CN240 Data Science for Signal Processing

Code Explanation: Deep Learning

1. Chonlasit Mooncorn 6110613020
2. Prach Chantasantitam 6110613202
3. Weeraphat Leelawittayanon 6210612823
4. Raned Chuphueak 6210612864

Import Library ที่ใช้สำหรับการเทรนโมเดล

```
import numpy as np
import pandas as pd
import os
from sklearn.model_selection import KFold, StratifiedKFold
import tensorflow as tf
from keras.layers import Input, Lambda, Dense, Flatten, Conv2D, MaxPool2D
from keras.models import Model
from keras import applications
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential

import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve
from sklearn.metrics import confusion_matrix
```

Input Data เข้ามาโดยส่วนของการ Input จะมีการไปอ่านไฟล์ CSV ที่เก็บ Filename และ Label ของไฟล์นั้น ๆ ไว้เพื่อที่จะนำมาแบ่งเป็น 5 Fold

```
train_data = pd.read_csv('dataset_train.csv')
#default label 0 = glaucoma 1 = normal 2 = others
train_data.loc[train_data.label == 0, 'label'] = 3
train_data.loc[train_data.label == 2, 'label'] = 0
train_data.loc[train_data.label == 1, 'label'] = 0
train_data.loc[train_data.label == 3, 'label'] = 1
train_data['label'] = train_data['label'].astype(str)
Y = train_data[['label']]
skf = StratifiedKFold(n_splits = 5, random_state = 7, shuffle = True)


idg = ImageDataGenerator(rescale = 1./255,
                         shear_range = 0.2,
                         zoom_range = 0.2,
                         horizontal_flip = True)

print(f'Train dataset : {len(Y)} images')
```

Function: create_new_model

ทำการสร้างโมเดลขึ้นมาใหม่จากตัวโมเดลที่ได้รับการเทรนมาก่อนหน้า เช่น DenseNet121 หรือ VGG16 ซึ่งการทำแบบนี้เรียกว่า Transfer learning ในรูปทำการยกตัวอย่างเป็นตัว DenseNet121 ซึ่งหากต้องการเปลี่ยนเป็น Pre-trained Model ตัวอื่น เช่น VGG16 ให้ทำการเปลี่ยน ในส่วนของ DenseNet121 เป็น VGG16 แทน

```
def create_new_model():
    IMAGE_SIZE = [32, 32]
    DenseNet121 = applications.DenseNet121(input_shape=IMAGE_SIZE + [3]
                                           , weights='imagenet', include_top=False)
    for layer in DenseNet121.layers:
        layer.trainable = False
    x = Flatten()(DenseNet121.output)
    prediction = Dense(1, activation='sigmoid')(x)
    model = Model(inputs=DenseNet121.input, outputs=prediction)
    return model

def get_model_name(k):
    return 'model_'+str(k)+'.h5'
```

Function: main

ทำการ split ข้อมูลออกมาเป็นจำนวน 5 Fold และ Input ข้อมูลผ่าน ImageDataGenerator.flow_from_dataframe

```
def main():
    VALIDATION_ACCURACY = []
    VALIDATION_LOSS = []

    image_dir = 'Dataset/dataset_train'
    save_dir = 'Models/gsaved_models/'
    fold_var = 1
    num_epochs = 80

    for train_index, val_index in skf.split(np.zeros(2313),Y):
        training_data = train_data.iloc[train_index]
        validation_data = train_data.iloc[val_index]

        train_data_generator = idg.flow_from_dataframe(training_data, directory = image_dir,
                                                       x_col = 'filename', y_col = 'label',
                                                       target_size = (32, 32),batch_size = 80,
                                                       class_mode = "binary", shuffle = True)
        valid_data_generator  = idg.flow_from_dataframe(validation_data, directory = image_dir,
                                                        x_col = 'filename', y_col = 'label',
                                                        target_size = (32, 32),batch_size = 80,
                                                        class_mode = "binary", shuffle = True)
```

เป็นส่วนของการเรียกใช้ Function create_new_model() ที่สร้างขึ้นมาก่อนหน้า และทำการสร้างจุด Checkpoint ไว้เป็นจุดเซฟ Model ในโฟลด์นั้น ๆ โดยจะเซฟ Model ใหม่ทุกครั้งที่ Fit model แล้วค่า val_accuracy เพิ่มขึ้น

```
# CREATE NEW MODEL
model = create_new_model()

# COMPILE NEW MODEL
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# CREATE CALLBACKS
checkpoint = tf.keras.callbacks.ModelCheckpoint(save_dir + get_model_name(fold_var),
                                                monitor='val_accuracy', verbose=1,
                                                save_best_only=True, mode='max')
callbacks_list = [checkpoint]
```

Fit model

```
# FIT THE MODEL
history = model.fit(train_data_generator,
                    epochs=80,
                    callbacks=callbacks_list,
                    validation_data=valid_data_generator)
```

ทำการแยกค่าที่อยู่ในส่วน validation มาทำการพล็อตกราฟ ROC

```
# Extract valid_data_generator
valid_data_generator.reset()

X_test, y_test = next(valid_data_generator)

batch_index = 0
while batch_index <= valid_data_generator.batch_index:
    img, label = next(valid_data_generator)
    X_test = np.append(X_test, img, axis=0 )
    y_test = np.append(y_test, label, axis=0)
    batch_index = batch_index + 1
```

เก็บข้อมูลสำหรับ Plot กราฟ ROC

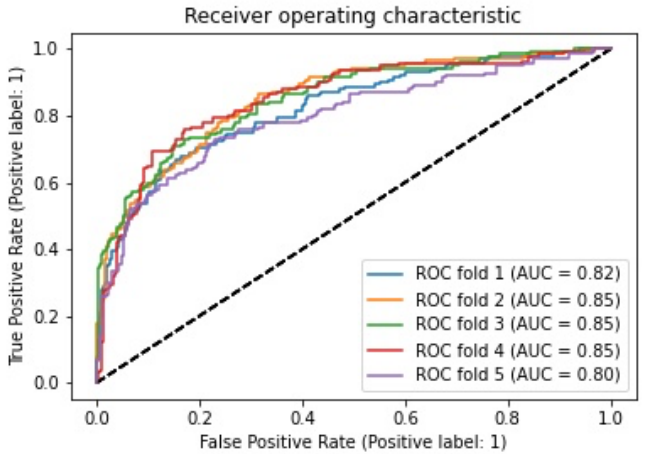
```
# PLOT HISTORY
y_pred = model.predict(X_test).ravel()
fpr_keras, tpr_keras, thresholds_keras = roc_curve(y_test, y_pred)

from sklearn.metrics import auc
auc_keras = auc(fpr_keras, tpr_keras)

plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='ROC fold {} (AUC = {:.2f})'.format(fold_var,
                                                                        auc_keras))
```

โหลด โมเดลที่ดีที่สุดมาทำการ evaluate เพื่อดูประสิทธิภาพของตัวโมเดลในโฟลต์นั้น

```
# LOAD BEST MODEL to evaluate the performance of the model
model.load_weights("Models/gsaved_models/model_" + str(fold_var) + ".h5")

results = model.evaluate(valid_data_generator)
results = dict(zip(model.metrics_names, results))


VALIDATION_ACCURACY.append(results['accuracy'])
VALIDATION_LOSS.append(results['loss'])

tf.keras.backend.clear_session()
```

ทำการพล็อตกราฟ ROC และทำการเซฟ

```
plt.xlabel('False Positive Rate (Positive label: 1)')
plt.ylabel('True Positive Rate (Positive label: 1)')
plt.title('Receiver operating characteristic')
plt.legend(loc='best')
plt.savefig('Graph/DenseNet121_glaucoma_graph.jpg')
plt.show()
```



Plot Confusoin Matrix

ทำการโหลด datasetที่เป็นส่วนที่แบ่งไว้สำหรับการทดสอบโดยเฉพาะมา

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
test_data = pd.read_csv('dataset_test.csv')
#default label 0 = glaucoma 1 = normal 2 = others
test_data.loc[test_data.label == 0, 'label'] = 3
test_data.loc[test_data.label == 2, 'label'] = 0
test_data.loc[test_data.label == 1, 'label'] = 0
test_data.loc[test_data.label == 3, 'label'] = 1
test_data['label'] = test_data['label'].astype(str)

image_dir = 'Dataset/dataset_test'

test_generator = ImageDataGenerator(rescale=1./255)

test_data_generator  = test_generator.flow_from_dataframe(test_data, directory = image_dir,
                                                          x_col = 'filename', y_col = 'label',
                                                          target_size = (32, 32),batch_size = 16,
                                                          class_mode = "binary", shuffle = True)

test_data_generator.reset()
```

จากนั้นทำการแยกข้อมูลออกจากตัว ImageDataGenerator ที่เราทำการเรียกใช้เพื่อที่จะจัดการข้อมูล

```
X_test, y_test = next(test_data_generator)

batch_index = 0
while batch_index <= test_data_generator.batch_index:
    img, label = next(test_data_generator)
    X_test = np.append(X_test, img, axis=0 )
    y_test = np.append(y_test, label, axis=0)
    batch_index = batch_index + 1
```

จากนั้นทำการโหลดโมเดล และทำการ predict

```
# CREATE NEW MODEL
model = create_new_model2()

# COMPILE NEW MODEL
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.load_weights("Models/gsaved_models/model_4.h5")
y_pred = model.predict(X_test)
y_pred = np.transpose(y_pred)[0]
y_pred = list(map(lambda x: 0 if x < 0.5 else 1, y_pred))
```

เอาค่าที่ได้จากขั้นตอนที่แล้วมาทำการสร้าง Confusion Matrix ขึ้นแล้วทำการเซฟ

```
from sklearn.metrics import classification_report
import seaborn as sn
print(classification_report(y_test, y_pred))

data = confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(data,index = np.unique(y_test))
plt.figure(figsize = (6,5))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, cmap="Blues", annot=True,annot_kws={"size": 16},fmt="d")
plt.savefig('Matrix/DenseNet121_glaucoma_matrix.jpg')

print(data)
```