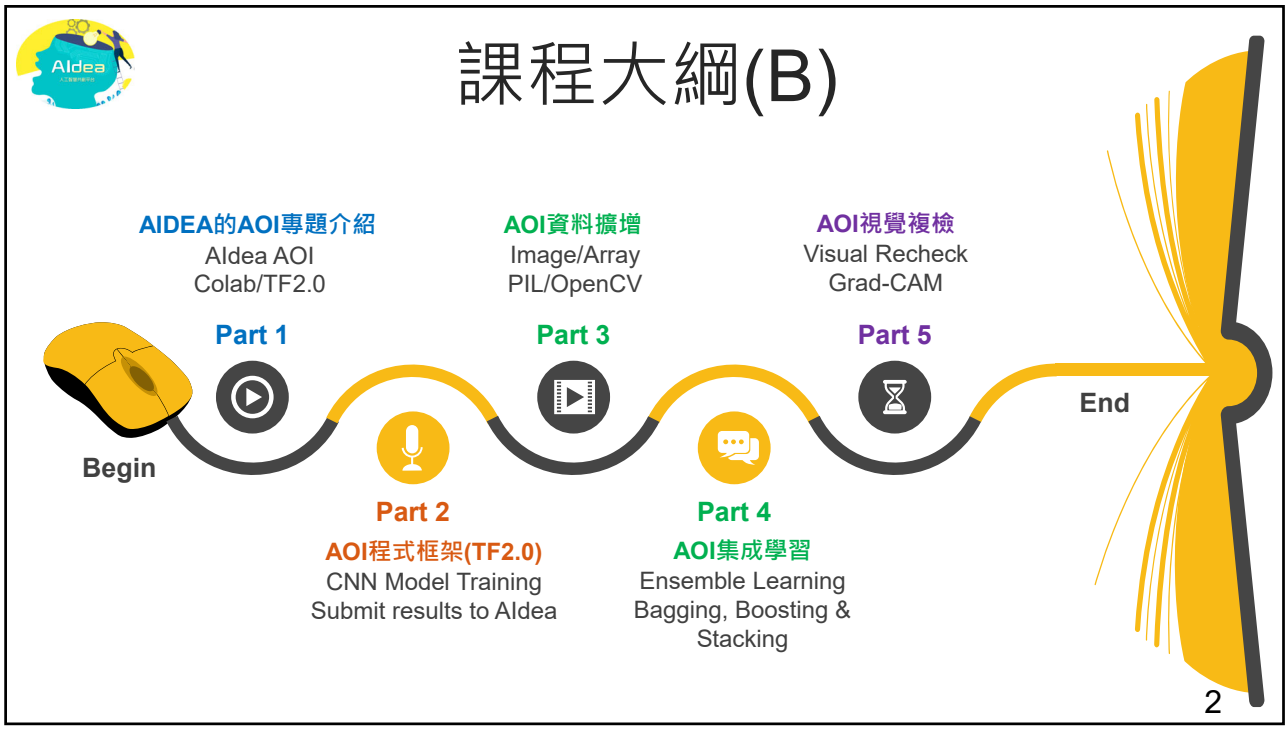

工研院產業學院
「AOI自動光學檢測瑕疵分類實作班」
(B)
亞洲大學朱學亭副教授
2020/07/01


課程大綱(B)
AIDEA的AOI專題介紹
AIdea AOI
Colab/TF2.0
Part 1
AOI資料擴增
Image/Array
PIL/OpenCV
Part 3
AOI視覺複檢
Visual Recheck
Grad-CAM
Part 5
Begin
End
Part 2
AOI程式框架(TF2.0)
CNN Model Training
Submit results to AIdea
Part 4
AOI集成學習
Ensemble Learning
Bagging, Boosting &
Stacking
2

Three issues for AOI

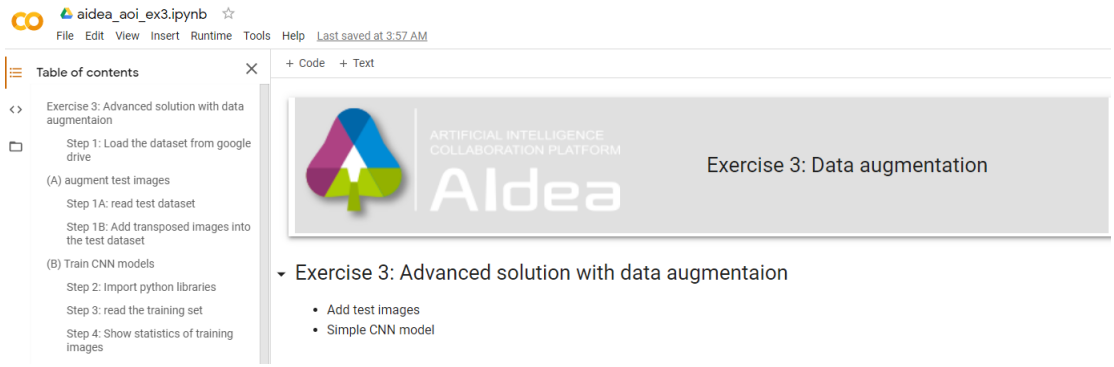
(1) Data Augmentation 資料增強

(2) Ensemble Learning 集成學習

(3) Visual Recheck 視覺複檢

3

3. AOI資料擴增



4

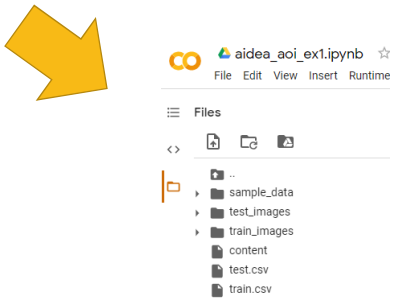
Step 1a: Load the dataset from google drive

AUAOI Ex3.

▾ Step 1: Load the AIdea AOI dataset from google drive

```
[ ] from google_drive_downloader import GoogleDriveDownloader
    GoogleDriveDownloader.download_file_from_google_drive(file_id='      yhVsQZU2iiK19xlJubw0afQ
```

```
Downloading      yhVsQZU2iiK19xlJubw0afQ2EMu5 into ./content... Done.
Unzipping...Done.
```



5

Step 1b: read the training set

AUAOI Ex3.

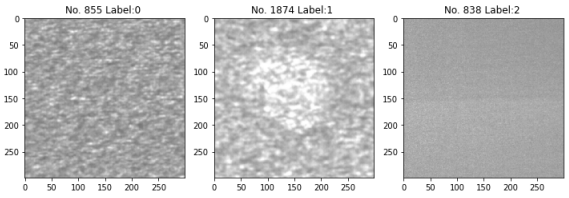
train.csv

```
import pandas as pd
df_train = pd.read_csv('[redacted]')#,dtype=str
train_files = df_train.iloc[:,0].values
train_labels = df_train.iloc[:,1].values
df_train.head()
```

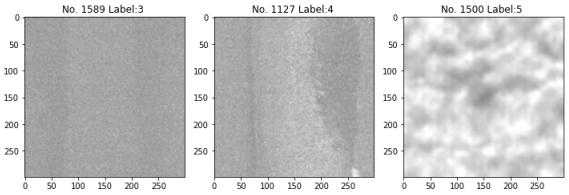
	ID	Label
0	train_00000.png	0
1	train_00001.png	1
2	train_00002.png	1
3	train_00003.png	5
4	train_00004.png	5

6

資料擴增



0 (normal), 1 (void), 2 (horizontal defect)



3 (vertical defect), 4 (edge defect), 5 (particle)

7

Step 1c: Add transposed images into the test dataset

AUAOI Ex3.

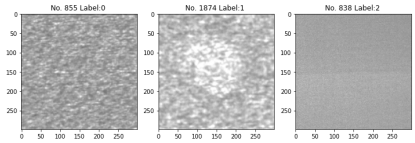
```
from PIL import Image
train_path = "train_images/"
df_train2 = pd.DataFrame(df_train)
for file,label in zip(train_files,train_labels ):
    img = Image.open(train_path+file)
    if label == 2 or label == 3 or label == 4:
        #flip image
        img_n1 = img.transpose(Image.FLIP_LEFT_RIGHT)
        file_n1 = file.replace(".png", "-h.png")
        img_n1.save(train_path+file_n1)
        new_row = {'ID':file_n1, 'Label':label}
        df_train2 = df_train2.append(new_row, ignore_index=True)#append row to the dataframe

        img_n2 = img.transpose(Image.FLIP_TOP_BOTTOM)
        file_n2 = file.replace(".png", "-v.png")
        img_n2.save(train_path+file_n2)
        new_row = {'ID':file_n2, 'Label':label}
        df_train2 = df_train2.append(new_row, ignore_index=True)#append row to the dataframe
    elif label == 1 or label == 5:
        img_n1 = img.transpose(Image.ROTATE_90)
        file_n1 = file.replace(".png", "-r1.png")
        img_n1.save(train_path+file_n1)
        new_row = {'ID':file_n1, 'Label':label}
        df_train2 = df_train2.append(new_row, ignore_index=True)#append row to the dataframe

        img_n2 = img.transpose(Image.ROTATE_180)
        file_n2 = file.replace(".png", "-r2.png")
        img_n2.save(train_path+file_n2)
        new_row = {'ID':file_n2, 'Label':label}
        df_train2 = df_train2.append(new_row, ignore_index=True)#append row to the dataframe

        img_n3 = img.transpose(Image.ROTATE_270)
        file_n3 = file.replace(".png", "-r3.png")
        img_n3.save(train_path+file_n3)
        new_row = {'ID':file_n3, 'Label':label}
        df_train2 = df_train2.append(new_row, ignore_index=True)#append row to the dataframe
print(df_train2.shape)
```

Image.transpose(*method*) [source]

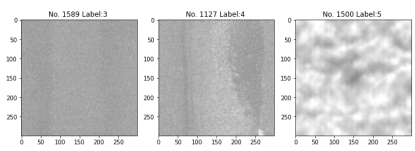
Transpose image (flip or rotate in 90 degree steps)

Parameters: **method** – One of `PIL.Image.FLIP_LEFT_RIGHT`, `PIL.Image.FLIP_TOP_BOTTOM`, `PIL.Image.ROTATE_90`, `PIL.Image.ROTATE_180`, `PIL.Image.ROTATE_270`, `PIL.Image.TRANSPOSE` or `PIL.Image.TRANSVERSE`.

Returns: Returns a flipped or rotated copy of this image.



0 (normal), 1 (void), 2 (horizontal defect)



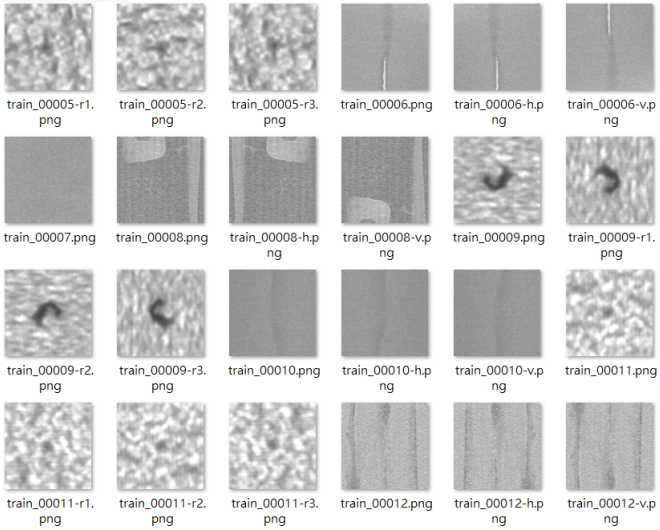
3 (vertical defect), 4 (edge defect), 5 (particle)

8

Step 1d: Add transposed images into the test dataset

AUAOI Ex3.

```
df_train2.to_csv("train2.csv", index=False)
```



9

Step 2: Import python libraries

AUAOI Ex3.

```
[2] import tensorflow as tf
    tf.config.experimental.set_memory_growth(tf.config.list_physical_devices('GPU')[0], True)
    print(tf.__version__)
    print(tf.config.list_physical_devices('GPU'))
```

```
2.2.0
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```
[3] import numpy as np
    import matplotlib.pyplot as plt
    %matplotlib inline
```

```
[4] from tensorflow.keras import Sequential
    from tensorflow.keras.models import Model
    from tensorflow.keras.layers import Dense, Activation, Flatten
    from tensorflow.keras.layers import Input
    from tensorflow.keras.layers import Dropout, Flatten, Activation
    from tensorflow.keras.layers import Conv2D, MaxPooling2D
    from tensorflow.keras.optimizers import Adam,SGD,Adagrad,Adadelta,RMSprop
```

10

Step 3: read the training set

AUAOI Ex3.

train2.csv str

```
import pandas as pd
df_train = pd.read_csv("                ",dtype=    )
print(df_train.shape)
```

```
[6] df_train.head()
```

	ID	Label
0	train_00000.png	0
1	train_00001.png	1
2	train_00002.png	1
3	train_00003.png	5
4	train_00004.png	5

```
[7] train_files = df_train.iloc[:,0].values
    train_labels = df_train.iloc[:,1].values
    print(train_labels[:10])
```

```
['0' '1' '1' '5' '5' '5' '3' '0' '3' '5']
```

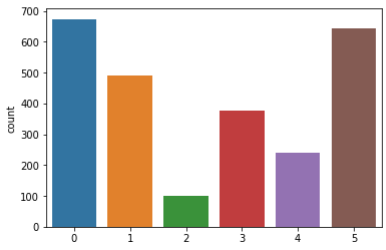
11

Step 4: Show statistics of training images

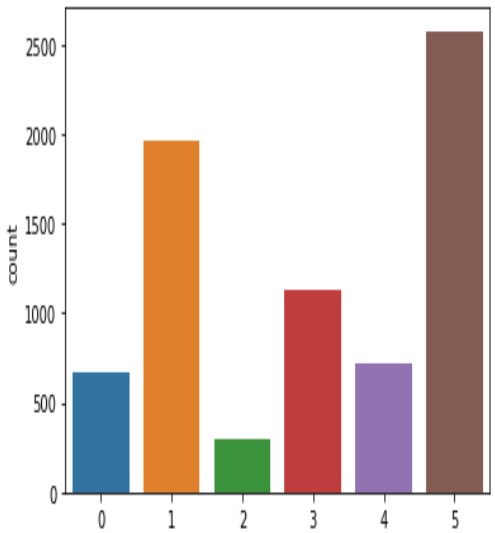
AUAOI Ex3.

```
import seaborn as sns
g = sns.countplot(train_labels)
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_te
  import pandas.util.testing as tm
```



num_classes=6



12

Step 5: Choose one of CNN models

AUAOI Ex3.

```
[11] from tensorflow.keras.applications import
     model = (include_top = True, input_shape=(299,299,3), weights=None, classes=num_classes)

[12] model.summary()
```

- DenseNet121(...): Instantiates the Densenet121 architecture.
- DenseNet169(...): Instantiates the Densenet169 architecture.
- DenseNet201(...): Instantiates the Densenet201 architecture.
- InceptionResNetV2(...): Instantiates the Inception-ResNet v2 architecture.
- InceptionV3(...): Instantiates the Inception v3 architecture.
- MobileNet(...): Instantiates the MobileNet architecture.
- MobileNetV2(...): Instantiates the MobileNetV2 architecture.
- NASNetLarge(...): Instantiates a NASNet model in ImageNet mode.
- NASNetMobile(...): Instantiates a Mobile NASNet model in ImageNet mode.
- ResNet101(...): Instantiates the ResNet101 architecture.
- ResNet101V2(...): Instantiates the ResNet101V2 architecture.
- ResNet152(...): Instantiates the ResNet152 architecture.
- ResNet152V2(...): Instantiates the ResNet152V2 architecture.
- ResNet50(...): Instantiates the ResNet50 architecture.
- ResNet50V2(...): Instantiates the ResNet50V2 architecture.
- VGG16(...): Instantiates the VGG16 model.
- VGG19(...): Instantiates the VGG19 architecture.
- Xception(...): Instantiates the Xception architecture.

13

Step 6: Instancing an ImageDataGenerator

AUAOI Ex3.

preprocess_input

```
[13] from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.applications.xception import preprocess_input
     img_gen = ImageDataGenerator(preprocessing_function=
```

```
tf.keras.preprocessing.image.ImageDataGenerator(
    featurewise_center=False, samplewise_center=False,
    featurewise_std_normalization=False, samplewise_std_normalization=False,
    zca_whitening=False, zca_epsilon=1e-06, rotation_range=0, width_shift_range=0.0,
    height_shift_range=0.0, brightness_range=None, shear_range=0.0, zoom_range=0.0,
    channel_shift_range=0.0, fill_mode='nearest', cval=0.0, horizontal_flip=False,
    vertical_flip=False, rescale=None, preprocessing_function=None,
    data_format=None, validation_split=0.0, dtype=None
)
```

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

14

Step 7: Set up a train_generator with flow_from_dataframe

AUAOI Ex3.

```
[14] train_generator = img_gen.flow_from_dataframe(dataframe=[redacted],
                directory="[redacted]",
                x_col="[redacted]",
                y_col="[redacted]",
                subset=None,
                batch_size=8,
                shuffle=False,
                class_mode="categorical",
                color_mode="rgb",
                target_size=(299,299))
```

```
Found 2528 validated image filenames belonging to 6 classes.
```

```
[15] train_generator.class_indices
```

```
{'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5}
```

df_train2

train_images

ID

Label

```
flow_from_dataframe(
   dataframe, directory=None, x_col='filename', y_col='class', weight_col=None,
   target_size=(256, 256), color_mode='rgb', classes=None,
   class_mode='categorical', batch_size=32, shuffle=True, seed=None,
   save_to_dir=None, save_prefix='', save_format='png', subset=None,
   interpolation='nearest', validate_filenames=True, **kwargs
)
```

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

15

Step 8: step_size_train

AUAOI Ex3.

```
if train_generator.n % train_generator.batch_size ==0:
  step_size_train=train_generator.n//train_generator.batch_size
else:
  step_size_train=train_generator.n//train_generator.batch_size + 1
print(step_size_train)
```

16

Step 9:

ModelCheckpoint

Callback to save the Keras model or model weights at some frequency.

AUAOI Ex3.

```
[17] # Include the epoch in the file name (uses `str.format`)
     import os
     checkpoint_path = "training_cp/cp-{epoch:03d}.ckpt"
     checkpoint_dir = os.path.dirname(checkpoint_path)
     # Create a callback that saves the model's weights
     cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=[          ], save_weights_only=True)
```

checkpoint_path

```
tf.keras.callbacks.ModelCheckpoint(
    filepath, monitor='val_loss', verbose=0, save_best_only=False,
    save_weights_only=False, mode='auto', save_freq='epoch', **kwargs
)
```

https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint

17

Step 10:

EarlyStopping

Stop training when a monitored metric has stopped improving.

AUAOI Ex3.

```
[18] # Create a callback that stop the model
     es_callback = tf.keras.callbacks.EarlyStopping(monitor='[  ]', patience=5)
```

loss

```
tf.keras.callbacks.EarlyStopping(
    monitor='val_loss', min_delta=0, patience=0, verbose=0, mode='auto',
    baseline=None, restore_best_weights=False
)
```

18

Step 11: Compile model

AUAOI Ex3.

```
[19] #compile model using accuracy to measure model performance
     from tensorflow.keras import optimizers
     model.compile(loss='categorical_crossentropy',
                   optimizer=optimizers.[    ](lr=3e-3),
                   metrics=['accuracy'])
```

Adam/SGD

```
compile(
    optimizer='rmsprop', loss=None, metrics=None, loss_weights=None,
    sample_weight_mode=None, weighted_metrics=None, **kwargs
)
```

tf.keras.optimizers

Optimizer	特點
SGD	• 有機會跳出目前局部收斂進而進到另一個局部收斂而得到最小值，而得到全局最小值 • 需自行設定learning rate，較難選擇到合適的learning rate • 會造成loss function有嚴重的震蕩 • 需要較長時間收斂至最小值
Momentum	• 能夠在相關方向加速SGD，抑制SGD的嚴重震蕩，進而加快收斂 • 需自行設定learning rate與β，有可能會使參數的移動方向偏移梯度下分的方向，進而導至沒有那麼快速的收斂
AdaGrad	• 能夠自動調整learning rate，進而調整收斂 • 適合處理稀疏梯度 • 依然需要人工設置一個全局的learning rate • 後期，分母梯度平方的累加會越來越大，會使梯度趨近於0，使得訓練結束
Adam	• 結合了AdaGrad與Momentum的優點 • 適用於大數據集和高維空間的資料 • 目前最常使用的一個Optimizer

19

Step 12: Train model

AUAOI Ex3.

cp_callback, es_callback

train_generator step_size_train

```
hist = model.fit_generator(generator=[          ], steps_per_epoch=[          ], callbacks=[[          ], [          ]], epochs=100)
```

```
WARNING:tensorflow:From <ipython-input-20-5be68c6e0f2d>:1: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecate
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/100
111/316 [=========>.....................] - ETA: 1:09 - loss: 1.7552 - accuracy: 0.4155
```

```
fit_generator(
    generator, steps_per_epoch=None, epochs=1, verbose=1, callbacks=None,
    validation_data=None, validation_steps=None, validation_freq=1,
    class_weight=None, max_queue_size=10, workers=1, use_multiprocessing=False,
    shuffle=True, initial_epoch=0
)
```

20

Step 13: Evaluate saved checkpoints

AUAOI Ex3.

```
##checkpoint 1
model.load_weights("training_cp/cp-001.ckpt")
train_generator.reset()
model.evaluate_generator(generator=train_generator, steps=step_size_train, verbose=1)
```

```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7fcc08582cc0>
```

```
##checkpoint 2
model.load_weights("training_cp/cp-001.ckpt")
train_generator.reset()
model.evaluate_generator(generator=train_generator, steps=step_size_train, verbose=1)
```

```
##checkpoint 3
model.load_weights("training_cp/cp-001.ckpt")
train_generator.reset()
model.evaluate_generator(generator=train_generator, steps=step_size_train, verbose=1)
```

21

Step 14: Save the trained model

AUAOI Ex3.

```
model.load_weights("training_cp/cp-001.ckpt")
model.save("AOI-InceptionV3-0626.h5")
```

```
save(
    filepath, overwrite=True, include_optimizer=True, save_format=None,
    signatures=None, options=None
)
```

https://www.tensorflow.org/api_docs/python/tf/keras/Model#save

22

Step 15: Check training results

AUAOI Ex3.

train_generator

```
#y_predictions = model.predict(X_train, batch_size=20)
train_generator.reset()
y_predictions = model.predict_generator(generator=[redacted], steps=step_size_train, verbose=1)
```

```
WARNING:tensorflow:From <ipython-input-18-9c359a3ebada>:3: Model.predict_generator (from tensorflow.pytho
Instructions for updating:
Please use Model.predict, which supports generators.
316/316 [==============================] - 9s 29ms/step
```

```
print(y_predictions[:2])
type(y_predictions)
```

```
predicts = np.argmax(y_predictions,axis=1)
print(predicts[0:10])
```

```
[0 1 1 5 5 5 3 0 3 5]
```

```
labels = train_labels.astype(int)
print(labels[:10])
```

```
[0 1 1 5 5 5 3 0 3 5]
```

23

Step 16: Analyze training results

AUAOI Ex3.

labels predicts

```
from sklearn.metrics import confusion_matrix
confusion=confusion_matrix([redacted], [redacted])
print(confusion)
```

```
[[674   0   0   0   0   0]
 [  4 484   0   2   1   1]
 [  0   0 100   0   0   0]
 [  0   0   0 376   1   1]
 [  0   0   0   2 238   0]
 [  0   4   0   0   0 640]]
```

```
overkill= []
underkill = []
for i in range(train_num):
  if labels[i] == 0 and predicts[i] !=0:
    overkill.append(i)
  if labels[i] != 0 and predicts[i] ==0:
    underkill.append(i)
print('# of overkill= {}; # of underkill= {} '.format(len(overkill), len(underkill)))
```

24

Step 17: Load the test set


```
df_test = pd.read_csv("[redacted]",dtype=str)
print(df_test.shape)
```

test.csv

```
df_test.head()
```

```
test_files  = df_test.iloc[:,0].values
test_labels = df_test.iloc[:,1].values
print(test_labels[:10])
```

25

Step 18: Set up a test_generator with flow_from_dataframe


```
test_generator = img_gen.flow_from_dataframe(dataframe=[redacted]
            directory="[redacted]",
            x_col="ID",
            y_col="Label",
            batch_size=32,
            shuffle=False,
            class_mode=[redacted],
            target_size=(299,299))
```

df_test

test_images

None

26

Step 19: step_size_test

AUAOI Ex3.

```
if test_generator.n % test_generator.batch_size ==0:
  step_size_test=test_generator.n//test_generator.batch_size
else:
  step_size_test=test_generator.n//test_generator.batch_size + 1
print(step_size_test)
```

27

Step 20: Check test results

AUAOI Ex3.

```
#y_predictions = model.predict(X_train, batch_size=20)
test_generator.reset()
y_predictions = model.predict_generator(generator=test_generator, steps=step_size_test,verbose=1)
```

```
import numpy as np
predicts=np.argmax(y_predictions,axis=1)
predicts[:10]
```

28

Step 21:


```
df_out = pd.DataFrame(df_test)
df_out.shape
```

```
df_out['Label'] = predicts
df_out.to_csv("0626-xception.csv", index=False)
```

29

4. Ensemble Learning 集成學習

https://en.wikipedia.org/wiki/Ensemble_learning

30

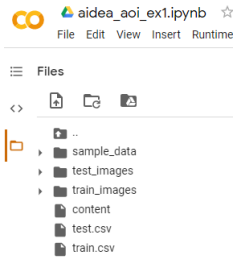
Step 1: Load the dataset from google drive

AUAOI Ex4.

▾ Step 1: Load the AIdea AOI dataset from google drive

```
[ ] from google_drive_downloader import GoogleDriveDownloader
    GoogleDriveDownloader.download_file_from_google_drive(file_id='      yhVsQZU2iiK19xlJubw0afQ
```

```
Downloading      yhVsQZU2iiK19xlJubw0afQ2EMu5 into ./content... Done.
Unzipping...Done.
```



31

Step 2: Import python libraries

AUAOI Ex3.

```
[2] import tensorflow as tf
    tf.config.experimental.set_memory_growth(tf.config.list_physical_devices('GPU')[0], True)
    print(tf.__version__)
    print(tf.config.list_physical_devices('GPU'))
```

```
2.2.0
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```
[3] import numpy as np
    import matplotlib.pyplot as plt
    %matplotlib inline
```

```
[4] from tensorflow.keras import Sequential
    from tensorflow.keras.models import Model
    from tensorflow.keras.layers import Dense, Activation, Flatten
    from tensorflow.keras.layers import Input
    from tensorflow.keras.layers import Dropout, Flatten, Activation
    from tensorflow.keras.layers import Conv2D, MaxPooling2D
    from tensorflow.keras.optimizers import Adam,SGD,Adagrad,Adadelta,RMSprop
```

32

Step 3: Load pretrained models

AUAOI Ex4.

- m1: auaoi-InceptionResNetV2
- m2: auaoi-InceptionV3
- m3: auaoi-Xception

```
#load AIdea AOI trained CNN models
from google_drive_downloader import GoogleDriveDownloader
GoogleDriveDownloader.download_file_from_google_drive(file_id='[redacted]5v1fkLU3EC-xZCIq6mmvYkKyRUq9',dest_path='./model', unzip=True)
```

```
Downloading      5v1fkLU3EC-xZCIq6mmvYkKyRUq9 into ./model... Done.
Unzipping...Done.
```

```
model_files = ['m1.h5', 'm2.h5', 'm3.h5']
renames = ['a', 'b', 'c']
models = []
for file in model_files:
  model = tf.keras.models.load_model(file)
  models.append(model)
```

33

Step 4: read the training set

AUAOI Ex3.

train.csv str

```
[ ] import pandas as pd
    df_train = pd.read_csv('[redacted]',dtype=[redacted])
    print(df_train.shape)
```

```
(2528, 2)
```

```
[ ] df_train.head()
```

	ID	Label
0	train_00000.png	0
1	train_00001.png	1
2	train_00002.png	1
3	train_00003.png	5
4	train_00004.png	5

```
[ ] train_files = df_train.iloc[:,0].values
    train_labels = df_train.iloc[:,1].values
    print(train_labels[:20])
```

```
['0' '1' '1' '5' '5' '5' '3' '0' '3' '5' '3' '5' '3' '3' '1' '1' '1' '1'
 '5' '1']
```

```
[ ] import seaborn as sns
    g = sns.countplot(train_labels)
```

```
[ ] num_classes=6
```

34

Step 5: Set up an ImageDataGenerator and a Train_generator

AUAOI Ex4.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.xception import preprocess_input
#from tensorflow.keras.applications.inception_v3 import preprocess_input
img_gen = ImageDataGenerator(preprocessing_function=[redacted])
```

```
train_generator = img_gen.flow_from_dataframe(dataframe=[redacted],
            directory="[redacted]",
            x_col="[redacted]",
            y_col="[redacted]",
            subset=None,
            batch_size=8,
            shuffle=False,
            class_mode="categorical",
            color_mode="rgb",
            target_size=(299,299))
print(train_generator.n)
```

```
Found 2528 validated image filenames belonging to 6 classes.
```

```
train_generator.class_indices
```

```
{'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5}
```

```
if train_generator.n % train_generator.batch_size ==0:
  step_size_train=train_generator.n//train_generator.batch_size
else:
  step_size_train=train_generator.n//train_generator.batch_size + 1
print(step_size_train)
```

35

Step 6: Evaluation of pretrained models with the training set

AUAOI Ex4.

```
##Evaluation of pretrained models
for model in models:
    train_generator.reset()
    model.evaluate_generator(generator=[redacted], steps=[redacted], verbose=1)
```

```
WARNING:tensorflow:From <ipython-input-22-92dba17c3dc4>:4: Model.evaluate_generator (from tensorflow.python.keras.en
Instructions for updating:
Please use Model.evaluate, which supports generators.
316/316 [==============================] - 73s 232ms/step - loss: 0.0239 - accuracy: 0.9941
316/316 [==============================] - 31s 98ms/step - loss: 0.0090 - accuracy: 0.9976
316/316 [==============================] - 47s 148ms/step - loss: 0.0069 - accuracy: 0.9980
```

36

Step 7: Prediction of pretrained models with the training set

AUAOI Ex4.

```
#y_predictions = model.predict(X_train, batch_size=20)
y_predictions_All = []
for model in models:
    train_generator.reset()
    y_predictions = model.predict_generator(generator=[redacted], steps=[redacted], verbose=1)
    y_predictions_All.append(y_predictions)
```

```
WARNING:tensorflow:From <ipython-input-23-1f686040bea4>:5: Model.predict_generator (from tensorflow.python
Instructions for updating:
Please use Model.predict, which supports generators.
316/316 [==============================] - 71s 224ms/step
316/316 [==============================] - 30s 95ms/step
316/316 [==============================] - 46s 145ms/step
```

37

Step 8: Confusion matrices of pretrained models with the training set

AUAOI Ex4.

```
labels = train_labels.astype(int)
print(labels[:10])
```

```
[0 1 1 5 5 5 3 0 3 5]
```

```
from sklearn.metrics import confusion_matrix
predicts_all = []
for y_predictions in y_predictions_All:
    predicts = np.[redacted](y_predictions,axis=1)
    predicts_all.append(predicts)
    confusion=confusion_matrix([redacted], [redacted])
    print(confusion)
```

38

Step 9: Show differences of predictions between the models

AUAOI Ex4.

```
for i in range(len(labels)):
    label=labels[i]
    pred0=predicts_all[0][i]
    pred1=predicts_all[1][i]
    pred2=predicts_all[2][i]
    if label!=pred0 or label!=pred1 or label!=pred2:
        print(f'{label}->({pred0}, {pred1}, {pred2})')
```

39

Step 10: Ensemble predictions by indiviual models

AUAOI Ex4.

```
y_predictions_ensemble = y_predictions_All[0]+y_predictions_All[1]+y_predictions_All[2]
print(y_predictions_ensemble[:2])
type(y_predictions)
```

```
[[2.9977081e+00 2.6643804e-05 2.0590229e-03 2.0168223e-04 4.4715466e-06
  3.2687109e-09]
 [4.1103329e-11 2.9999866e+00 1.3372697e-05 3.4153191e-10 7.0617383e-09
  1.2877229e-08]]
numpy.ndarray
```

```
predicts_ensemble = np.a[redacted](y_predictions_ensemble,axis=1)
print(predicts_ensemble[0:10])
```

40

Step 11: Show differences of predictions by the ensembled models

AUAOI Ex4.

```
for i in range(len(labels)):
    label=labels[i]
    pred0=predicts_all[0][i]
    pred1=predicts_all[1][i]
    pred2=predicts_all[2][i]
    predx=predicts_ensemble[i]
    if label!=pred0 or label!=pred1 or label!=pred2:
        print(f'{label}->({pred0}, {pred1}, {pred2})=>{predx}')
```

```
confusion=confusion_matrix(labels, [redacted])
print(confusion)
```

predicts_ensemble

```
[[674   0   0   0   0   0]
 [  1 490   0   1   0   0]
 [  0   0 100   0   0   0]
 [  0   0   0 378   0   0]
 [  0   0   0   1 239   0]
 [  0   0   0   0   0 644]]
```

```
overkill= []
underkill = []
for i in range(train_num):
  if labels[i] == 0 and predicts_ensemble[i] !=0:
    overkill.append(i)
  if labels[i] != 0 and predicts_ensemble[i] ==0:
    underkill.append(i)
print('# of overkill= {}; # of underkill= {} '.format(len(overkill), len(underkill)))
```

41

Step 12: Preparing test images

AUAOI Ex4.

```
df_test = pd.read_csv('[redacted]',dtype=str)
print(df_test.shape)
```

```
(10142, 2)
```

```
df_test.head()
```

```
test_files  = df_test.iloc[:,0].values
test_labels = df_test.iloc[:,1].values
print(test_labels[:10])
```

```
[nan nan nan nan nan nan nan nan nan nan]
```

```
img_gen2 = ImageDataGenerator(horizontal_flip=False, vertical_flip=False, preprocessing_function=[redacted])
test_generator = img_gen2.flow_from_dataframe(dataframe=df_test,
            directory="[redacted]",
            x_col="ID",
            y_col="Label",
            batch_size=32,
            shuffle=False,
            class_mode=[redacted],
            target_size=(299,299))
```

```
Found 10142 validated image filenames.
```

```
if test_generator.n % test_generator.batch_size ==0:
  step_size_test=test_generator.n//test_generator.batch_size
else:
  step_size_test=test_generator.n//test_generator.batch_size + 1
print(step_size_test)
```

42

Step 13: Prediction of pretrained models with the testing set

AUAOI Ex4.

```
#y_predictions = model.predict(X_train, batch_size=20)
y_predictions_All = []
for model in models:
    test_generator.reset()
    y_predictions = model.predict_generator(generator=[redacted], steps=[redacted],verbose=1)
    y_predictions_All.append(y_predictions)
```

```
317/317 [==============================] - 221s 696ms/step
317/317 [==============================] - 84s 265ms/step
317/317 [==============================] - 168s 529ms/step
```

43

Step 14: Ensemble Predictions

AUAOI Ex4.

```
y_predictions_ensemble = y_predictions_All[0]+y_predictions_All[1]+y_predictions_All[2]
y_predictions_ensemble.shape
```

```
(10142, 6)
```

```
predicts_ensemble=np.[redacted](y_predictions_ensemble,axis=1)
```

44

Step 15: Output the predictions


```
df_out = pd.DataFrame(df_test)
df_out.shape
```

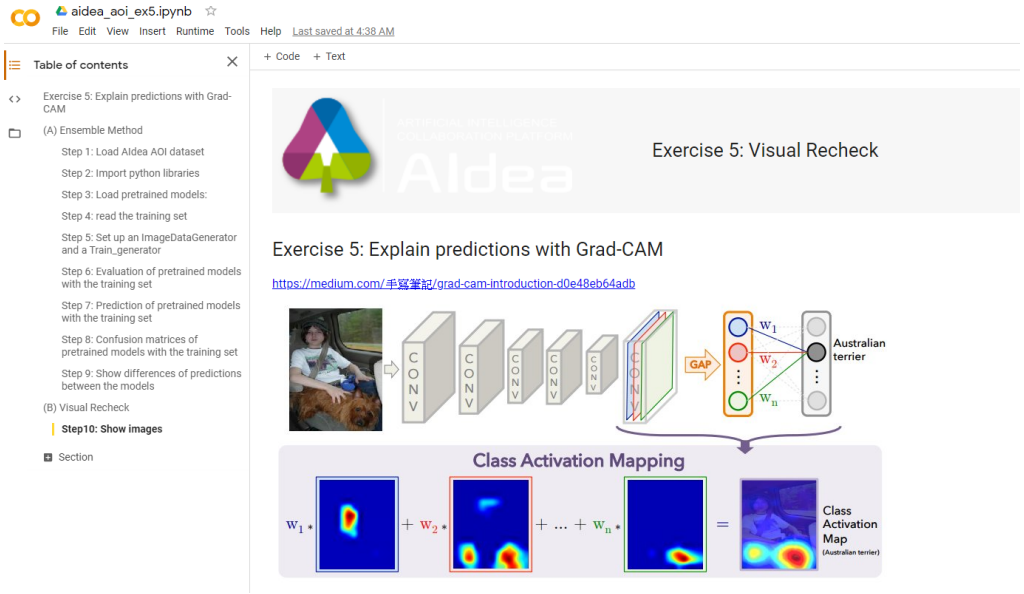
```
(10142, 2)
```

```
df_out['Label'] = [redacted]
df_out.to_csv("0627-ensemble.csv", index=False)
```

predicts_ensemble

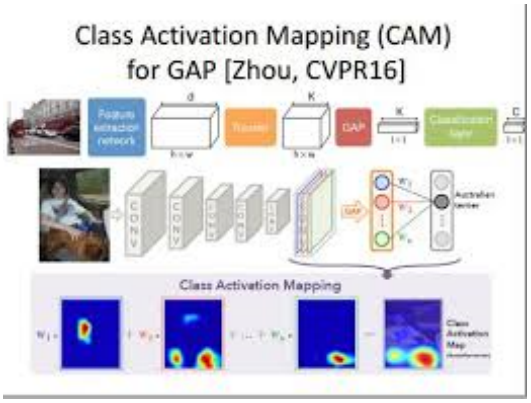
45

5. Visual Recheck 視覺複檢



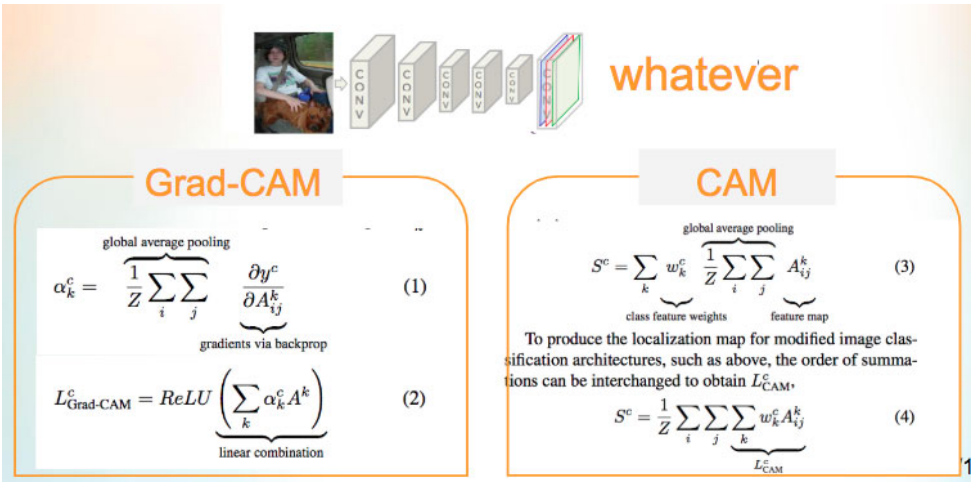
46

Class Activation Mapping



47

Grad-CAM

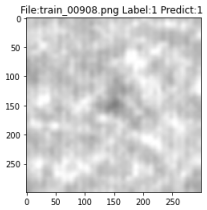


48

Step 10: Show images

AUAOI Ex5.

```
sel = 908
file = train_files[sel]
label= train_labels[sel]
pred = predicts[sel]
img = image.load_img("train_images/"+file, color_mode="rgb", target_size = (299, 299))
fig,ax=plt.subplots(1,1)
ax.imshow(img, cmap='gray')
ax.set_title('File:{} Label:{} Predict:{} '.format(file, label, pred))
plt.show()
```



49

Step 11: Build grad_models

AUAOI Ex5.

```
grad_models=[]
```

```
#Model: "inception_resnet_v2"
models[0].summary()
```

```
grad_model0 = Model([models[0].inputs], [models[0].get_layer('conv_7b_ac').output, models[0].output])
grad_models.append(grad_model0)
```

```
#Model: "inception_v3"
models[1].summary()
```

```
grad_model1 = Model([models[1].inputs], [models[1].get_layer('mixed10').output, models[1].output])
grad_models.append(grad_model1)
```

```
#Model: "xception"
models[2].summary()
```

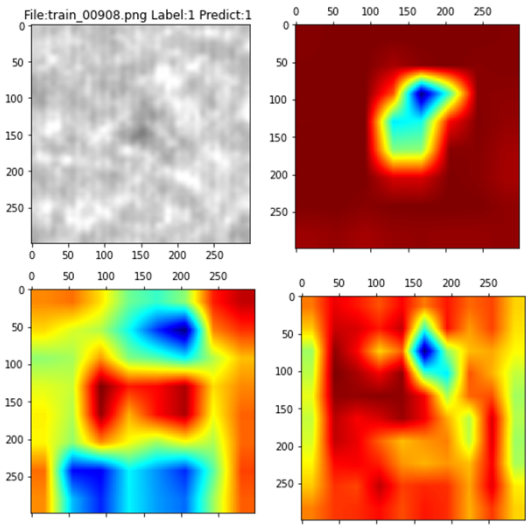
```
grad_model2 = Model([models[2].inputs], [models[2].get_layer('block14_sepconv2_act').output, models[2].output])
grad_models.append(grad_model2)
```

50

Step 12: Show CAM images

AUAOI Ex5.

```
grad_model = grad_model0
sel = 908
pred= 5
file = train_files[sel]
label= train_labels[sel]
img = image.load_img("train_images/"+file, color_mode="rgb", target_size = (299, 299))
x = image.img_to_array(img)
img_array = preprocess_input(x, mode = 'tf' )

with tf.GradientTape() as tape:
    conv_outputs, predictions = grad_model(np.array([img_array]))
    loss = predictions[:, pred]

output = conv_outputs[0]
grads = tape.gradient(loss, conv_outputs)[0]

gate_f = tf.cast(output > 0, 'float32')
gate_r = tf.cast(grads > 0, 'float32')
guided_grads = tf.cast(output > 0, 'float32') * tf.cast(grads > 0, 'float32') * grads

weights = tf.reduce_mean(guided_grads, axis=(0, 1))

cam = np.ones(output.shape[0: 2], dtype = np.float32)

for i, w in enumerate(weights):
    cam += w * output[:, :, i]

cam = cv2.resize(cam.numpy(), (299, 299))
cam = np.maximum(cam, 0)
heatmap = (cam - cam.min()) / (cam.max() - cam.min())

cam = cv2.applyColorMap(np.uint8(255*heatmap), cv2.COLORMAP_JET)
# Display heatmap
plt.matshow(cam)
plt.show()
```



51

~~End~~

Thanks!

Q&A

行走江湖難免遇到八鴿...