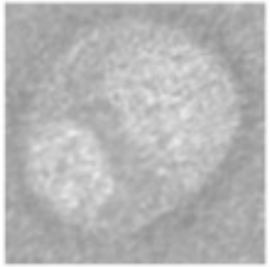
- Automated Optical Inspection (AOI) is utilized to detect defects.
- The dataset is collected from AOI.
- We want you to classify the defects.

Dataset

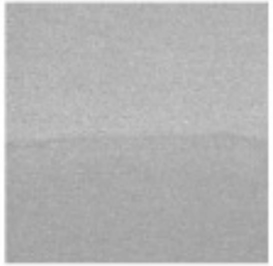
- train_images.zip : Training and testing data
- train.csv: There are two columns，ID和Label
- ID: The file name of images
- Label: The class of defect
- 0: normal
- 1: void
- 2: horizontal defect
- 3: vertical defect
- 4: edge defect
- 5: particle

Dataset

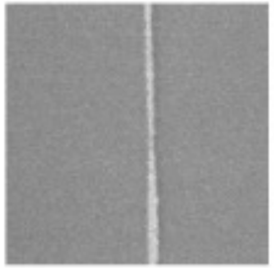
Normal



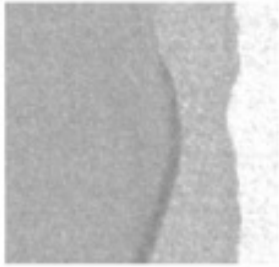
Void



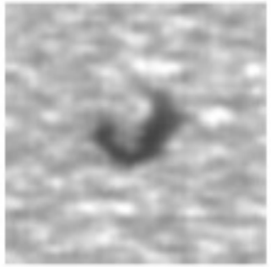
Vertical Defect



Horizontal Defect



Edge Defect

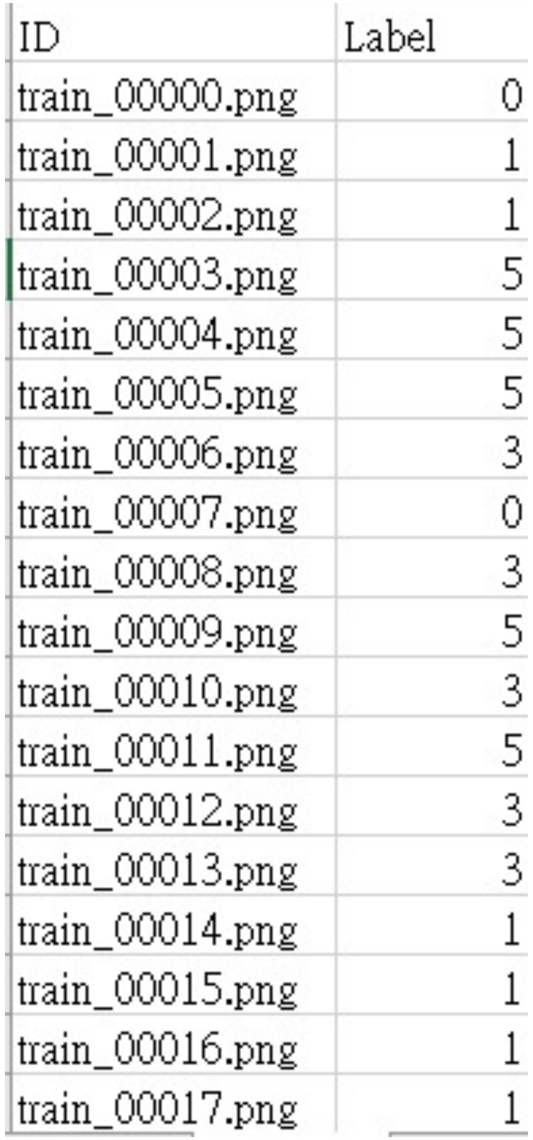


Particle

0 is normal，1 is void，2 is horizontal defect，
3 is vertical defect，4 is edge defect，5 is particle

Dataset

train.csv



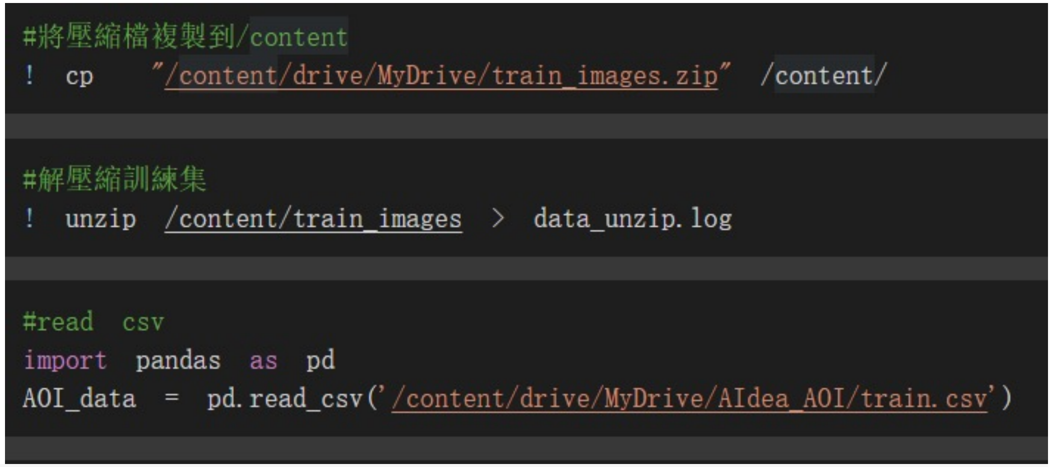
ID	Label
train_00000.png	0
train_00001.png	1
train_00002.png	1
train_00003.png	5
train_00004.png	5
train_00005.png	5
train_00006.png	3
train_00007.png	0
train_00008.png	3
train_00009.png	5
train_00010.png	3
train_00011.png	5
train_00012.png	3
train_00013.png	3
train_00014.png	1
train_00015.png	1
train_00016.png	1
train_00017.png	1

Code

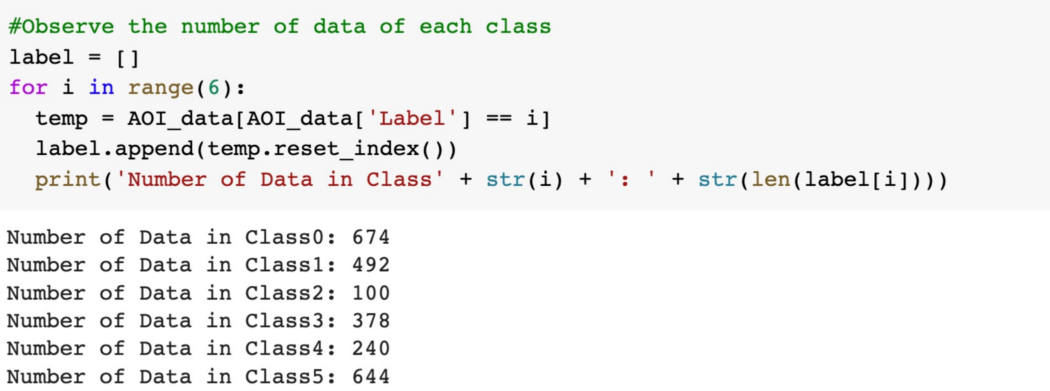
- Use AOI-ToStudents.ipynb to finish the code

Load Data

- Load the dataset and to observe the number of data of each class.



```
#將壓縮檔複製到/content
! cp ″/content/drive/MyDrive/train_images.zip″ /content/

#解壓縮訓練集
! unzip /content/train_images > data_unzip.log

#read csv
import pandas as pd
AOI_data = pd.read_csv(’/content/drive/MyDrive/AIdea_AOI/train.csv’)
```

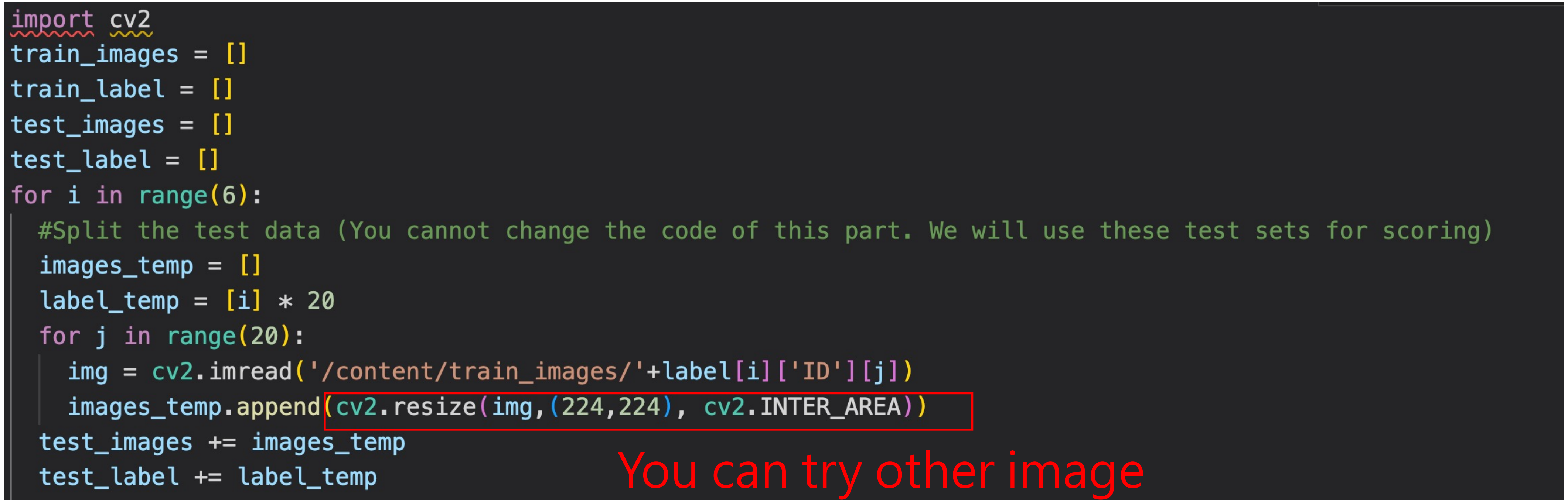


```
#Observe the number of data of each class
label = []
for i in range(6):
  temp = AOI_data[AOI_data['Label'] == i]
  label.append(temp.reset_index())
  print('Number of Data in Class' + str(i) + ': ' + str(len(label[i])))
```

```
Number of Data in Class0: 674
Number of Data in Class1: 492
Number of Data in Class2: 100
Number of Data in Class3: 378
Number of Data in Class4: 240
Number of Data in Class5: 644
```

Split test set

- Split test set (You cannot change the code of this part. We will use these test sets for scoring)

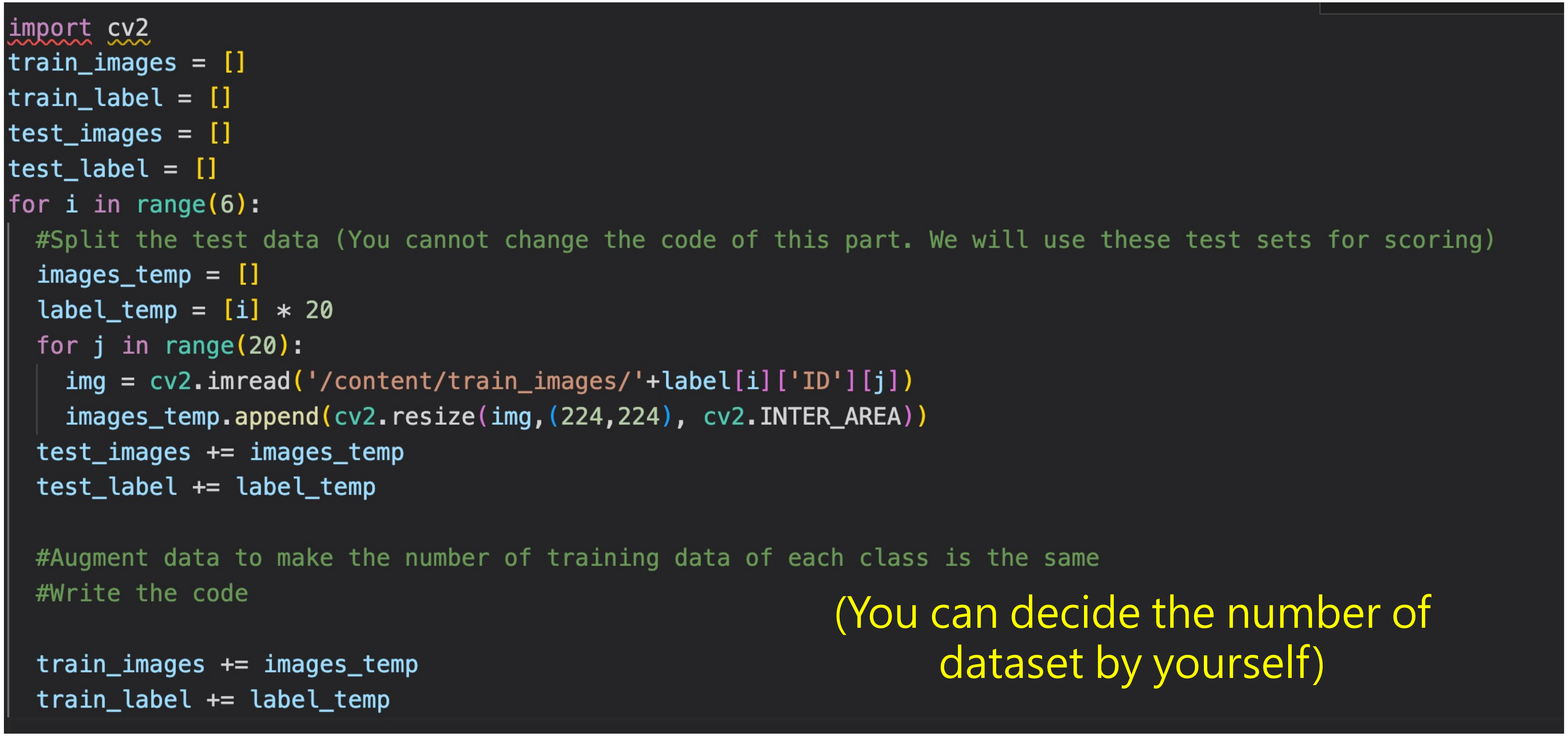


```
import cv2
train_images = []
train_label = []
test_images = []
test_label = []
for i in range(6):
  #Split the test data (You cannot change the code of this part. We will use these test sets for scoring)
  images_temp = []
  label_temp = [i] * 20
  for j in range(20):
    img = cv2.imread('/content/train_images/'+label[i]['ID'][j])
    images_temp.append(cv2.resize(img,(224,224), cv2.INTER_AREA))
  test_images += images_temp
  test_label += label_temp
```

You can try other image preprocessing method

Data Augmentation

- Augment data to make the number of data of each class is the same.

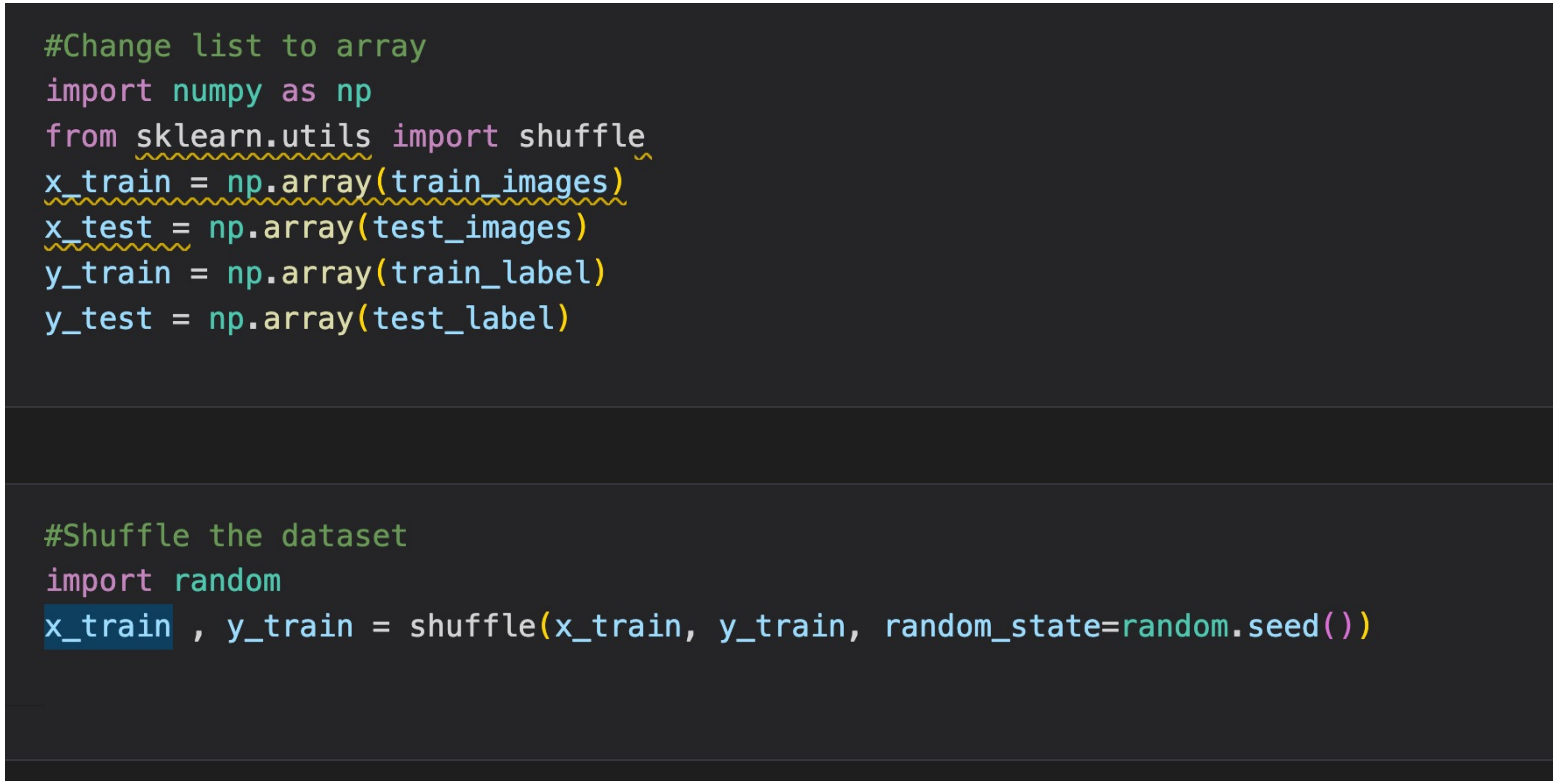


```
import cv2
train_images = []
train_label = []
test_images = []
test_label = []
for i in range(6):
  #Split the test data (You cannot change the code of this part. We will use these test sets for scoring)
  images_temp = []
  label_temp = [i] * 20
  for j in range(20):
    img = cv2.imread('/content/train_images/'+label[i]['ID'][j])
    images_temp.append(cv2.resize(img,(224,224), cv2.INTER_AREA))
  test_images += images_temp
  test_label += label_temp

  #Augment data to make the number of training data of each class is the same
  #Write the code

  train_images += images_temp
  train_label += label_temp
```

(You can decide the number of dataset by yourself)

Change the data and Shuffle

- Change the data to numpy and shuffle the training data.

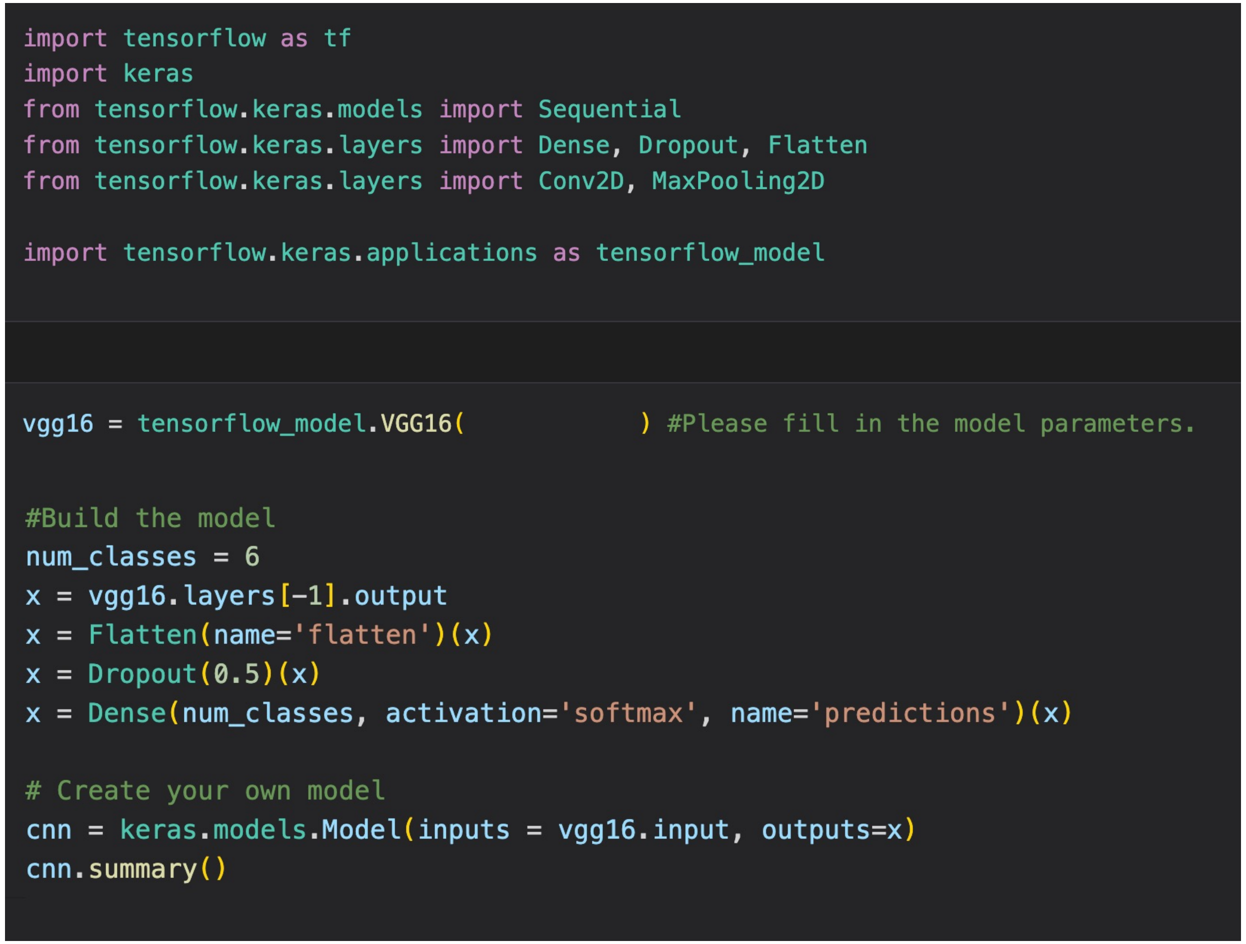


```
#Change list to array
import numpy as np
from sklearn.utils import shuffle
x_train = np.array(train_images)
x_test = np.array(test_images)
y_train = np.array(train_label)
y_test = np.array(test_label)
```

```
#Shuffle the dataset
import random
x_train , y_train = shuffle(x_train, y_train, random_state=random.seed())
```

Build the model

- You can build try you model. Here we use VGG16 as a demo.

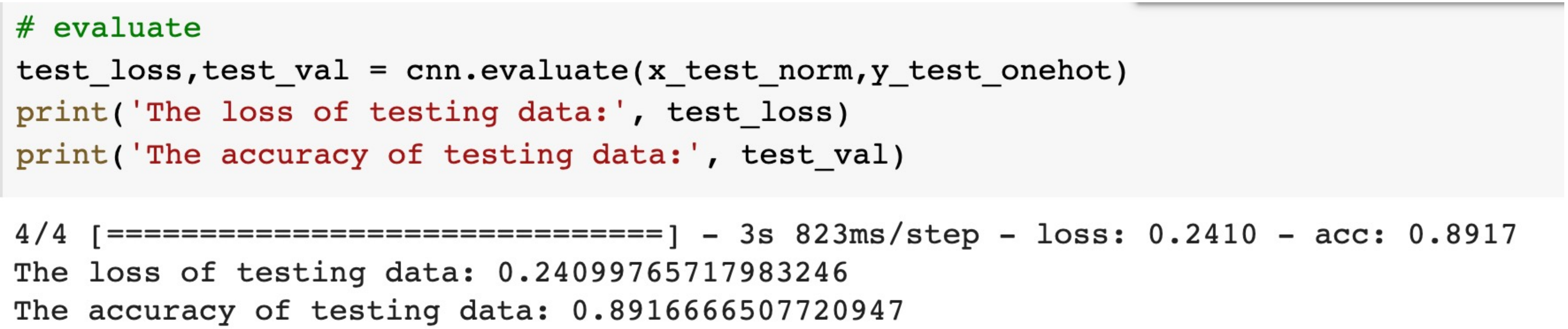


```
import tensorflow as tf
import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D

import tensorflow.keras.applications as tensorflow_model
```

```
vgg16 = tensorflow_model.VGG16(           ) #Please fill in the model parameters.


#Build the model
num_classes = 6
x = vgg16.layers[-1].output
x = Flatten(name='flatten')(x)
x = Dropout(0.5)(x)
x = Dense(num_classes, activation='softmax', name='predictions')(x)

# Create your own model
cnn = keras.models.Model(inputs = vgg16.input, outputs=x)
cnn.summary()
```

Loss and Accuracy

- The testing accuracy should be higher than 89%.



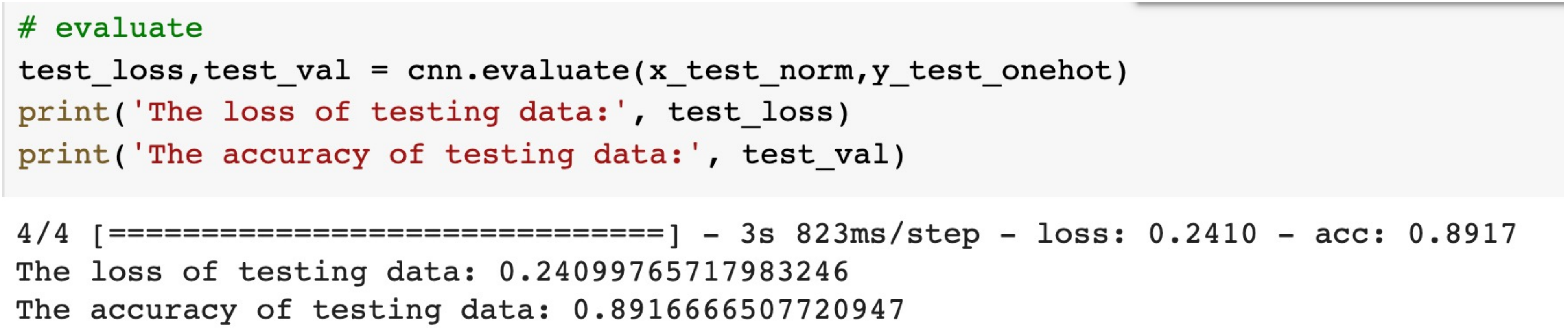
```
# evaluate
test_loss,test_val = cnn.evaluate(x_test_norm,y_test_onehot)
print('The loss of testing data:', test_loss)
print('The accuracy of testing data:', test_val)
```

```
4/4 [==============================] - 3s 823ms/step - loss: 0.2410 - acc: 0.8917
The loss of testing data: 0.24099765717983246
The accuracy of testing data: 0.8916666507720947
```

HW

- Build your own AOI classification model

- Printscreen the accuracy higher than 89%



```
# evaluate
test_loss,test_val = cnn.evaluate(x_test_norm,y_test_onehot)
print('The loss of testing data:', test_loss)
print('The accuracy of testing data:', test_val)
```

```
4/4 [==============================] - 3s 823ms/step - loss: 0.2410 - acc: 0.8917
The loss of testing data: 0.24099765717983246
The accuracy of testing data: 0.891666507720947
```

- Zip the printscreeen and code.py (or .ipynb)

- Deadline: 2022/1/11 23:30