AOI Course
hands-on training (1)

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

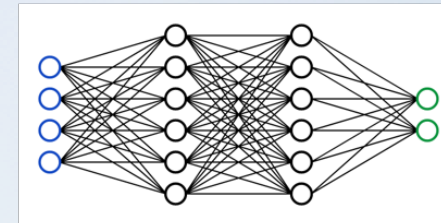# Tutorial 1: training a full CNN model for AOI

(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

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

import tensorflow as tf
print(tf.__version__)
```
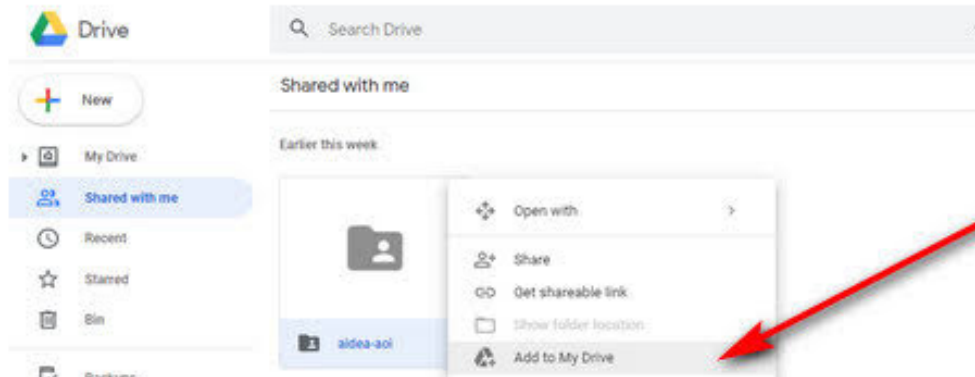
```
TensorFlow 2.x selected.
2.1.0-rc1
```

%tensorflow_version 2.x

Step 2: Add the data folder to your Google drive

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

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

```python
#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

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

(2528, 2)

```python
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

```python
#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

```python
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="grayscale")
    train_images.append(img)
    if len(train_images)%100 == 0:
        print('.', end='')
print(len(train_images))
```

"grayscale"

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)

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

```python
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

```python
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

```python
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()
```
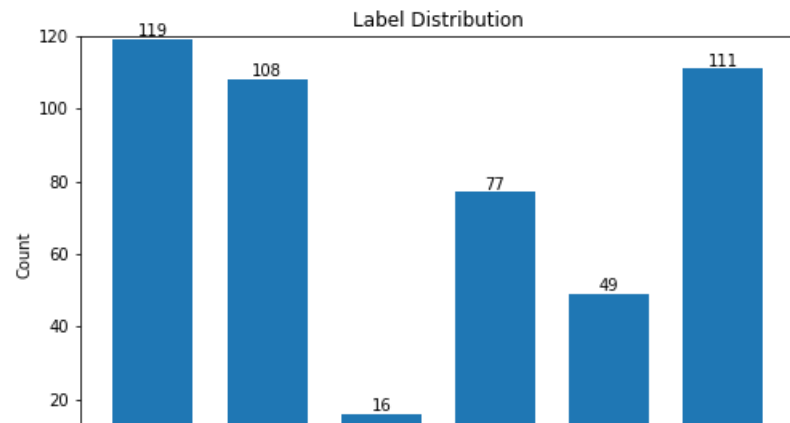

labels, counts

Step 11: Check the shape of single image

Step 11: Check the shape of single image

```python
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

```python
arr = []
from tensorflow.keras.preprocessing.image import img_to_array
for img in train_images:
    img_array = img_to_array(img)/127.5 -1
    arr.append(img_array)
X_train = np.array(arr)
print(X_train.shape)
```

```
(480, 512, 512, 1)
```

```python
# The pixel value in [0,1)
print(X_train[0, 0 , 0 , 0])
```

```
0.3411765
```

img_array = img_to_array(img)/127.5 -1

arr.append(img_array)

Step 13:One-hot encoding for labels

```python
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

```python
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
```

```python
#create model
model = Sequential()
#add model layers

model.add(Conv2D(filters = 32, kernel_size = (5,5),activation ='relu', input_shape = (512,512,1)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters =96, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters = 96, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dense(6, activation = "softmax"))
```

model = Sequential()

Step 15: compile the model

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

optimizer='adam'
loss='categorical_crossentropy'

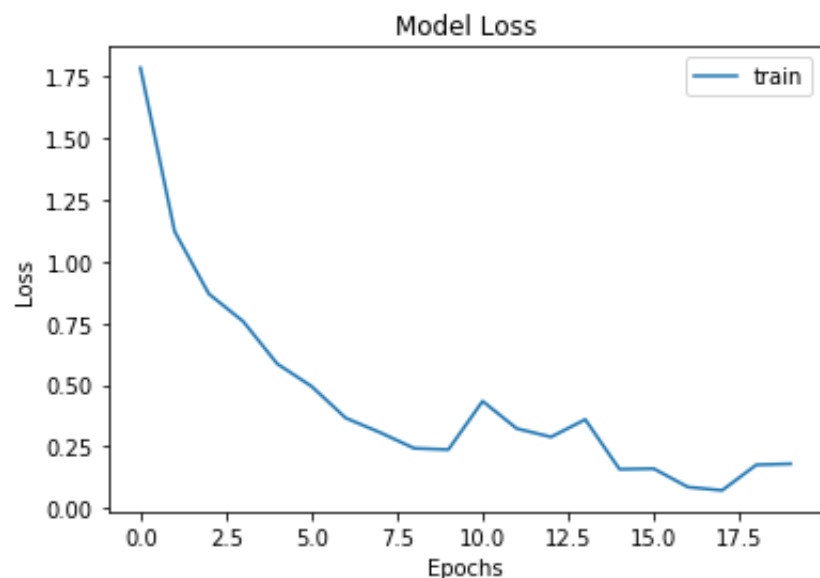# Step 16: fit the model

Step 16: fit the model

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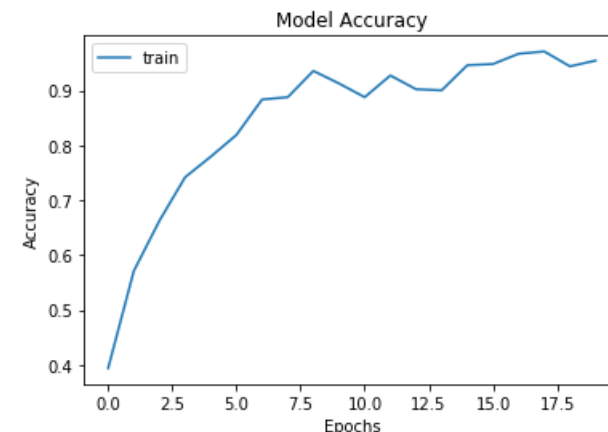
X_train, y_train

Step 17: evaluate the model

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

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

```python
#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 ¶

```python
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]]
```

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

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

```python
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

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

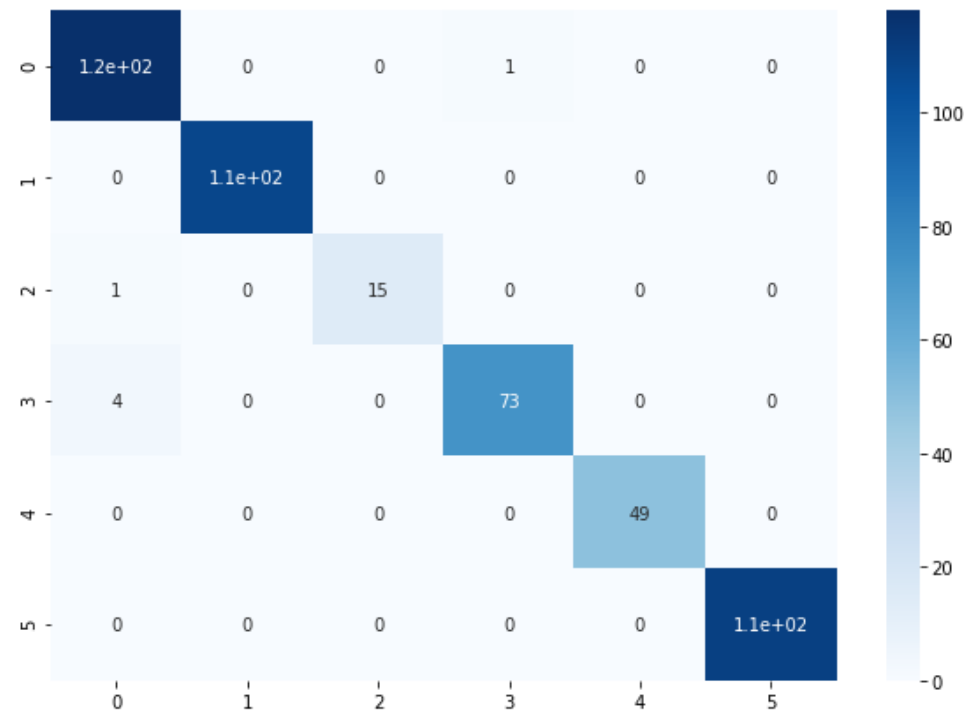```
[[118   0   0   1   0   0]
 [0 108   0   0   0   0]
 [1   0 15   0   0   0]
 [4   0   0 73   0   0]
 [0   0   0   0 49   0]
 [0   0   0   0   0 111]]
```

# Step 20: Plot the confusion matrix



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
```
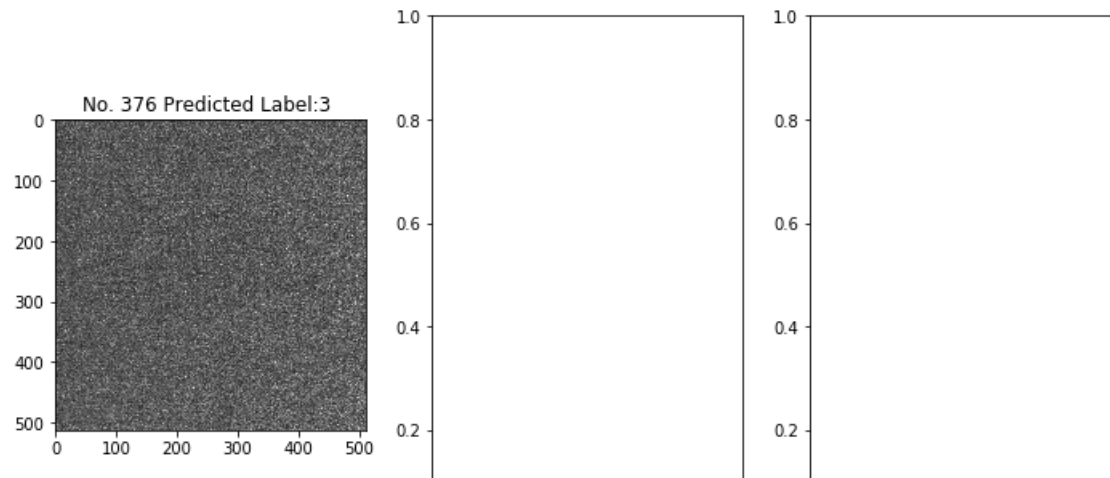
predict[i] !=0

predict[i] ==0

# 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**

```python
#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()
```

No. 68 train_labels:3   No. 197 train_labels:3   No. 261 train_labels:3
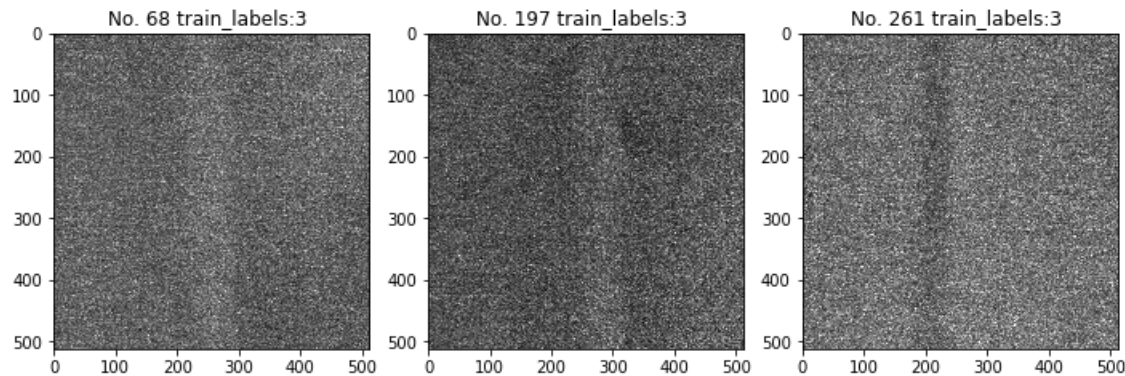
Step 24: Save the model

```
model.save("AOICNN_10epochs-2020.h5")
```

```
model = tf.keras.models.load_model('AOICNN_10epochs-2020.h5')
model.summary()
```

Step 25: Delete training data in memory

Step 25: Delete training data in memory

```python
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

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

```python
df_test.head()
```

test.csv

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

```python
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/'
```

```python
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="grayscale")
    test_images.append(img)
    if len(test_images)%100 == 0:
        print('.', end='')
print(len(test_images))
```

grayscale

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

```python
arr = []
from tensorflow.keras.preprocessing import image
for img in test_images:
    img_array = img_to_array(img)/127.5 -1
    arr.append(img_array)
X_test = np.array(arr)
print(X_test.shape)
```

```python
print(X_test[0, 0 , 0 , 0])
```

img_array = img_to_array(img)/127.5 -1
arr.append(img_array)

Step 31: predict with the model for the test set

```python
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

```python
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

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

'Label'