```

```

```
from google.colab import drive
drive.mount('/content/drive')
```

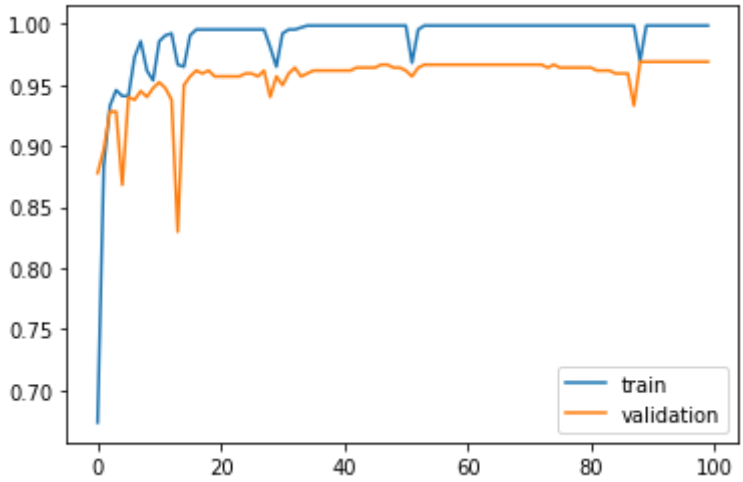
```
cd /content/drive/My Drive/soundClasstion

```

```
!ls
```

```
import os
import keras
import librosa
import numpy as np
import matplotlib.pyplot as plt
from keras import Sequential
from keras.utils import to_categorical
from keras.layers import Dense
from sklearn.model_selection import train_test_split

DATA = 'data.npy'
TARGET = 'target.npy'


# 加载标签
def load_label(label_path):
  label = os.listdir(label_path)
  return label


# 提取 mfcc 参数
def wav2mfcc(path, max_pad_size=11):
  y, sr = librosa.load(path=path, sr=None, mono=1)
  y = y[::3]  #每三个点选用一个
  audio_mac = librosa.feature.mfcc(y=y, sr=16000)
  y_shape = audio_mac.shape[1]
  if y_shape < max_pad_size:
      pad_size = max_pad_size - y_shape
      audio_mac = np.pad(audio_mac, ((0, 0), (0, pad_size)), mode='constant')
  else:
      audio_mac = audio_mac[:, :max_pad_size]
  return audio_mac


# 存储处理过的数据，方便下一次的使用
def save_data_to_array(label_path, max_pad_size=11):
  mfcc_vectors = []
  target = []
  labels = load_label(label_path=label_path)
  for i, label in enumerate(labels):
      path = label_path + '/' + label
      wavfiles = [path + '/' + file for file in os.listdir(path)]
```

```
        for wavfile in wavfiles:
            wav = wav2mfcc(wavfile, max_pad_size=max_pad_size)
            mfcc_vectors.append(wav)
            target.append(i)
    np.save(DATA, mfcc_vectors)
    np.save(TARGET, target)
    # return mfcc_vectors, target


# 获取训练集与测试集
def get_train_test(split_ratio=.6, random_state=42):
    X = np.load(DATA)
    y = np.load(TARGET)
    assert X.shape[0] == y.shape[0]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=(1 - split_ratio), random
                                                        shuffle=True)
    return X_train, X_test, y_train, y_test


def main():

    x_train, x_test, y_train, y_test = get_train_test()
    x_train = x_train.reshape(-1, 220)
    x_test = x_test.reshape(-1, 220)
    y_train_hot = to_categorical(y_train)
    y_test_hot = to_categorical(y_test)
    model = Sequential()
    model.add(Dense(64, activation='relu', input_shape=(220,)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(2, activation='softmax'))

    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.RMSprop(),
                  metrics=['accuracy'])
    history = model.fit(x_train, y_train_hot, batch_size=100, epochs=100, verbose=1,
                        validation_data=(x_test, y_test_hot))
    plot_history(history)
    model.save("classaud.h5")




def plot_history(history):
    plt.plot(history.history['acc'],label='train')
    plt.plot(history.history['val_acc'],label='validation')
    plt.legend()
    plt.show()


if __name__ == "__main__":
    #save_data_to_array("./data/", max_pad_size=11)
    main()

```

```
625/625 [==============================] - 0s 77us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 81/100
625/625 [==============================] - 0s 71us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 82/100
625/625 [==============================] - 0s 67us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 83/100
625/625 [==============================] - 0s 72us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 84/100
625/625 [==============================] - 0s 69us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 85/100
625/625 [==============================] - 0s 66us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 86/100
625/625 [==============================] - 0s 78us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 87/100
625/625 [==============================] - 0s 65us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 88/100
625/625 [==============================] - 0s 67us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 89/100
625/625 [==============================] - 0s 68us/step - loss: 0.2672 - acc: 0.9696 - val_los
Epoch 90/100
625/625 [==============================] - 0s 66us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 91/100
625/625 [==============================] - 0s 71us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 92/100
625/625 [==============================] - 0s 68us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 93/100
625/625 [==============================] - 0s 69us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 94/100
625/625 [==============================] - 0s 68us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 95/100
625/625 [==============================] - 0s 64us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 96/100
625/625 [==============================] - 0s 67us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 97/100
625/625 [==============================] - 0s 73us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 98/100
625/625 [==============================] - 0s 69us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 99/100
625/625 [==============================] - 0s 73us/step - loss: 0.0258 - acc: 0.9984 - val_los
Epoch 100/100
625/625 [==============================] - 0s 77us/step - loss: 0.0258 - acc: 0.9984 - val_los
```



```
import wave
import numpy as np
import os
from keras.models import load_model



def get_wav_mfcc(wav_path):
    f = wave.open(wav_path,'rb')
    params = f.getparams()
    # print("params:",params)
    nchannels, sampwidth, framerate, nframes = params[:4]
    strData = f.readframes(nframes)#读取音频，字符串格式
    waveData = np.fromstring(strData,dtype=np.int16)#将字符串转化为int
    waveData = waveData*1.0/(max(abs(waveData)))#wave幅值归一化
    waveData = np.reshape(waveData,[nframes,nchannels]).T
    f.close()

    ### 对音频数据进行长度大小的切割，保证每一个的长度都是一样的【因为训练文件全部是1秒钟长度，
    data = list(np.array(waveData[0]))
```