train.py

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout, AveragePoo
ling2D, GlobalAveragePooling2D
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.optimizers import Adam
print(tf.__version__)

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import csv
import scipy.stats as stats

from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
from decimal import Decimal

processedList = []
tempi = 0
# 這邊下面在進行讀檔。
with open("data.csv", "r") as f:
    for line in f:
        sepline = line.split(",")
        sepline[5] = sepline[5].replace("\n","")
        # 第一行標籤忽略
        if tempi == 0:
            tempi = 1
            continue
        temp = [sepline[0], sepline[1], sepline[2], sepline[3],sepline[
4],sepline[5]]
        processedList.append(temp)
# 這邊再將資料分成 time , x , y, z ,total ,label 的標籤
columns = ['time','x', 'y', 'z','total','label']
```

```
data = pd.DataFrame(data = processedList, columns = columns)
# 以下註解的 print()都是為了確認數值，可以打開來看看。

# print(data.head())
# print(data.shape)
# print(data.info())
# print(data.isnull().sum())
# print(data['label'].value_counts())

data['x'] = data['x'].astype('float')
data['y'] = data['y'].astype('float')
data['z'] = data['z'].astype('float')
Fs = 100 # Hz 的數字。
frame_size = Fs*2 # 200 # 乘以二是為了兩秒。
hop_size = Fs*1 # 100 # 跳躍點，為了重疊資料而生。
activities = data['label'].value_counts().index
# 把三類的資料放進 activites
# print(activities)

def plot_activity(activity, data):
    fig, (ax0, ax1, ax2) = plt.subplots(nrows=3, figsize=(10, 6), share
x=True)
    plot_axis(ax0, data['time'], data['x'], 'X-Axis')
    plot_axis(ax1, data['time'], data['y'], 'Y-Axis')
    plot_axis(ax2, data['time'], data['z'], 'Z-Axis')
    plt.subplots_adjust(hspace=0.2)
    fig.suptitle(activity)
    plt.subplots_adjust(top=0.90)
    plt.show()

def plot_axis(ax, x, y, title):
    ax.plot(x, y, color=(255/255,100/255,100/255))
    ax.set_title(title)
    ax.xaxis.set_visible(False)
    ax.set_ylim([min(y) - np.std(y), max(y) + np.std(y)])
    ax.set_xlim([min(x), max(x)])
    ax.grid(True)
# 這邊上面都在做畫圖的 function，不贅述。
```

```
for activity in activities:
    data_for_plot = data[(data['label'] == activity)][:Fs*10]
    plot_activity(activity, data_for_plot)
# 這邊再畫圖


df = data.drop(['total','time'], axis = 1).copy()
# 把不需要的值丟掉
# print(df.head())
# print(df['label'].value_counts())
walk = df[df['label']=='walk'].copy()
scooter = df[df['label']=='scooter'].copy()
MRTt = df[df['label']=='MRT'].copy()
# 把 df 的參數載入

col_count = 1
bar_width = 0.2
x_scores = (len(walk.values))
y_scores = (len(scooter.values))
z_scores = (len(MRTt.values))
index = np.arange(col_count)
A = plt.bar(index,
           y_scores,
           bar_width,
           alpha=.4,
           label="scooter")
B = plt.bar(index+0.3,
            x_scores,
            bar_width,
            alpha=.4,
            label="walk")
C = plt.bar(index+0.6,
            z_scores,
            bar_width,
            alpha=.4,
            label="MRT") # x,y ,width
def createLabels(data):
    for item in data:
```

```
        height = item.get_height()
        plt.text(
            item.get_x()+item.get_width()/2.,
            height*1.05,
            '%d' % int(height),
            ha = "center",
            va = "bottom",
        )
createLabels(A)
createLabels(B)
createLabels(C)
plt.tick_params(
    axis='x',
    which='both',
    bottom=False,
    top=False,
    labelbottom=False)
plt.ylabel("times")
plt.title("Test Data")
plt.legend()
plt.grid(True)
plt.show()
#畫第二張圖


balanced_data = pd.DataFrame()
balanced_data = balanced_data.append([walk, scooter, MRTt])
# 放進一個變數中

# print(balanced_data.shape)
# print(balanced_data['label'].value_counts())
# print(balanced_data.head())

label = LabelEncoder()
balanced_data['label'] = label.fit_transform(balanced_data['label'])
# 把 label 變成 0,1,2

# print(balanced_data.head())
# print(label.classes_)
```

```
X = balanced_data[['x', 'y', 'z']]
y = balanced_data['label']
# 把值存入 X,y
scaler = StandardScaler()
X = scaler.fit_transform(X)
# 計算均值何方差，然後標準化。

scaled_X = pd.DataFrame(data = X, columns = ['x', 'y', 'z'])
scaled_X['label'] = y.values
# 這邊是把它存成 panda 的資料型態。

# print(scaled_X.head())
def get_frames(df, frame_size, hop_size):

    N_FEATURES = 3

    frames = []
    labels = []
    for i in range(0, len(df) - frame_size, hop_size):
        x = df['x'].values[i: i + frame_size]
        y = df['y'].values[i: i + frame_size]
        z = df['z'].values[i: i + frame_size]
        # 這行是做把兩秒兩秒的資料放在一個 array

        label = stats.mode(df['label'][i: i + frame_size])[0][0]
        # 這行是為了符合前一項，就取 label 中出現最多次的，去當作我們的
label。(相接處需要用到)

        frames.append([x, y, z])
        labels.append(label)
        # 一起 append 是為了讓資料量相同

    frames = np.asarray(frames).reshape(-1, frame_size, N_FEATURES)
    # 這行是先把上面的 array 變成一行，然後做成 200x3 的陣列。
    labels = np.asarray(labels)
    # 這行是為了讓 label 跟上面的資料型態一樣。
```

```
    return frames, labels

X, y = get_frames(scaled_X, frame_size, hop_size)
#把這兩個丟進 X,y。

# print(X.shape, y.shape)

#資料總數/Hz*秒數 = 處理後的資料總數(也就是下方的 859+215)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0
.2, random_state = 0, stratify = y)
# 這邊是將程式分成 train 跟 validation

print(X_train.shape, X_test.shape)
print(X_train[0].shape, X_test[0].shape)

# 藉由上面的 print 去看出怎麼分的，並且把 X_train 跟 X_test reshape，為了讓數
據符合 CNN 的 model 以及符合他的內容值，放入 3 維的值
X_train = X_train.reshape(859, 200, 3, 1)
X_test = X_test.reshape(215, 200, 3, 1)

# print(X_train[0].shape, X_test[0].shape)
# 檢查後會變成(200,3,1)

model = Sequential()
# 建立一個順序模型，是最簡單的模型，從頭到尾的結構順序。
model.add(Conv2D(filters=16,kernel_size=(3,3),strides=(1,1),padding="sa
me", activation = 'relu', input_shape = X_train[0].shape))
# 這是 input 層
# 使用 Relu 函數去掉負值，更能淬煉出物體的形狀
model.add(AveragePooling2D(pool_size=(2,2)))
#AveragePooling2D 是為了平均整體值，減少誤差。
model.add(Dropout(0.5))
# Dropout 就是在不同的訓練過程中隨機扔掉一部分神經元，也就是暫時不更新權重。
# Dropout 是用來避免過度配適（Overfitting）的。

model.add(Conv2D(filters=32,kernel_size=(3,3),strides=(1,1),padding="sa
me", activation='relu'))
model.add(Dropout(0.5))
```

```
# 第二層 Convolution 層（32 個神經元）
model.add(GlobalAveragePooling2D())
#model.add(Flatten())
# 將特徵值平攤，GlobalAveragePooling2D 去做。

model.add(Dense(64, activation = 'relu'))
model.add(Dropout(0.5))
# 第三層 Dense 層（64 個神經元）

model.add(Dense(3, activation='softmax'))
# output 3 種結果

model.compile(optimizer=Adam(learning_rate = 0.001), loss = 'sparse_cat
egorical_crossentropy', metrics = ['accuracy'])
# 以 compile 函數定義損失函數(loss)、優化函數(optimizer)及成效衡量指標
(mertrics)

# 進行訓練, 訓練過程會存在 train_history 變數中
history = model.fit(X_train, y_train, batch_size = 16, epochs = 100, va
lidation_data= (X_test, y_test), verbose=1)

def plot_learningCurve(history, epochs):

  epoch_range = range(1, epochs+1)
  plt.plot(epoch_range, history.history['accuracy'])
  plt.plot(epoch_range, history.history['val_accuracy'])
  plt.title('Model accuracy')
  plt.ylabel('Accuracy')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Val'], loc='upper left')
  plt.show()

  plt.plot(epoch_range, history.history['loss'])
  plt.plot(epoch_range, history.history['val_loss'])
  plt.title('Model loss')
  plt.ylabel('Loss')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Val'], loc='upper left')
```

```
  plt.show()

plot_learningCurve(history, 100)
model.summary()
#上面是把圖跟架構畫出來

#print(model.predict(X_test))

# 預測(prediction)
y_pred = model.predict_classes(X_test)

# print(X_test.shape)

mat = confusion_matrix(y_test, y_pred)
# print(label.classes_)
plot_confusion_matrix(conf_mat=mat, class_names=label.classes_, show_no
rmed=True, figsize=(7,7))
test_accuracy = 0
for i in range(len(y_pred)):
    if y_test[i] == y_pred[i]:
        test_accuracy += 1
test_accuracy /= len(y_pred)
test_accuracy *= 100

print("mission success rate = "+str(Decimal(test_accuracy).quantize(Dec
imal("0.00")))+"%")
plt.show()
# 畫出 confusion_matrix 去看看對的有哪些
model.save('my_model.h5')
# 把 model 存起來

print("work!")
# 結束了!! print 一個 work!代表成功
```

test.py

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

```
from tensorflow.keras.models import load_model
# from keras import load_model
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, LabelEncoder
import csv
import scipy.stats as stats
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
from decimal import Decimal
model = load_model('my_model_10epochs.h5')
processedList = []
tempi = 0
with open("testData.csv", "r") as f:
    for line in f:
        sepline = line.split(",")
        sepline[5] = sepline[5].replace("\n","")
        if tempi == 0:
            tempi = 1
            continue
        temp = [sepline[0], sepline[1], sepline[2], sepline[3],sepline[
4],sepline[5]]
        processedList.append(temp)
columns = ['time','x', 'y', 'z','total','label']
data = pd.DataFrame(data = processedList, columns = columns)
data['x'] = data['x'].astype('float')
data['y'] = data['y'].astype('float')
data['z'] = data['z'].astype('float')

df = data.drop(['total','time'], axis = 1).copy()
walk = df[df['label']=='walk'].copy()
scooter = df[df['label']=='scooter'].copy()
MRTt = df[df['label']=='MRT'].copy()
balanced_data = pd.DataFrame()
balanced_data = balanced_data.append([walk, scooter, MRTt])
label = LabelEncoder()
balanced_data['label'] = label.fit_transform(balanced_data['label'])
```

```
X = balanced_data[['x', 'y', 'z']]
y = balanced_data['label']
scaler = StandardScaler()
X = scaler.fit_transform(X)

scaled_X = pd.DataFrame(data = X, columns = ['x', 'y', 'z'])
scaled_X['label'] = y.values


def get_frames(df, frame_size, hop_size):
    N_FEATURES = 3

    frames = []
    labels = []
    for i in range(0, len(df) - frame_size, hop_size):
        x = df['x'].values[i: i + frame_size]
        y = df['y'].values[i: i + frame_size]
        z = df['z'].values[i: i + frame_size]

        label = stats.mode(df['label'][i: i + frame_size])[0][0]
        frames.append([x, y, z])
        labels.append(label)

    frames = np.asarray(frames).reshape(-1, frame_size, N_FEATURES)
    labels = np.asarray(labels)

    return frames, labels
Fs = 100
frame_size = Fs*2
hop_size = Fs*1
scaled_X = pd.DataFrame(data = X, columns = ['x', 'y', 'z'])
scaled_X['label'] = y.values
X, y = get_frames(scaled_X, frame_size, hop_size)
print(X.shape)
X = X.reshape(528, 200, 3, 1)
y_pred = model.predict_classes(X)
mat = confusion_matrix(y, y_pred)
plot_confusion_matrix(conf_mat=mat, class_names=label.classes_, show_no
rmed=True, figsize=(7,7))
```

```
test_accuracy = 0
for i in range(len(y_pred)):
    if y[i] == y_pred[i]:
        test_accuracy += 1
test_accuracy /= len(y_pred)
test_accuracy *= 100

model.summary()
print("mission success rate = "+str(Decimal(test_accuracy).quantize(Dec
imal("0.00")))+"%")
#多了成功率
plt.show()
```