SPEECH ANALYSIS II

版本與套件

- **Python 3.6.8**
- pip install tensorflow　　#用於模型建構 >= 2
- pip install sklearn　　#使用sklearn的函式分割出訓練集與測試集
- pip install librosa　　#用於提取語音的特徵

檔案介紹

https://drive.google.com/file/d/1mwoKSUO3xnNtlOYnMbR6guW2fgtjdZ8o/view?usp=sharing

- data – 是資料集
  - *bed*
  - *cat*
  - *happy*
- bed.npy / cat.npy / happ.npy 是上述音檔的 mfcc 特徵存檔
- ASR.h5 - 訓練好的模型
- preprocess.py – 語音前處理程式
- train.py – 訓練程式
- test.py – 測試程式

資料集介紹

- bed 語音共有1,713筆資料
- cat 語音共有1,733筆資料
- cat 語音共有1,742筆資料

載入函式庫

```
1  import librosa
2  import os
3  from sklearn.model_selection import train_test_split
4  from tensorflow.keras.utils import to_categorical
5  import numpy as np
6
7  DATA_PATH = "./data/"
```

取得語音檔的標籤

使用 one hot encoding 將標籤 [‘bed’, ‘cat’, ‘happy’] 轉換成類別 [0, 1, 2]

```
 9  # Input: Folder Path
10  # Output: Tuple (Label, Indices of the labels, one-hot encoded labels)
11  def get_labels(path=DATA_PATH):
12      labels = os.listdir(path)
13      label_indices = np.arange(0, len(labels))
14      return labels, label_indices, to_categorical(label_indices)
```

資料前處理

- 前處理上，聲音是由一串不固定頻率之波譜組成的而其中每個區段的又可能是由不同頻率的小波譜疊加而成ㄉ而從時域上是無法發現各類頻率的，所以聲音資料通常都需要藉由復立葉轉換將其轉變為頻域進行分析
- 通常是透過librosa套件中的函數進行特徵擷取 https://librosa.github.io/librosa/
- 常用的librosa函數mfcc()
- mfcc :使用於類人聽覺

取得mfcc特徵

- Sample_rate – sr是多少毫秒取一次
- mfcc()將剛剛取得之時域陣列轉換為頻域

```
17  # Handy function to convert wav2mfcc
18  def wav2mfcc(file_path, max_pad_len=11):
19      wave, sr = librosa.load(file_path, mono=True, sr=None)
20      #wave = wave[::3]
21      wave =np.ascontiguousarray(wave[::3])
22      mfcc = librosa.feature.mfcc(wave, sr=16000)
23      pad_width = max_pad_len - mfcc.shape[1]
24      mfcc = np.pad(mfcc, pad_width=((0, 0), (0, pad_width)), mode='constant')
25      return mfcc
```

將每個音檔的 mfcc 特徵存成 npy 檔案

```
28  def save_data_to_array(path=DATA_PATH, max_pad_len=11):
29      labels, _, _ = get_labels(path)
30
31      for label in labels:
32          # Init mfcc vectors
33          mfcc_vectors = []
34
35          wavfiles = [path + label + '/' + wavfile for wavfile in os.listdir(path + '/' + label)]
36          for wavfile in wavfiles:
37              mfcc = wav2mfcc(wavfile, max_pad_len=max_pad_len)
38              mfcc_vectors.append(mfcc)
39          np.save(label + '.npy', mfcc_vectors)
```

將語音資料切成80%訓練集20%測試集

```
42  def get_train_test(split_ratio=0.8, random_state=42):
43      # Get available labels
44      labels, indices, _ = get_labels(DATA_PATH)
45
46      save_data_to_array(DATA_PATH)
47
48      # Getting first arrays
49      X = np.load(labels[0] + '.npy')
50      y = np.zeros(X.shape[0])
51
52      # Append all of the dataset into one single array, same goes for y
53      for i, label in enumerate(labels[1:]):
54          x = np.load(label + '.npy')
55          X = np.vstack((X, x))
56          y = np.append(y, np.full(x.shape[0], fill_value= (i + 1)))
57
58      assert X.shape[0] == len(y)
59
60      return train_test_split(X, y, test_size= (1 - split_ratio), random_state=random_state, shuffle=True)
```

模型構成

- 模型採用CNN網路架設
- CNN --https://brohrer.mcknote.com/zh-Hant/how_machine_learning_works/how_convolutional_neural_networks_work.html
- 捲積層 4層
- 激活函數 選用RELU 解決梯度消失 增加收斂速度
- Dropout(0.1) 正規化

訓練結果

- 執行train.py進行模型訓練 會產生一個ASR.h5的模型權重檔

```python
# 導入函式庫
from preprocess import *
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical

# 載入 data 資料夾的訓練資料，並自動分為『訓練組』及『測試組』
X_train, X_test, y_train, y_test = get_train_test()
X_train = X_train.reshape(X_train.shape[0], 20, 11, 1)
X_test = X_test.reshape(X_test.shape[0], 20, 11, 1)

# 類別變數轉為one-hot encoding
y_train_hot = to_categorical(y_train)
y_test_hot = to_categorical(y_test)
print("X_train.shape=", X_train.shape)
```

訓練結果

- 執行train.py進行模型訓練 會產生一個ASR.h5的模型權重檔

```
18  # 建立簡單的線性執行的模型
19  model = Sequential()
20  # 建立卷積層, filter=32,即 output size, Kernal Size: 2x2, activation function 採用 relu
21  model.add(Conv2D(32, kernel_size=(2, 2), activation='relu', input_shape=(20, 11, 1)))
22  # 建立池化層, 池化大小=2x2, 取最大值
23  model.add(MaxPooling2D(pool_size=(2, 2)))
24  # Dropout層隨機斷開輸入神經元, 用於防止過度擬合, 斷開比例:0.25
25  model.add(Dropout(0.25))
26  # Flatten層把多維的輸入一維化, 常用在從卷積層到全連接層的過渡。
27  model.add(Flatten())
28  # 全連接層: 128個output
29  model.add(Dense(128, activation='relu'))
30  model.add(Dropout(0.25))
31  # Add output layer
32  model.add(Dense(3, activation='softmax'))
33  # 編譯: 選擇損失函數、優化方法及成效衡量方式
34  model.compile(loss=tensorflow.keras.losses.categorical_crossentropy,
35                optimizer=tensorflow.keras.optimizers.Adadelta(),
36                metrics=['accuracy'])
```

訓練結果

- 執行train.py進行模型訓練 會產生一個ASR.h5的模型權重檔

```
38  # 進行訓練, 訓練過程會存在 train_history 變數中
39  model.fit(X_train, y_train_hot, batch_size=100, epochs=200, verbose=1, validation_data=(X_test, y_test_hot))
40
41
42  X_train = X_train.reshape(X_train.shape[0], 20, 11, 1)
43  X_test = X_test.reshape(X_test.shape[0], 20, 11, 1)
44  score = model.evaluate(X_test, y_test_hot, verbose=1)
45
46  # 模型存檔
47  from tensorflow.keras.models import load_model
48  model.save('ASR.h5')  # creates a HDF5 file 'model.h5'
```

訓練結果

- 執行train.py進行模型訓練 會產生一個ASR.h5的模型權重檔

命令提示字元

```
42/42 [==============================] - 0s 7ms/step - loss: 6.2613 - accuracy: 0.4851 - val_loss: 1.2417 - val_accuracy: 0.7004
Epoch 190/200
42/42 [==============================] - 0s 7ms/step - loss: 5.9795 - accuracy: 0.5010 - val_loss: 1.2359 - val_accuracy: 0.7033
Epoch 191/200
42/42 [==============================] - 0s 7ms/step - loss: 5.9365 - accuracy: 0.4973 - val_loss: 1.2303 - val_accuracy: 0.7042
Epoch 192/200
42/42 [==============================] - 0s 6ms/step - loss: 6.1598 - accuracy: 0.5014 - val_loss: 1.2309 - val_accuracy: 0.7042
Epoch 193/200
42/42 [==============================] - 0s 6ms/step - loss: 6.0303 - accuracy: 0.4846 - val_loss: 1.2231 - val_accuracy: 0.7033
Epoch 194/200
42/42 [==============================] - 0s 6ms/step - loss: 5.8888 - accuracy: 0.4925 - val_loss: 1.2213 - val_accuracy: 0.7023
Epoch 195/200
42/42 [==============================] - 0s 6ms/step - loss: 6.0161 - accuracy: 0.4853 - val_loss: 1.2212 - val_accuracy: 0.7071
Epoch 196/200
42/42 [==============================] - 0s 6ms/step - loss: 5.8184 - accuracy: 0.5063 - val_loss: 1.2186 - val_accuracy: 0.7071
Epoch 197/200
42/42 [==============================] - 0s 6ms/step - loss: 5.9035 - accuracy: 0.4976 - val_loss: 1.2141 - val_accuracy: 0.7081
Epoch 198/200
42/42 [==============================] - 0s 6ms/step - loss: 5.7483 - accuracy: 0.4928 - val_loss: 1.2054 - val_accuracy: 0.7052
Epoch 199/200
42/42 [==============================] - 0s 6ms/step - loss: 5.9470 - accuracy: 0.4896 - val_loss: 1.2029 - val_accuracy: 0.7071
Epoch 200/200
42/42 [==============================] - 0s 6ms/step - loss: 5.6944 - accuracy: 0.4867 - val_loss: 1.1940 - val_accuracy: 0.7071
33/33 [==============================] - 0s 1ms/step - loss: 1.1940 - accuracy: 0.7071
labels= (['bed', 'cat', 'happy'], array([0, 1, 2]), array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]], dtype=float32))
predict= 1
```

測試方法

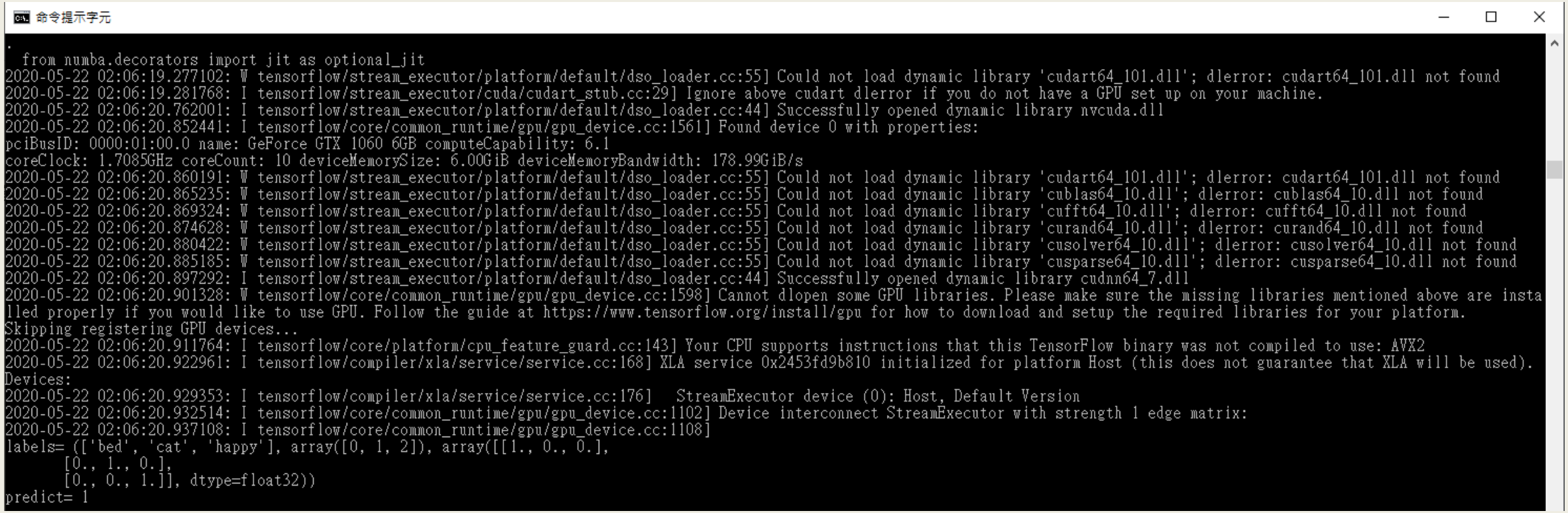
- 執行test.py將訓練後的模型讀入並丟入一個音檔進行預測

```
# 導入函式庫
from preprocess import *
# 模型讀檔
from tensorflow.keras.models import load_model
model = load_model('ASR.h5')  # load a HDF5 file 'model.h5'


# 預測(prediction)
mfcc = wav2mfcc('./data/cat/00b01445_nohash_0.wav')
mfcc_reshaped = mfcc.reshape(1, 20, 11, 1)
print("labels=", get_labels())
print("predict=", np.argmax(model.predict(mfcc_reshaped)))
```

測試方法

- 執行test.py將訓練後的模型讀入並丟入一個音檔進行預測



參考文獻

- https://ithelp.ithome.com.tw/articles/10195763