گزارش پروژه نهایی درس مبانی هوش محاسباتی

سیستم Speech recognition برای تشخیص کلمات کلیدی

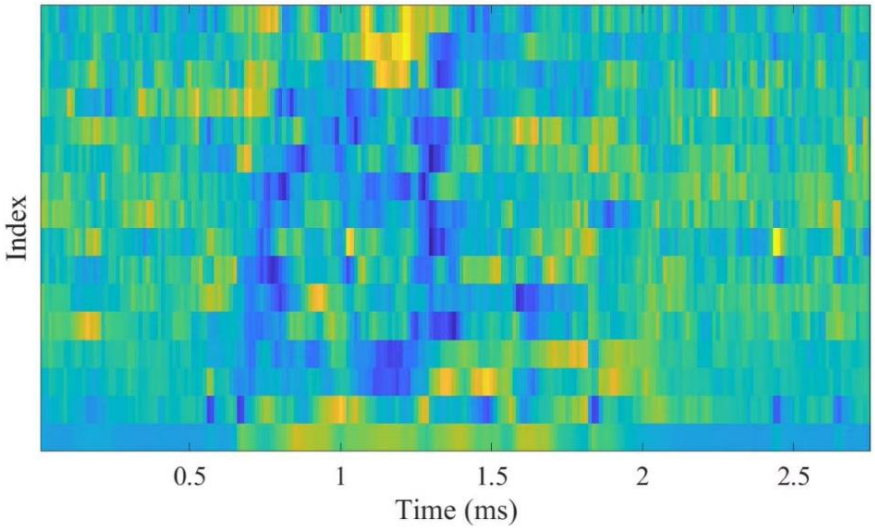
محمدرضا افشاری

220797033

شرح پروژه:

هدف این پروژه ساخت یک سیستم برای تشخیص چند کلمه کلیدی از روی صوت ورودی به آن میباشد. این کلمات عبارتند از: down, go, left, no, off, on, right, stop, up, yes که میتوانند به عنوان کلمات دستوری برای یک دستیار صوتی استفاده شوند.

در این پروژه از مدل CNN (Convolutional neural network) برای ساخت یک multi-class classifier استفاده شده است. دلیل استفاده از این مدل هم نوع فیچر های ورودی مسئله (که میتوان به آنها به فرم یک تصویر نگاه کرد) میباشد. برای این مدل فیچر MFCC (Mel-Frequency Cepstral Coefficient) استفاده شده است که برای این دست مسائل مناسب هستند. این فیچر ها را میتوان از سیگنال های صوت استخراج کرد. در مسئله ما به این دلیل که اندازه ی ورودی شبکه های cnn باید همگی به یک اندازه باشند، تمامی فایل های صوتی را به 1 ثانیه تبدیل میکنیم. این 1 ثانیه سیگنال که 22050 سمپل را شامل میشود، به قسمت های پشت سر هم با 512 سمپل فاصله تقسیم میشوند که در نهایت هر صوت به 44 قسمت تقسیم میشود. سپس ضرایب mfcc برای هر یک از این قسمت ها استخراج میشود. برای این مسئله از 13 ضریب اول که مقداری رایج برای تعداد ضرایب است و بیشتر اطلاعات سیگنال صوتی را نیز در خود دارد استفاده شده است. در نهایت فایل های صوتی که در ابتدا یک سیگنال به ابعاد (1, 22050) بودند به ماتریسی به ابعاد (44,13) تبدیل میشود که در هر خانه از این ماتریس یک عدد وجود دارد و میتوان به آن مانند یک تصویر نگاه کرد.



- **preprocess.py**

در این کد دیتا ها به فرمت قابل استفاده برای مدل تبدیل میشوند. در این پروژه از دیتاست Google Speech commands dataset 2017 استفاده شده که شامل فایل های صوتی یک ثانیه ای از گفتن کلمات مختلف میباشد که زیرمجموعه ای از این کلمات به عنوان دیتای این پروژه استفاده شده است.

```
import librosa
import os
import json
import numpy as np

data = {
        "map": ['down' , 'off' , 'on' , 'no' , 'yes' , 'stop' , 'up' , 'right' , 'left' , 'go'],
        "labels": [],     # Label of each audio file
        "MFCC": [],       # MFCC of each audio file
        "file": []        # path of each audio file
        }

for i in range(len(data['map']) ):
    files = os.listdir('./raw data/' + data['map'][i])
    for f in files:
        filename = './raw data/' + data['map'][i] + '/' + f
        print(filename)

        signal , sr = librosa.load(filename)

        if len(signal) >= 22050:  # only add audio files that are longer than 1s
            signal = signal[:22050]  # we only want 1s
            mfcc = librosa.feature.mfcc(signal, sr=22050, n_mfcc=13, hop_length=512, n_fft=2048)

            data['labels'].append(i) # index of label in map
            data['MFCC'].append(mfcc.T.tolist())
            data['file'].append(filename)


with open('data.json','w') as f:
    json.dump(data,f,indent=4)
```

دیتاستی که برای train کردن مدل استفاده میشود در فایل data.json ذخیره میشود. نحوه ایجاد این دیتاست در کد بالا آمده است. ابتدا یک دیکشنری که اطلاعات معنای هر لیبل، آرایه ای از خود لیبل ها، ماتریس های mfcc برای هر فایل صوتی و نام فایل صوتی در آن ذخیره میشود را ایجاد میکنیم. این دیکشنری در نهایت به فایل json تبدیل میشود.

با بررسی تک به تک فایل ها و استخراج mfcc هر فایل صوتی با استفاده از کتابخانه ی librosa، فیچر های تمامی فایل های صوتی را بدست آورده و در دیکشنری data ذخیره میکنیم.

در این کد از دو تابع کتابخانه ی librosa استفاده شده است. این کتابخانه یک کتابخانه ی متن باز برای کار با فایل های صوتی میباشد. از تابع load برای خواندن فایل های صوتی استفاده شده که در خروجی یک آرایه نامپای به اندازه ی تعداد سمپل ها بازمیگرداند. از تابع mfcc هم برای استخراج mfcc ها از سیگنال صوتی استفاده شده.

- model.py

در این کد مدل cnn ایجاد و train شده است. برای اینکار از کتابخانه ی keras استفاده شده است.

```
# prepairing the datasets
with open('data.json') as f:
    data = json.load(f)
X = data['MFCC']
y = data['labels']
X = np.array(X)
y = np.array(y)
X_train, X_test, Y_train, Y_test = train_test_split(X,y,test_size=0.1)
X_train, X_val, Y_train,Y_val = train_test_split(X_train,Y_train,test_size=0.1)
# adding number of channels to our data ( so far the data is: [# of segments,# coefs])
X_train = X_train[...,np.newaxis]
X_test = X_test[...,np.newaxis]
X_val = X_val[...,np.newaxis]
```

در ابتدای کار دیتاست اصلی به سه قسمت تبدیل شده است. یک دیتاست برای train یکی برای validation و دیگری برای test.

دیتاست test یک دهم کل دیتاست اصلی را شامل میشود و دیتاست 0.81 train دیتای اصلی و دیتاست 0.09 validation دیتای اصلی را شامل میشود.

```
#****************************************** build the model ***************************************************
input_shape = (X_train.shape[1], X_train.shape[2], X_train.shape[3])  # ( # of segments, # of coefs, # of channels)
learning_rate = 0.0001

model = keras.Sequential()

#conv layer 1
model.add(keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=input_shape,kernel_regularizer=keras.regularizer

model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPool2D( (3,3), strides=(2,2) , padding='same' ))

#conv layer 2
model.add(keras.layers.Conv2D(32, (3,3), activation='relu', kernel_regularizer=keras.regularizers.l2(0.001)))

model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPool2D( (3,3), strides=(2,2) , padding='same' ))


#conv layer 3
model.add(keras.layers.Conv2D(32, (2,2), activation='relu', kernel_regularizer=keras.regularizers.l2(0.001)))

model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPool2D( (2,2), strides=(2,2) , padding='same' ))

# dense layer
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dropout(0.3))

#softmax
model.add(keras.layers.Dense(10,activation='softmax'))

#compile
optimiser = keras.optimizers.Adam(learning_rate)
model.compile(optimizer = optimiser, loss='sparse_categorical_crossentropy', metrics=['accuracy'])

model.summary()

#*************************************************************************************************************
```

در این بخش مدل ایجاد میشود. برای اینکار از مدل Sequential کتابخانه ی keras استفاده میشود. اطلاعات لایه های مختلف این شبکه عصبی به صورت زیر میباشد:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_30 (Conv2D)           (None, 42, 11, 64)        640
_________________________________________________________________
batch_normalization_30 (Batc (None, 42, 11, 64)        256
_________________________________________________________________
max_pooling2d_30 (MaxPooling (None, 21, 6, 64)         0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 19, 4, 32)         18464
_________________________________________________________________
batch_normalization_31 (Batc (None, 19, 4, 32)         128
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 10, 2, 32)         0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 9, 1, 32)          4128
_________________________________________________________________
batch_normalization_32 (Batc (None, 9, 1, 32)          128
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 5, 1, 32)          0
_________________________________________________________________
flatten_10 (Flatten)         (None, 160)               0
_________________________________________________________________
dense_20 (Dense)             (None, 64)                10304
_________________________________________________________________
dropout_10 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_21 (Dense)             (None, 10)                650
=================================================================
Total params: 34,698
Trainable params: 34,442
Non-trainable params: 256
```

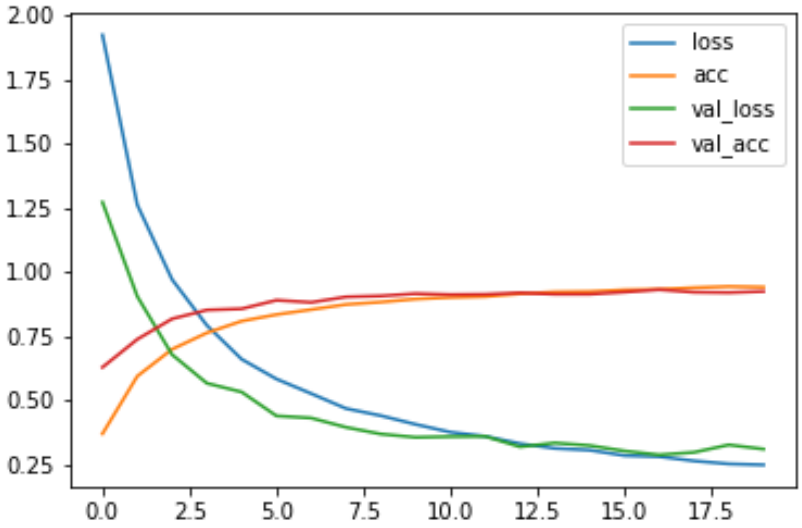
```
Epoch 40/40
17262/17262 [==============================] - 82s 5ms/sample - loss: 0.2247 - acc: 0.9463 - val_loss: 0.3383 -
val_acc: 0.9155
2132/2132 [==============================] - 1s 558us/sample - loss: 0.3338 - acc: 0.9212
test error: 0.33375176211459107
test accuracy: 0.92120075
```

یک مدل دیگر با معماری متفاوت:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 42, 11, 128)       1280
_________________________________________________________________
batch_normalization (BatchNo (None, 42, 11, 128)       512
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 21, 6, 128)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 19, 4, 64)         73792
_________________________________________________________________
batch_normalization_1 (Batch (None, 19, 4, 64)         256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 2, 64)         0
_________________________________________________________________
flatten (Flatten)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 64)                81984
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 158,474
Trainable params: 158,090
Non-trainable params: 384
```

```
test error: 0.33953416825570937
test accuracy: 0.912758
```

ارزیابی عملکرد مدل:



- **Predict.py**

در این کد فایل صوتی ورودی دریافت میشود و کلمه ای که در آن گفته شده است توسط مدل پیشبینی میشود.

```
import tensorflow.keras as keras
import numpy as np
import librosa
import matplotlib.pyplot as plt


mapping = ['down' , 'off' , 'on' , 'no' , 'yes' , 'stop' , 'up' , 'right' , 'left' , 'go']

#load the model
model = keras.models.load_model('model.h5')

def predict(filepath):
    signal, sr = librosa.load(filepath)
    signal = signal[:22050]
    mfcc = librosa.feature.mfcc(y=signal,sr=22050,n_mfcc=13,n_fft=2048,hop_length=512).T
    plt.Figure(figsize=(10,20),dpi=400)
    plt.imshow(librosa.power_to_db(mfcc.T**2))

    #  (# of samples, # of segments, # of coefs, # of channels)
    mfcc = mfcc[np.newaxis, ... ,np.newaxis]
    preds = model.predict(mfcc)
    idx = np.argmax(preds)
    print('predicted keyword is ',end='')
    print(mapping[idx])



predict('test/right.wav')
```

ابتدا فیچر های mfcc برای صوت ورودی محاسبه میشود و این فیچر ها به فرم قابل استفاده برای مدل تبدیل میشوند و در نهایت به مدل داده شده و نتیجه ی پیشبینی مدل چاپ میشود.