AOI Course
hands-on training (2)

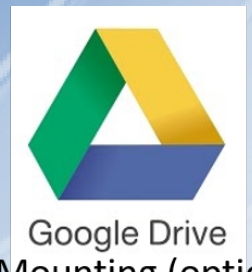
Hsueh-Ting Chu/朱學亭
htchu@asia.edu.tw

AOI Hands-on Course

- Tutorial 1: training a full CNN model for AOI
- Tutorial 2: Transfer a CNN model for AOI

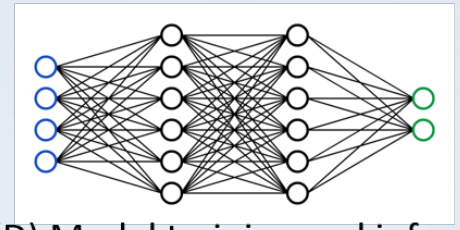
Tutorial 2: Transfer a CNN model for AOI

(A) Setup TF 2.0



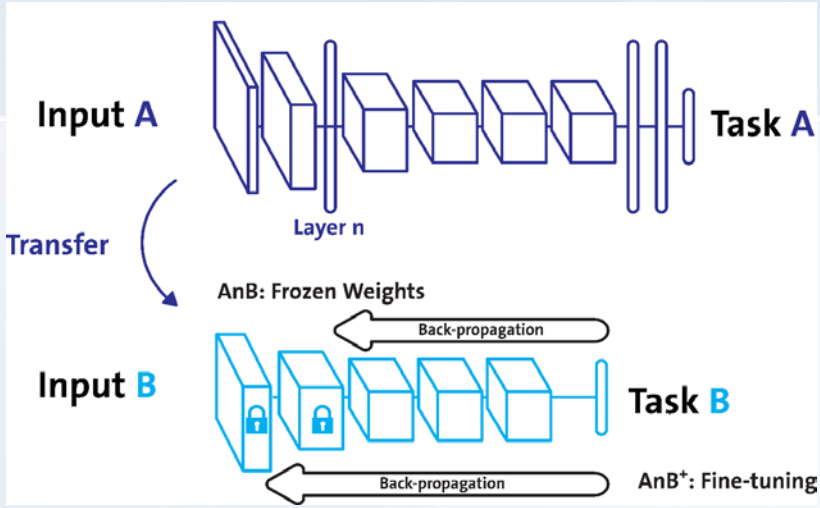
(B) Mounting (optional)

(C) Input training data



(D) Model training and inference

(E) Output test result



Step 1: Choose tensorflow_version

(A) Use TF 2.0 (Optional)

Step 1: Choose tensorflow_version

```
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass

import tensorflow as tf
print(tf.__version__)
```

```
TensorFlow 2.x selected.
2.1.0-rc1
```

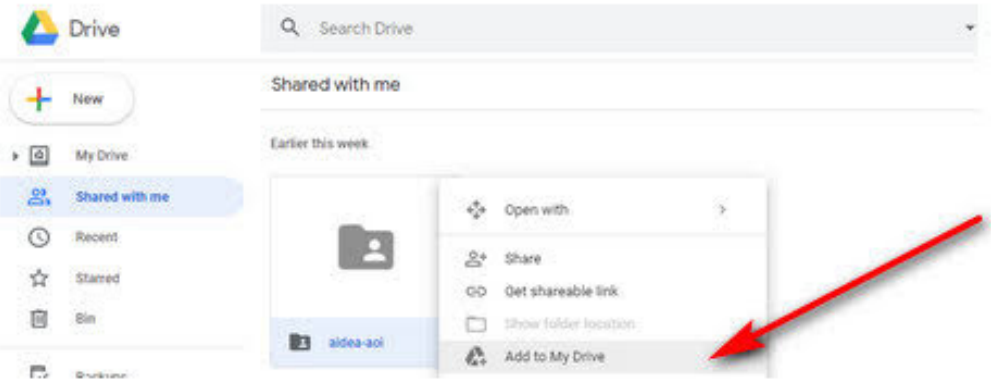
%tensorflow_version 2.x

Step 2: Add the data folder to your Google drive

(B) Setup Colab and mount the AOI data folder

Step 2: Add the data folder to your Google drive

Click https://drive.google.com/open?id=15tGIHAPAatgdB8iZh_m80jCBPa-CrI_P



Add to My Drive

https://drive.google.com/open?id=15tGIHAPAatgdB8iZh_m80jCBPa-CrI_P

Step 3: Mount the AOI folder

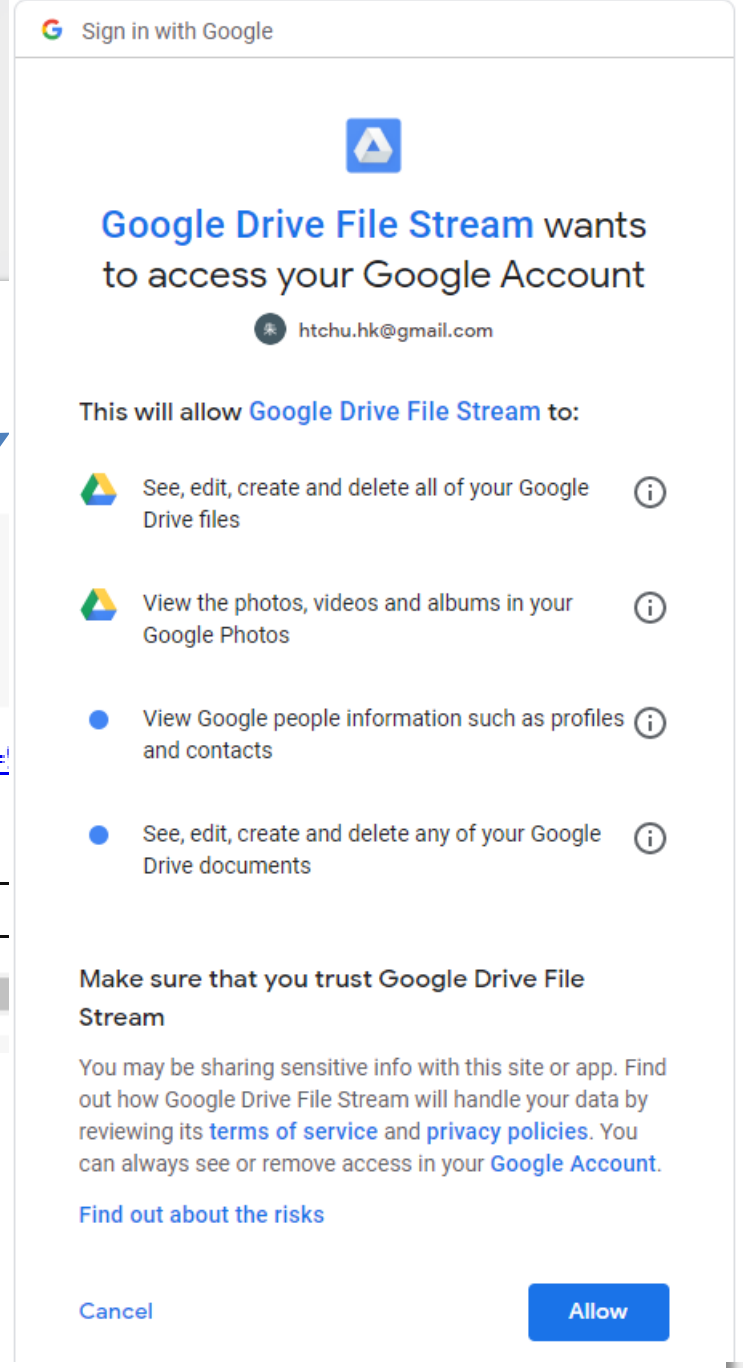
Step 3: Mount the AOI folder

If error, check https://myaccount.google.com/u/2/permissions

```
[ ]  #Step 3a: Mount your Google Drive
     from google.colab import drive
     drive.mount("/content/drive", force_remount=True)
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=

Enter your authorization code:
```



Step 4: Check the AOI data pathersion

Step 4: Check the AOI data path

```
data_path = "/content/drive/My Drive/aidea-aoi2/data/"
!ls "/content/drive/My Drive/aidea-aoi2/data/"
```

```
#alternative data path for local computer
data_path = "../data/"
#!ls "../data/"
!dir/w "../data/"
```

data_path = "/content/drive/My Drive/aidea-aoi2/data/"

Step 5: read lalels of the training set

Step 5: read lalels of the training set

```
import pandas as pd
df_train = pd.read_csv(data_path+ "train.csv")
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.csv"

Step 6: Build the lists of training images and labels from the dataframe

Step 6: Build the lists of training images and labels from the dataframe

```
#limit the amount of training images for the class process
#train_num = df_train.shape[0]
train_num = 480
if train_num >= df_train.shape[0]:
  train_num = df_train.shape[0]
train_files = df_train.iloc[:train_num,0].values
train_labels = df_train.iloc[:train_num,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]
```

train_labels = df_train.iloc[:train_num,1].values

Step 7: read images of the training set

Step 7: read images of the training set

```
train_path = data_path+ "train_images/"
train_images = []
from tensorflow.keras.preprocessing import image
for file in train_files:
    img = image.load_img(train_path+file, color_mode="rgb", target_size = (224, 224))
    train_images.append(img)
    if len(train_images)%100 == 0:
      print('.', end='')
print(len(train_images))
```

color_mode="rgb", target_size = (224, 224)

Step 8: show AOI images of the classes

Step 8: show AOI images of the classes:

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

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
import random
curclass = 0
fig,ax=plt.subplots(2, 3)
fig.set_size_inches(10,10)
for i in range(2):
    for j in range (3):
        sel=random.randint(0,train_num)
        while train_labels[sel]!=curclass:
          sel +=1
          if sel == train_num -1:
            sel = 0
        curclass += 1
        curclass %= 6
        #sel=random.randint(0,train_num)
        ax[i,j].imshow(train_images[sel], cmap='gray')
        ax[i,j].set_title('No. {} Label:{} '.format(sel, train_labels[sel]))
plt.tight_layout()
```

train_num

Step 9: Show statistics of training images in the 6 classes

Step 9: Show statistics of training images in the 6 classes

```
import numpy as np
labels, counts = np.unique(train_labels, return_counts=True)
print(labels, counts)
```

```
[0 1 2 3 4 5] [119 108  16  77  49 111]
```

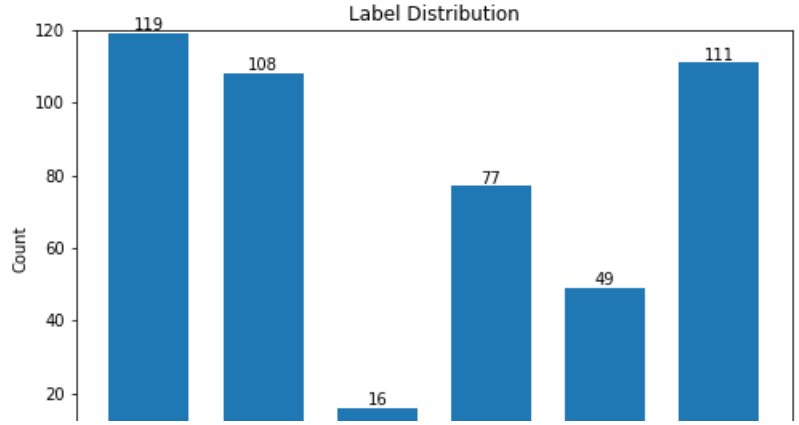
np.unique(train_labels, return_counts=True)

Step 10: Plot the counts

Step 10: Plot the counts

```
fig = plt.figure(figsize=(8, 5))
plt.bar(labels, counts, width=0.7, align='center')
plt.title("Label Distribution")
plt.xlabel('Label')
plt.ylabel('Count')
plt.xticks(labels)
plt.ylim(0, 120)

for a, b in zip(labels, counts):
    plt.text(a, b, '%d' % b, ha='center', va='bottom', fontsize=10)
plt.show()
```



labels, counts

Step 11: Check the shape of single image

Step 11: Check the shape of single image

```
from tensorflow.keras.preprocessing.image import img_to_array
# convert to numpy array
img_array0 = img_to_array(train_images[0])
print(img_array0.shape)
del img_array0
```

```
(512, 512, 1)
```

Step 12: Convert each training image into a numpy array and collect

Step 12: Convert each training image into a numpy array and collect

	Input size	Data format	mode
Xception	299x299	channels_last	tf
VGG16	224x224	channels_first / channels_last	caffe
VGG19	224x224	channels_first / channels_last	caffe
ResNet50	224x224	channels_first / channels_last	caffe
InceptionV3	299x299	channels_first / channels_last	tf
InceptionResNetV2	299x299	channels_first / channels_last	tf
MobileNet	224x224	channels_last	tf
DenseNet	224x224	channels_first / channels_last	torch
NASNet	331x331 / 224x224	channels_first / channels_last	tf
MobileNetV2	224x224	channels_last	tf

Pre-train models of tf.Keras includes Xception、VGG16、VGG19、ResNet50、InceptionV3、InceptionResNetV2、Mobi
MobileNetV2

```
arr = []
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.python.keras.applications.imagenet_utils import preprocess_input
for img in train_images:
    x = image.img_to_array(img)
    img_array = preprocess_input(x, mode = 'caffe' )
    arr.append(img_array)
X_train = np.array(arr)
print(X_train.shape)
```

```
(2528, 224, 224, 3)
```

preprocess_input(x, mode = 'caffe')

Step 13:One-hot encoding for labels

Step 13:One-hot encoding for labels

```
from tensorflow.keras.utils import to_categorical
# one-hot encoding
num_classes = 6
y_train = to_categorical(train_labels, num_classes)
print(y_train)
```

```
[[1. 0. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 ...
 [1. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 1. 0. 0.]
 [0. 0. 0. ... 1. 0. 0.]]
```

train_labels, num_classes

Step 14: define the CNN model

Step 14: define the CNN model

```
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
from tensorflow.keras.applications.vgg19 import VGG19
```

```
#Use the VGG19 model
image_input = Input(shape=(224, 224, 3))
###  CODE HERE ###  (≈ 1 lines)
vgg_model = VGG19(input_tensor=image_input, include_top=True, weights='imagenet')
vgg_model.summary()
```

```
last_layer = vgg_model.get_layer('fc2').output
out = Dense(num_classes, activation='softmax', name='output')(last_layer)
model = Model(image_input, out)
for layer in model.layers[:-1]:
    layer.trainable = False
```

layer.trainable = False

Step 15: compile the model

Step 15: compile the model

```
#compile model using accuracy to measure model performance
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

optimizer='rmsprop'
loss='categorical_crossentropy'

Step 16: fit the model

Step 16: fit the model

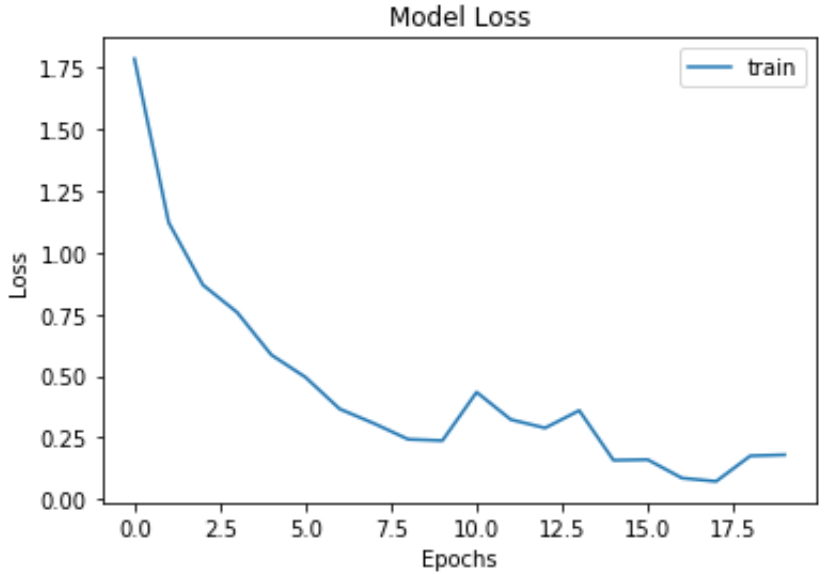
```
#train the model
hist = model.fit(X_train, y_train, batch_size=20, epochs=20)
```

X_train, y_train

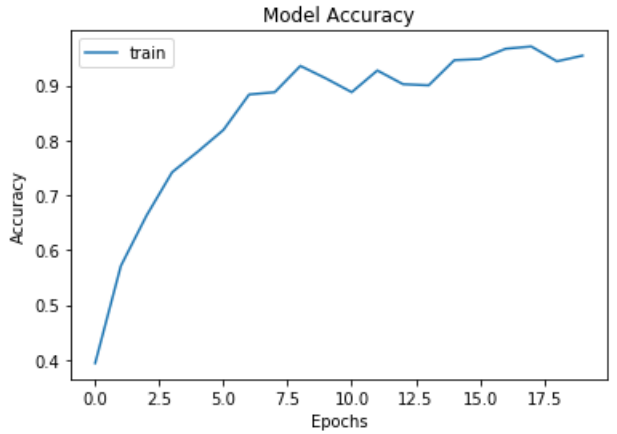
Step 17: evaluate the model

Step 17: evaluate the model ¶

```
plt.plot(hist.history['loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train'])
plt.show()
```




```
plt.plot(hist.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train'])
plt.show()
```




```
#if tf.__version__ < "2.x":
plt.plot(hist.history['acc'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train'])
plt.show()
```

Step 18: predict with the model for the training set

Step 18: predict with the model for the training set ¶

```
y_prediction = model.predict(X_train, batch_size=20)
print(y_prediction[:2])
```

```
[[6.9206482e-01 3.2412426e-03 8.9412570e-02 2.1242268e-01 2.7442379e-03
  5.7099078e-05 3.9558348e-05 1.7788072e-05]
 [3.3183892e-06 9.8436850e-01 1.5163666e-03 1.4108860e-02 2.8986462e-06
  1.7211991e-07 1.6921112e-09 1.5864232e-09]]
```

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

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

```
print(train_labels[:10])
```

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

X_train

np.argmax(y_prediction,axis=1)

Step 19: Compute confusion matrix (混淆矩陣)

Step 19: Compute confusion matrix (混淆矩陣)

train_labels, predict

```
from sklearn.metrics import confusion_matrix
confusion=confusion_matrix(train_labels, predict)
print(confusion)
```

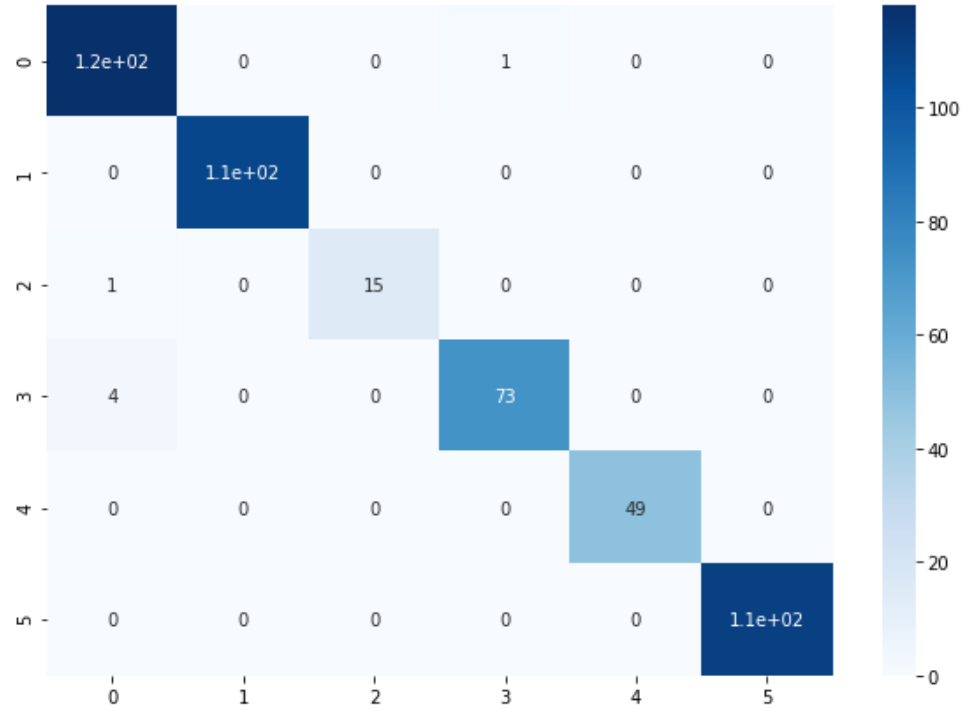
```
[[118   0   0   1   0   0]
 [  0 108   0   0   0   0]
 [  1   0  15   0   0   0]
 [  4   0   0  73   0   0]
 [  0   0   0   0  49   0]
 [  0   0   0   0   0 111]]
```

Step 20: Plot the confusion matrix

Step 20: Plot the confusion matrix

```
import seaborn as sn
df_cm = pd.DataFrame(confusion)
plt.figure(figsize = (10,7))
sn.heatmap(df_cm, annot=True, cmap="Blues")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f29000818d0>
```



df_cm

Step 21: List overkills and underkills

Step 21: List overkills and underkills

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

```
# of overkill= 1; # of underkill= 5
```

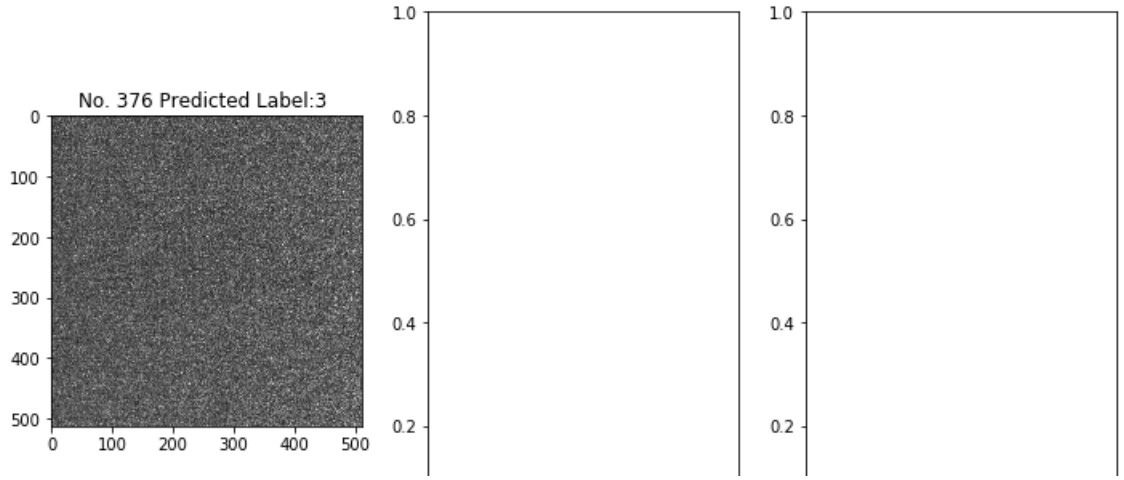
predict[i] !=o

predict[i] ==o

Step 22: Check overkills

Step 22: Check overkills

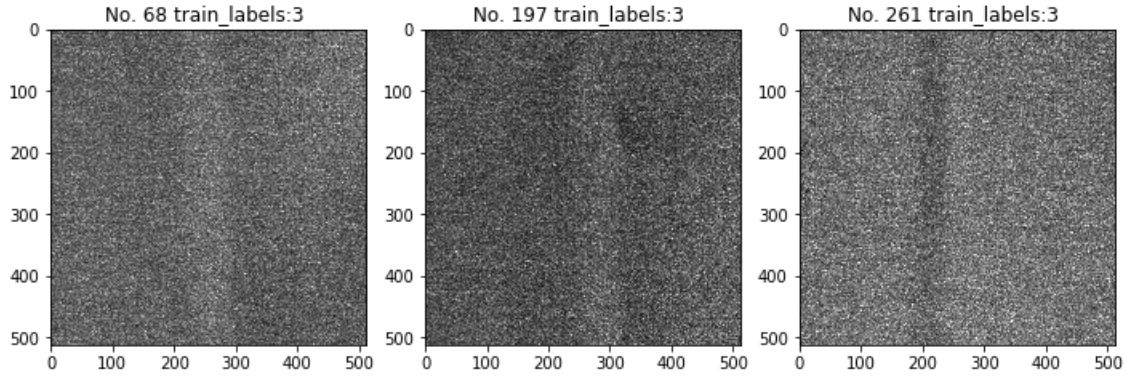
```
#overkill
overkill_num = len(overkill)
no = 0
fig,ax=plt.subplots(2, 3)
fig.set_size_inches(10,10)
for i in range(2):
    for j in range (3):
        if no>=overkill_num:
          break
        sel = overkill[no]
        ax[i,j].imshow(train_images[sel], cmap='gray')
        ax[i,j].set_title('No. {} Predicted Label:{} '.format(sel, predict[sel]))
        no += 1
plt.tight_layout()
```



Step 23: Check underkills

Step 23: Check underkills

```
#underkill
underkill_num = len(underkill)
no = 0
fig,ax=plt.subplots(2, 3)
fig.set_size_inches(10,10)
for i in range(2):
    for j in range (3):
        if no>=underkill_num:
          break
        sel = underkill[no]
        ax[i,j].imshow(train_images[sel], cmap='gray')
        ax[i,j].set_title('No. {} train_labels:{} '.format(sel, train_labels[sel]))
        no += 1
plt.tight_layout()
```



Step 24: Save the model

Step 24: Save the model

```
model.save("AOIVGG_20epochs-2020.h5")
```

```
model = tf.keras.models.load_model('AOIVGG_20epochs-2020.h5')
model.summary()
```

Step 25: Delete training data in memory

Step 25: Delete training data in memory

```
del train_images
del X_train
#Do GC
import gc
gc.collect()
```

Step 26: read lalels of the test set

Step 26: read lalels of the test set

```
df_test = pd.read_csv(data_path+ "test.csv")
print(df_test.shape)
```

```
df_test.head()
```

test.csv

Step 27: Build the lists of test images and labels from the dataframe

Step 27: Build the lists of test images and labels from the dataframe

```
test_num = 480 #limit the amount of training images for the class process
#test_num = df_test.shape[0]
if test_num >= df_test.shape[0]:
  test_num = df_test.shape[0]
test_files  = df_test.iloc[:test_num,0].values
test_labels = df_test.iloc[:test_num,1].values
print(test_labels[:10])
```

480

Step 28: read images of the test set

Step 28: read images of the test set

```
!ls '/content/drive/My Drive/aidea-aoi2/data/test_images/'
```

```
test_path = data_path+ "test_images/"
test_images = []
from tensorflow.keras.preprocessing import image
for file in test_files:
    img = image.load_img(test_path+file, color_mode="rgb", target_size = (224, 224))
    test_images.append(img)
    if len(test_images)%100 == 0:
      print('.', end='')
print(len(test_images))
```

```
.....................................................................................................10142
```

rgb

Step 29: show AOI test images

Step 29: show AOI test images:

```
import random
fig,ax=plt.subplots(2,3)
fig.set_size_inches(10,10)
for i in range(2):
    for j in range (3):
        sel=random.randint(0,test_num)
        ax[i,j].imshow(test_images[sel], cmap='gray')
        ax[i,j].set_title('No. {} Label:Nan '.format(sel))
plt.tight_layout()
```

Step 30: Convert each test image into a numpy array and collect

Step 30: Convert each test image into a numpy array and collect

```
arr = []
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.python.keras.applications.imagenet_utils import preprocess_input
for img in test_images:
    x = image.img_to_array(img)
    img_array = preprocess_input(x, mode = 'caffe' )
    arr.append(img_array)
X_test = np.array(arr)
print(X_test.shape)
```

preprocess_input(x, mode = 'caffe')

Step 31: predict with the model for the test set

Step 31: predict with the model for the test set

```
y_prediction = model.predict(X_test, batch_size=20)
predict = np.argmax(y_prediction,axis=1)
print(predict[:20])
```

np.argmax(y_prediction,axis=1)

Step 32: show predictions

Step 32: show predictions

```
import random
fig,ax=plt.subplots(4,4)
fig.set_size_inches(10,10)
for i in range(4):
    for j in range (4):
        sel=random.randint(0,len(test_images))
        ax[i,j].imshow(test_images[sel], cmap='gray')
        ax[i,j].set_title('No. {} Predicted Label:{} '.format(sel, predict[sel]))
plt.tight_layout()
```

Step 33: output predictions

Step 33: output predictions

```
df_out = pd.DataFrame(df_test.iloc[:test_num])
df_out['Label'] = predict
df_out.to_csv("submission-20200114B.csv", index=False)
```

'Label'