

پروژه ی پایانی درس گفتار پردازى رقمى

نام : حسين سيم چى، على يزدانى

۹۸۴۴۳۱۱۹ (سيم چى)، ۹۸۴۴۳۲۵۷ (يزدانى)

استاد : آقاى دكتور ياسر شكفته

۱۳۹۹/۱۱/۱۴

توضیحات مربوط به فایل گزارش و پروژه :

پروژه ی نوشته شده شامل ۳ قسمت است

۱. قسمت اول شامل مقدمه، بررسی توابع استخراج ویژگی و کاربرد هریک از آن ها جهت تحلیل احساسات و همچنین بررسی خروجی اولین کد نوشته شده جهت دسته بندی احساسات موجود در یک جمله می باشد (در این کد قسمتی نیز به عنوان Prediction در نظر گرفته شده که قادر خواهد بود برای یک فایل صوتی دلخواه، احساس مربوط در آن را تشخیص دهد)

۲. قسمت دوم شامل دومین کد نوشته شده، توضیح کامل آن و آموزش کامل بر روی دیتاست “ShEMO” . (دارای تکنیک های مختلف یادگیری ماشین و عمیق و همچنین استفاده ی بیشتری ویژگی نسبت به کد اولی است)

۳. قسمت سوم، شامل توضیح نحوه ی اجرا شدن اولین کد بر روی دیتاست های “Berlin” و “ShEMO” به فرمت مقاله

کد اول و دوم به واسطه ی تعدد ویژگی ها و همچنین به لحاظ سادگی و دقت تفاوت هایی باهم دارند. کد دوم در نظر گرفته شده ۴۶ ویژگی را استخراج می کند درحالی که این عدد در کد اول کمتر بوده و ۳۶ ویژگی خواهد بود.

قسمت اول

کد اول

در این قسمت به بررسی توابع نوشته شده یا کلاس های مورد نیاز و همچنین مدل نوشته شده جهت دسته بندی احساسات می پردازیم. همچنین در قسمت دوم، سعی شده است مراحل انجام کد و نحوه ی اجرای آن را بررسی کنیم.

در این پروژه هدف پیاده سازی مقاله ی تشخیص احساسات ارائه شده می باشد به طوری که جهت بررسی بیشتر سعی شده است بر روی دو دیتاست دیگر از قبیل “Berlin” و “ShEMO” خروجی گرفته شود.

***** تصاویر بدست آمده از خروجی پروژه و مدل نوشته در طول توضیح گزارش قرار داده شده است.**

مقدمه

تشخیص احساسات یکی از چالشی ترین موضوعات مورد بررسی متخصصان هوش مصنوعی بوده است به طوری که در بسیاری از کاربردها مانند بازی های کامپیوتری، رباتها و ارتباط انسان ها، احساسات نقش بسیار مهمی دارند. یکی از راه های تشخیص احساس، استفاده از سیگنال گفتار و ارزیابی ویژگی هایی است که به بهتر شناسایی شدن احساس موجود در گفتار کمک می کند. با گسترش استفاده از روش های یادگیری عمیق در سال های گذشته، در این پروژه سعی شده است یکی از پراستنادترین روش های چندسال گذشته جهت تشخیص احساسات را استفاده کنیم. روش مورد نظر که با نام Bidirectional LSTM with Local Attention شناخته شده است در ابتدا هر جمله را به فریم هایی مجزا تقسیم میکند و سپس سعی میکند با توجه به واکنش بودن، بیواک بودن و یا ساکت بودن هر فریم وزنی را به آن ها اختصاص داده تا فریمی که دارای کلمات با احساس بیشتری است وزن بیشتری را داشته باشد و لذا در نهایت تاثیر بیشتری نیز بر روی به روزرسانی وزن ها داشته باشد. در این پروژه هدف پیاده سازی مقاله گفته شده و سپس ارزیابی بر روی چند دیتاست معروف از قبیل "Berlin" و "ShEMO" است. تمامی نتایج و قسمت های مختلف انجام پروژه به صورت گزارش در ادامه آمده شده است. در این قسمت سعی می کنیم توابع و مدل نوشته شده را با دقت بررسی و تحلیل کنیم و در قسمت دوم نیز مرحله ای که کد طی می کند تا خروجی را نشان دهد، بررسی می کنیم.

*** تمامی فایل ها، تصاویر و مدل های نوشته شده در فایل گزارش ارسال و تقدیم می گردد.

توابع نوشته شده جهت استخراج ویژگی (گام اول)

کلاس دیتاست

جهت برچسب گذاری داده های درون دیتاست از آن استفاده شده است.

دو پارامتر مهم که در زمان ساخت شی از کلاس باید به آن داده شوند، پارامترهای اسم دیتاست و مسیری است که دیتاست در آن قرار داده شده است.

*** از کتابخانه ی Librosa جهت خواندن فایل های صوتی و بدست آوردن فرکانس نمونه برداری و سیگنال در طول پروژه استفاده شده است.

هر فایل صوتی که در دیتاست قرار داده شده دارای یک نام است که خود دارای برچسب موردنظر برای داده است. به عنوان مثال برای دیتاست “Berlin” حرف پنجم و در دیتاست “ShEMO” حرف چهارم دارای برچسب مخصوص به داده است که با برخی از واژگان انگلیسی این کلاس ها مشخص شده اند.

درنهایت خروجی این کلاس شامل ۳ جز است

۱. کلاس استخراج شده برای هر داده
۲. سیگنال مربوط به فایل صوتی مورد نظر
۳. فرکانس نمونه برداری شده ی سیگنال

تابع استخراج ویژگی Chroma (stChromaFeaturesInit)

از این ویژگی برای بررسی فایل های صوتی مربوط به موسیقی استفاده می شود و به طور کلی برای کلاس بندی Pitch ها مورد استفاده قرار می گیرد، به این صورت که فرض می کنیم Pitch ها می توانند دارای ۱۲ نوع کلاس باشند. از این ویژگی ها بر اساس تحقیقات انجام شده برای شناسایی بلندی یا آرامی صدا استفاده می شود که در برخی از کاربردهای شناسایی احساسات مثل شناسایی عصبانی بودن مخاطب مورد استفاده قرار می گیرد.

تابع محاسبه ی فیلتربانک برای بدست آوردن ویژگی MFCC (mfccInitFilterBanks)

برای محاسبه ی ویژگی MFCC در ابتدا نیاز داریم تا سیگنال خود را به فضای فرکانس برده و سپس با اعمال تعداد فیلتربانک های مشخص بر روی آن و بقیه ی مراحل، این مقدار را بدست آوریم. (در این پروژه تعداد فیلتر ها برابر ۴۰ درنظر گرفته شده است)

مراحل استفاده شده جهت اعمال فیلتربانک

۱. مقدار دهی های لازم (تعداد فیلتر های برابر ۴۰)، فیلترهای اعمال شده مثلثی هستند.
۲. بدست آوردن فرکانس نقاطی که در داخل یک فیلتر قرار می گیرند
۳. محاسبه ی ضرایب فیلتربانک

درنهایت سیگنال فیلتر شده و ضرایب را برمی گردانیم.

*** از توابع و فرمول های موجود در اینترنت جهت استخراج ویژگی هایی که نام می بریم استفاده شده است.

مراحل بدست آوردن ویژگی MFCC



همانطور که فرآیند فوق مشخص است، برای بدست آوردن ویژگی MFCC باید ۴ مرحله ی فوق را طی کنیم که در این پروژه هر ۴ مرحله به صورت توابع نوشته شده است

تابع استخراج ویژگی Zero Crossing Rate (stZCR)

از ویژگی ZCR برای **Voice activity detection** استفاده می شود که منجر می شود تشخیص دهیم که آیا در فریم موردنظر گوینده حضور دارد یا خیر. با استفاده از این ویژگی می توانیم فریم های ساکت را از بقیه ی فریم ها جدا کنیم

تابع استخراج ویژگی انرژی (stEnergy)

از ویژگی انرژی سیگنال هم می توان بر دسته بندی فریم های ساکت، بی واک و واکنار استفاده نمود. چراکه می دانیم فریم هایی که دارای انرژی زیادی هستند، کلمات و جملات بیشتری را شامل شده اند.

تابع استخراج ویژگی آنترپی انرژی (stEnergyEntropy)

از ویژگی آنترپی انرژی نیز برای **Voice activity detection** استفاده می شود. در حالت کلی از ویژگی آنترپی (stSpectralEntropy) جهت تشخیص تعداد پیک های یک سیگنال در یک فریم استفاده می شود.

تابع استخراج ویژگی مرکز طیف سیگنال (stSpectralCentroidAndSpread)

مرکز طیف جایی است که در فضای Spectral دارای شدت روشنایی زیادی است و از این ویژگی می توانیم استفاده کنیم تا مقدار Spread را بدست آوریم. از مقدار Spread برای مشخص کردن نویزی بودن یا نبودن سیگنال می توان استفاده کرد. یعنی اگر سیگنالی دارای Spread زیادی باشد بدین معنا است که نویزی است.

تابع استخراج ویژگی Flux (stSpectralFlux)

از ویژگی Flux می توان برای محاسبه ی میزان سرعت تغییر سیگنال استفاده نمود. برای این منظور باید فریم فعلی را با قبلی مقایسه کنیم. از کاربردهای این ویژگی می توان به تشخیص شروع یک جمله یا کلمه و میزان صدای سیگنال اشاره کرد.

تابع استخراج ویژگی Rolloff (stSpectralRollOff)

از این ویژگی نیز در برخی موارد برای تشخیص صداها ی نویزی استفاده می شود. به این صورت سیگنالی که از مقدار Rolloff پایین تر باشد به عنوان صدای نویزی تشخیص داده می شود.

تابع محاسبه ی ماتریس DCT برای بدست آوردن MFCC

برای بدست آوردن ویژگی MFCC مطابق آنچه در کلاس توضیح داده شد نیازی داریم تا ماتریس DCT را بدست آوریم که توابع آن ها بدست آورده شده است.

تابع استخراج ویژگی (stFeatureExtraction)

کار این تابع استخراج تمامی ویژگی های نام برده شده و ذخیره در فایلی جداگانه است تا در زمان آموزش از آن استفاده کنیم.

مدل نوشته شده جهت آموزش و دسته بندی احساسات موجود در جملات (گام دوم)

۱. تابع نوشته شده جهت تقسیم دیتاست به دو قسمت آموزش و تست و همچنین مشخص کردن پارامترهایی جهت بدست آوردن مکانیزم Attention (get_data Function)
۲. تابع Create_model که دربردارنده ی هسته ی اصلی مدل، یعنی جایی که مدل را تعریف می کنیم تا در ادامه با استفاده از داده های آموزش و تست بر روی آن خروجی بگیریم.

تابع get_data

در مراحل قبل با استفاده از کلاس دیتاست و توابع استخراج ویژگی که در مورد آن ها صحبت کردیم و همچنین طبق روالی که در قسمت دوم گزارش به آن خواهیم پرداخت، داده های استخراج شده و برگه ای که مربوط به استخراج ویژگی های متناظر می باشد و اطلاعات هر داده که نشان دهنده ی تعداد ویژگی های استخراج شده از آن داده میباشد را باید Load کنیم. سپس باید با استفاده از تابع train_test_split داده های موجود را به دو بخش آموزش و تست تقسیم کنیم.

برای داده های آموزش و تست، باید مقداری را به بردارهای آن ها الحاق کنیم. و در نهایت با ضرب نظیر به نظیر ورودی در خروجی و عبور از تابع فعال ساز Softmax، مکانیزم Attention را لحاظ می کنیم. بردارهایی که

باید به داده اضافه کنیم را در پروژه u_train و u_test نامیدیم که بردارهایی به اندازه ی بردارهای train و test هستند و مقادیر آن ها یک تقسیم بر ۲۵۶ در نظر گرفته شده است در نتیجه بردار u مانند زیر خواهد بود:

$$\left[\frac{1}{256}, \frac{1}{256}, \dots, \frac{1}{256}\right]$$

ساختار شبکه در زیر آمده است که توضیحات آن نوشته شده است

1. Input attention >>> دو مرحله ی اول و دوم صرفاً برای مقدار دهی وزن ها است و تاثیری بر لایه های بعدی ندارد
2. Dense layer 1
3. Input features
4. Masking layer >>> از این لایه استفاده می شود تا بگوییم که توالی زمانی بین داده های ورودی نداریم
5. Dense layer 2
6. Dropout layer 1
7. Dense layer 3
8. Dropout layer 2
9. Bidirectional LSTM layer
10. Dot (num2 and num9) >>> خروجی لایه ی ۹ را در بردار وزن های اولیه ضرب می کنیم
11. Activation Softmax >>> مقادیر به صورت احتمالاتی بین صفر و یک قرار میگیرند
12. Dot (num11 and num9) >>> با ضرب لایه ی ۹ در ۱۱، خروجی به صورت وزن دارد خواهد بود
13. Output layer (Dense layer 4)

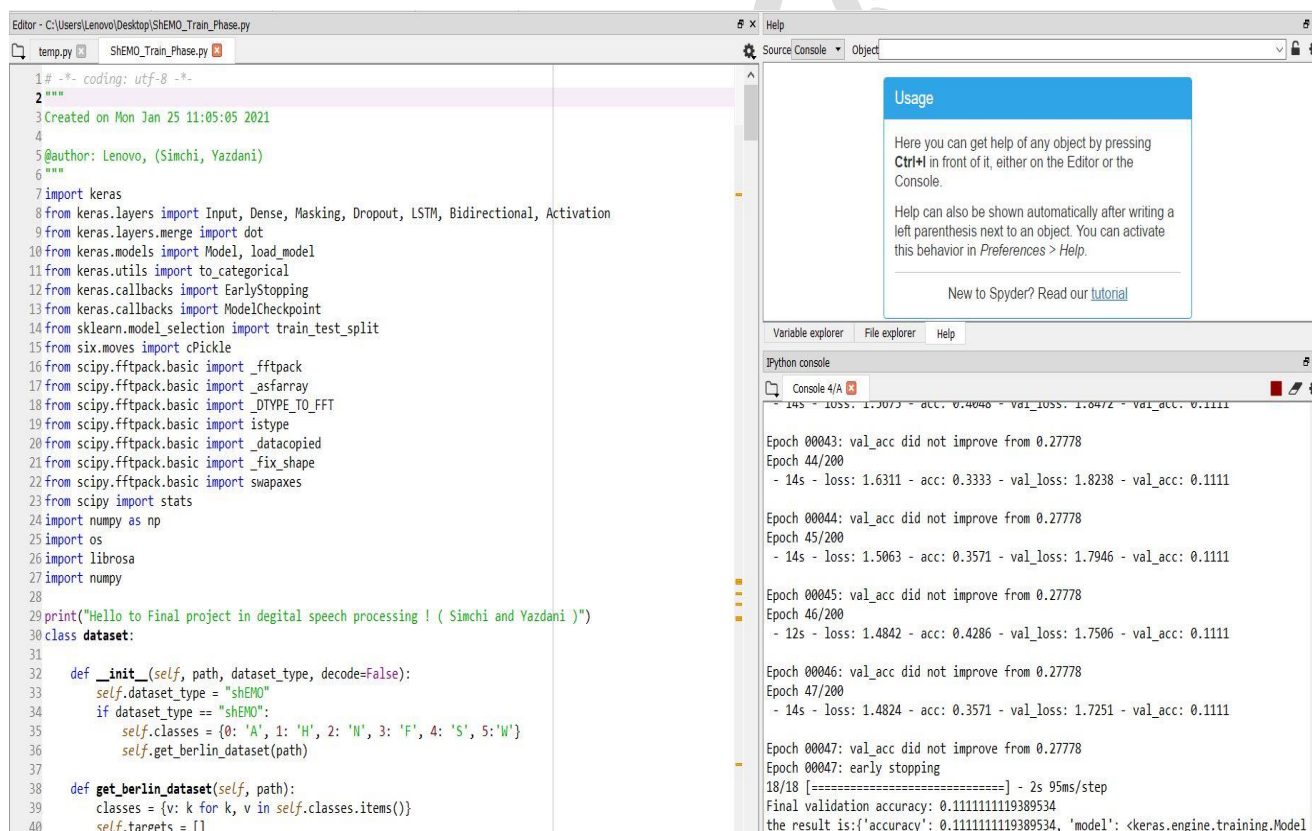
مدل نهایی به صورت زیر خواهد بود

```
model = Model(inputs=[input_attention, input_feature], outputs=output)
```


نتایج بدست آمده (اولین کد) بر روی دیتاست “ShEMO” و “Berlin”

یکی از مشکلات موجود برای آموزش و سپس تست در شبکه های عمیق برای اهداف پردازش سیگنال نیاز مبرم به حافظه ی زیاد جهت پردازش می باشد. درنتیجه به دلیل مشکلات سخت افزاری قادر به گرفتن خروجی بر روی کل دیتاست نبوده ایم و خروجی را بر روی تنها ۶۰ داده ی صوتی گرفته ایم. فایل دیتاست استفاده شده دارای صدای هر دو جنس مرد و زن میباشد که جهت افزایش دقت و پیش بینی، از هر دو صدا استفاده شده است.

۱) اگر داده ها را به صورت **۵ کلاسه** در نظر بگیریم یعنی فرض کنیم فایل های صوتی موجود در دیتاست از ۵ کلاس احساس مختلف تشکیل شده اند درنتیجه به دلیل کم بودن حجم داده های آموزش، دقت بسیار پایین خواهد بود. (همانطور که از فایل خروجی قابل مشاهده است دقت داده های Validation بین ۱۵ درصد تا ۳۰ درصد متغیر خواهد بود درحالی که دقت بر روی داده های آموزش حتی به ۵۰ درصد نیز می رسد. دلیل آن هم بخاطر کم بودن داده های آموزش می باشد که خاصیت تعمیم پذیری مدل را بشدت کاهش داده است و به اصطلاح Overfit بر روی داده های آموزش رخ داده است)



```
1# -*- coding: utf-8 -*-
2"""
3Created on Mon Jan 25 11:05:05 2021
4
5@author: Lenovo, (Simchi, Yazdani)
6"""
7import keras
8from keras.layers import Input, Dense, Masking, Dropout, LSTM, Bidirectional, Activation
9from keras.layers.merge import dot
10from keras.models import Model, load_model
11from keras.utils import to_categorical
12from keras.callbacks import EarlyStopping
13from keras.callbacks import ModelCheckpoint
14from sklearn.model_selection import train_test_split
15from six.moves import cPickle
16from scipy.fftpack.basic import fftpack
17from scipy.fftpack.basic import _asfarray
18from scipy.fftpack.basic import _DTYPE_TO_FFT
19from scipy.fftpack.basic import _istype
20from scipy.fftpack.basic import _datacopied
21from scipy.fftpack.basic import _fix_shape
22from scipy.fftpack.basic import swapaxes
23from scipy import stats
24import numpy as np
25import os
26import librosa
27import numpy
28
29print("Hello to Final project in digital speech processing ! ( Simchi and Yazdani )")
30class dataset:
31
32    def __init__(self, path, dataset_type, decode=False):
33        self.dataset_type = "shEMO"
34        if dataset_type == "shEMO":
35            self.classes = {0: 'A', 1: 'H', 2: 'N', 3: 'F', 4: 'S', 5: 'W'}
36            self.get_berlin_dataset(path)
37
38    def get_berlin_dataset(self, path):
39        classes = {v: k for k, v in self.classes.items()}
40        self.targets = []

```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in **Preferences > Help**.

New to Spyder? Read our [tutorial](#)

Variable explorer File explorer Help

Python console

Console 4/A

14s - loss: 1.3073 - acc: 0.4046 - val_loss: 1.8472 - val_acc: 0.1111

Epoch 00043: val_acc did not improve from 0.27778

Epoch 44/200

- 14s - loss: 1.6311 - acc: 0.3333 - val_loss: 1.8238 - val_acc: 0.1111

Epoch 00044: val_acc did not improve from 0.27778

Epoch 45/200

- 14s - loss: 1.5063 - acc: 0.3571 - val_loss: 1.7946 - val_acc: 0.1111

Epoch 00045: val_acc did not improve from 0.27778

Epoch 46/200

- 12s - loss: 1.4842 - acc: 0.4286 - val_loss: 1.7506 - val_acc: 0.1111

Epoch 00046: val_acc did not improve from 0.27778

Epoch 47/200

- 14s - loss: 1.4824 - acc: 0.3571 - val_loss: 1.7251 - val_acc: 0.1111

Epoch 00047: val_acc did not improve from 0.27778

Epoch 00047: early stopping

18/18 [=====] - 2s 95ms/step

Final validation accuracy: 0.1111111119389534

the result is: {'accuracy': 0.1111111119389534, 'model': <keras.engine.training.Model

۲) اگر به صورت **۲ کلاس** داده های خود را جمع آوری کنیم یعنی داده هایی را استخراج کنیم که تنها دو کلاس در بین آن ها وجود داشته باشد. در این صورت خواهیم دید که دقت داده های آموزش زمان آموزش بین ۵۵ تا ۷۰ و دقت داده های Validation نیز بین ۶۰ تا ۷۰ خواهد بود. به وضوح می توان دریافت زمانی که به دلایل مسایل سخت افزاری یا نرم افزاری نمی توانیم بر روی حجم زیادی از داده ها مدل را آموزش دهیم، هرچه تعداد کلاس ها را افزایش دهیم دقت نیز کاهش خواهد یافت که در شکل زیر خروجی حالت بیان شده را مشاهده می کنید

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Jan 25 11:05:05 2021
4
5 @author: Lenovo, (Simchi, Yazdani)
6 """
7 import keras
8 from keras.layers import Input, Dense, Masking, Dropout, LSTM, Bidirectional, Activation
9 from keras.layers.merge import dot
10 from keras.models import Model, load_model
11 from keras.utils import to_categorical
12 from keras.callbacks import EarlyStopping
13 from keras.callbacks import ModelCheckpoint
14 from sklearn.model_selection import train_test_split
15 from six.moves import cPickle
16 from scipy.fftpack.basic import _fftpack
17 from scipy.fftpack.basic import _asfarray
18 from scipy.fftpack.basic import _DTYPE_TO_FFT
19 from scipy.fftpack.basic import _istype
20 from scipy.fftpack.basic import _datacopied
21 from scipy.fftpack.basic import _fix_shape
22 from scipy.fftpack.basic import _swapaxes
23 from scipy import stats
24 import numpy as np
25 import os
26 import librosa
27 import numpy
28
29 print("Hello to Final project in digital speech processing ! ( Simchi and Yazdani )")
30 class dataset:
31
32     def __init__(self, path, dataset_type, decode=False):
33         self.dataset_type = "shEMO"
34         if dataset_type == "shEMO":
35             self.classes = {0: 'A', 1: 'S'}
36             self.get_berlin_dataset(path)
37
38     def get_berlin_dataset(self, path):
39         classes = {v: k for k, v in self.classes.items()}
40         self.targets = []

```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in **Preferences > Help**.

New to Spyder? Read our [tutorial](#)

Variable explorer File explorer Help

Python console

Console 4/A

```

Epoch 00018: val_acc did not improve from 0.66667
Epoch 19/200
- 8s - loss: 0.6510 - acc: 0.5714 - val_loss: 0.7447 - val_acc: 0.6111

Epoch 00019: val_acc did not improve from 0.66667
Epoch 20/200
- 9s - loss: 0.6293 - acc: 0.6905 - val_loss: 0.7534 - val_acc: 0.6667

Epoch 00020: val_acc did not improve from 0.66667
Epoch 21/200
- 8s - loss: 0.6409 - acc: 0.7143 - val_loss: 0.7648 - val_acc: 0.6111

Epoch 00021: val_acc did not improve from 0.66667
Epoch 22/200
- 9s - loss: 0.6958 - acc: 0.6190 - val_loss: 0.7866 - val_acc: 0.6111

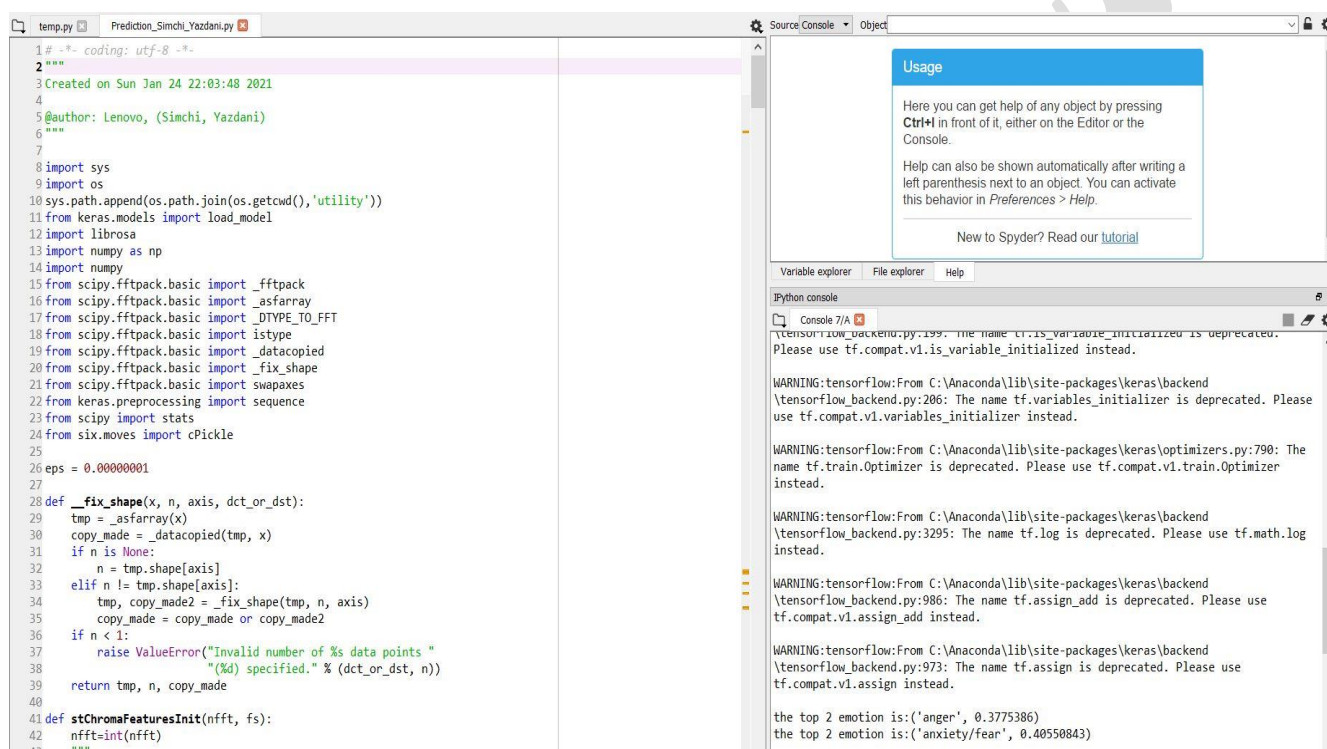
Epoch 00022: val_acc did not improve from 0.66667
Epoch 00022: early stopping
18/18 [=====] - 1s 49ms/step
Final validation accuracy: 0.6111111044883728
the result is: {'accuracy': 0.6111111044883728, 'model': <keras.engine.training.Model
object at 0x000001E33F3D0608>}

```

نتیجه گیری : زمانی که تعداد داده های درون دیتاست کم باشد، هرچه تعداد کلاس ها را افزایش دهیم دقت کاهش خواهد یافت. در نتیجه زمانی که تعداد کلاس های ما زیاد باشد (مانند دیتاست های موجود جهت پردازش سیگنال و صدا) باید حجم دیتاست را نیز زیاد در نظر بگیریم. به همین دلیل برای ارزیابی دیتاست “Berlin”، باید بر روی یک کامپیوتر قوی مدل را آموزش دهیم و سپس از مدل ذخیره شده استفاده کنیم تا عملیات پیش بینی را انجام دهیم. حال باتوجه به توضیحات داده شده، مدل از پیش آموزش دیده شده ی موجود بر روی دیتاست “Berlin” را برای پیش بینی بر داده های تست دیتاست “Berlin” استفاده خواهیم نمود.

پیش بینی نوع احساس موجود برای داده های جدید توسط مدل های آموزش دیده شده (کد اول)

برای پیش بینی، از مدل از پیش آموزش دیده بر روی دیتاست “Berlin” استفاده می کنیم. برای پیش بینی احساس موجود کافی است در کنار توابع استخراج ویژگی بیان شده در قسمت های قبل، تابعی تعریف کنیم تا ماتریس Confusion را محاسبه کند. این ماتریس به ما کمک می کند تا در نهایت بتوانیم احساس با بالاترین درصد ممکن را بدست آورده و به عنوان احساس موجود به آن جمله نسبت دهیم.



```
1 #-*- coding: utf-8 -*-
2 """
3 Created on Sun Jan 24 22:03:48 2021
4
5 @author: Lenovo, (Simchi, Yazdani)
6 """
7
8 import sys
9 import os
10 sys.path.append(os.path.join(os.getcwd(), 'utility'))
11 from keras.models import load_model
12 import librosa
13 import numpy as np
14 import numpy
15 from scipy.fftpack.basic import _fftpack
16 from scipy.fftpack.basic import _asfarray
17 from scipy.fftpack.basic import _DTYPE_TO_FFT
18 from scipy.fftpack.basic import _istype
19 from scipy.fftpack.basic import _datacopied
20 from scipy.fftpack.basic import _fix_shape
21 from scipy.fftpack.basic import _swapaxes
22 from keras.preprocessing import sequence
23 from scipy import stats
24 from six.moves import cPickle
25
26 eps = 0.00000001
27
28 def _fix_shape(x, n, axis, dct_or_dst):
29     tmp = _asfarray(x)
30     copy_made = _datacopied(tmp, x)
31     if n is None:
32         n = tmp.shape[axis]
33     elif n != tmp.shape[axis]:
34         tmp, copy_made2 = _fix_shape(tmp, n, axis)
35         copy_made = copy_made or copy_made2
36     if n < 1:
37         raise ValueError("Invalid number of %s data points "
38                            "%(n)s specified." % (dct_or_dst, n))
39     return tmp, n, copy_made
40
41 def stChromaFeaturesInit(nfft, fs):
42     nfft=int(nfft)
43     """
```

Usage

Here you can get help of any object by pressing **Ctrl-H** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

[New to Spyder? Read our tutorial](#)

Variable explorer | File explorer | Help

Python console

Console 7/A

WARNING:tensorflow:From C:\Anaconda\lib\site-packages\keras\backend\tensorflow_backend.py:199: The name tf.is_variable_initialized is deprecated. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From C:\Anaconda\lib\site-packages\keras\backend\tensorflow_backend.py:206: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

WARNING:tensorflow:From C:\Anaconda\lib\site-packages\keras\optimizers.py:790: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Anaconda\lib\site-packages\keras\backend\tensorflow_backend.py:3295: The name tf.log is deprecated. Please use tf.math.log instead.

WARNING:tensorflow:From C:\Anaconda\lib\site-packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From C:\Anaconda\lib\site-packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.

the top 2 emotion is: ('anger', 0.3775386)

the top 2 emotion is: ('anxiety/fear', 0.40550843)

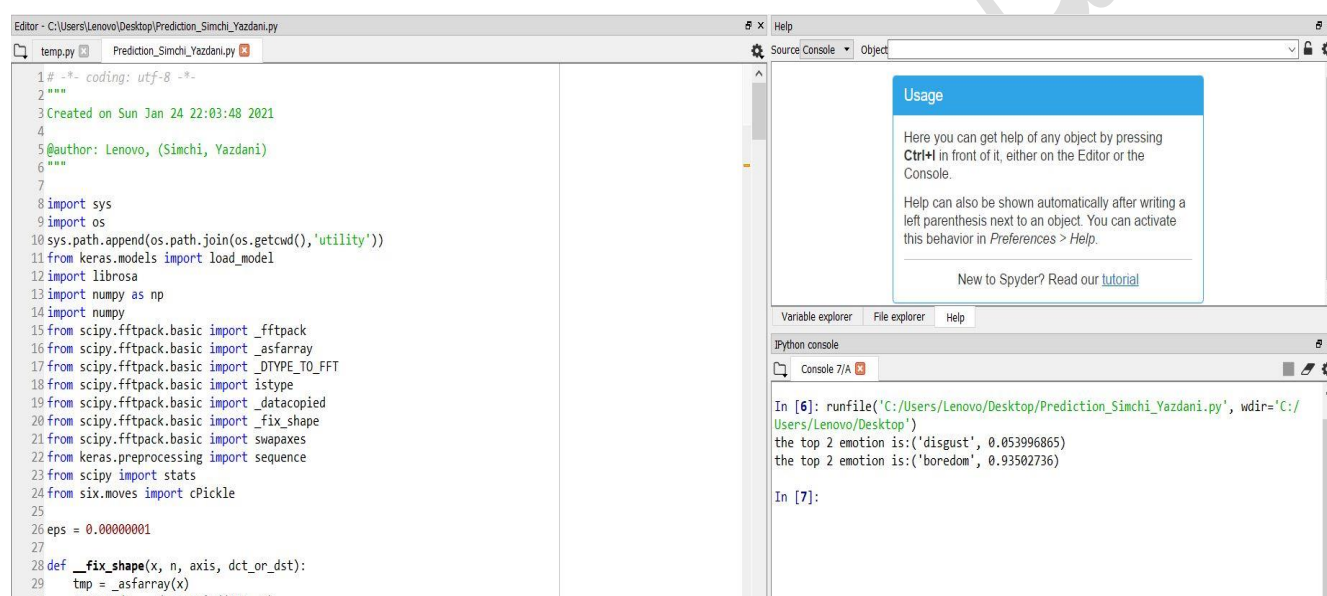
همانطور که از شکل فوق مشخص است، دو تا از بالاترین احتمالات را برگردانده و تشخیص احساس باتوجه به برچسب داده ی وارد شده کاملاً صحیح است (داده ی وارد شده دارای برچسب عصبانی بوده است که فایل آن در پوشه ی Berlin_Result قرار داده شده است)

حال می خواهیم بررسی کنیم آیا می توان مدل را بر روی دیتاست “Berlin” آموزش داد و پیش بینی را بر روی دیتاست “ShEMO” انجام داد ؟

*** باتوجه به اینکه برای هر داده برچسب نظیر آن را می دانیم درنتیجه باید قابل مقایسه باشد

درنتیجه، یکی از فایل های صوتی مربوط به دیتاست “ShEMO” را به مدل داده تا برچسب را برگرداند و سپس با مقایسه می توانیم به این سوال پاسخ دهیم.

پاسخ منفی است!!!



```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Jan 24 22:03:48 2021
4
5 @author: Lenovo, (Simchi, Yazdani)
6 """
7
8 import sys
9 import os
10 sys.path.append(os.path.join(os.getcwd(), 'utility'))
11 from keras.models import load_model
12 import librosa
13 import numpy as np
14 import numpy
15 from scipy.fftpack.basic import fftpack
16 from scipy.fftpack.basic import _asfarray
17 from scipy.fftpack.basic import _DTYPE_TO_FFT
18 from scipy.fftpack.basic import _istype
19 from scipy.fftpack.basic import _datacopied
20 from scipy.fftpack.basic import _fix_shape
21 from scipy.fftpack.basic import _swapaxes
22 from keras.preprocessing import sequence
23 from scipy import stats
24 from six.moves import cPickle
25
26 eps = 0.00000001
27
28 def _fix_shape(x, n, axis, dct_or_dst):
29     tmp = _asfarray(x)
30     new_shape = datacopied(tmp, n)
```

Usage

Here you can get help of any object by pressing **Ctrl+H** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

[New to Spyder? Read our tutorial](#)

Variable explorer | File explorer | Help

Python console

Console 7/A

In [6]: runfile('C:/Users/Lenovo/Desktop/Prediction_Simchi_Yazdani.py', wdir='C:/Users/Lenovo/Desktop')

the top 2 emotion is:('disgust', 0.053996865)

the top 2 emotion is:('boredom', 0.93502736)

In [7]:

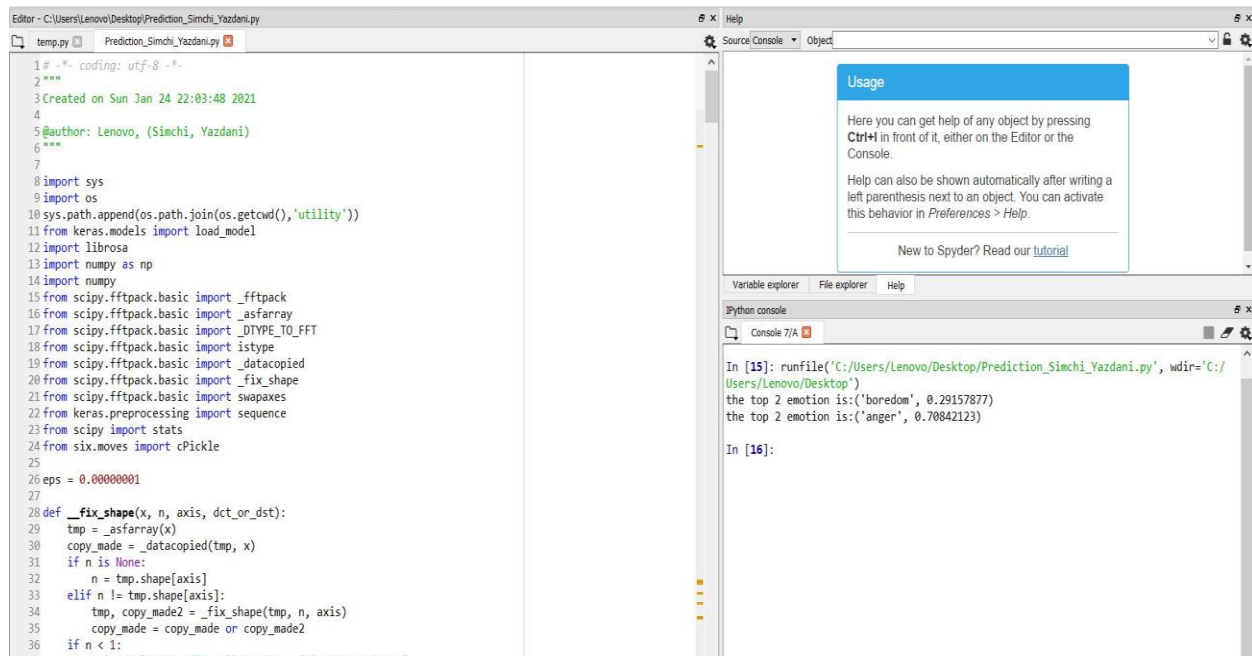
همانطور که از شکل فوق مشخص است، برچسب های داده شده، انزجار و یا دلزدگی را بیان می کند درحالی که صدای داده شده نوعی عصبانیت را نشان می دهد.

نتیجه گیری

نوع احساس موجود در جملات و هر زبان بستگی به گویش همان زبان و نحوه ی بیان آن فرهنگ دارد.

حال می خواهیم بر روی مدل کوچک دو کلاسه ی آموزش دیده شده ی بر روی دیتاست “ShEMO” پیشگویی را انجام دهیم

در صورتی که فایل ورودی دارای یکی از برچسب های موجود در دیتاست زمان آموزش باشد، به درستی پیش بینی را انجام می دهد



The screenshot shows the Spyder IDE interface. The main editor window displays a Python script named 'Prediction_Simchi_Yazdani.py'. The script includes a header with metadata (author: Lenovo, (Simchi, Yazdani)), imports for various libraries (sys, os, keras, librosa, numpy, scipy, tensorflow, etc.), and a function '_fix_shape'. The script is being executed, and the output is visible in the IPython console at the bottom. The console shows the results of two runs: the first run identifies the top 2 emotions as 'boredom' and 'anger', and the second run identifies the top 2 emotions as 'boredom' and 'anger'.

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Jan 24 22:03:48 2021
4
5 @author: Lenovo, (Simchi, Yazdani)
6 """
7
8 import sys
9 import os
10 sys.path.append(os.path.join(os.getcwd(), 'utility'))
11 from keras.models import load_model
12 import librosa
13 import numpy as np
14 import numpy
15 from scipy.fftpack.basic import _fftpack
16 from scipy.fftpack.basic import _asfarray
17 from scipy.fftpack.basic import _DTYPE_T0_FFT
18 from scipy.fftpack.basic import _istype
19 from scipy.fftpack.basic import _datacopied
20 from scipy.fftpack.basic import _fix_shape
21 from scipy.fftpack.basic import _swapaxes
22 from keras.preprocessing import sequence
23 from scipy import stats
24 from six.moves import cPickle
25
26 eps = 0.00000001
27
28 def _fix_shape(x, n, axis, dct_or_dst):
29     tmp = _asfarray(x)
30     copy_made = _datacopied(tmp, x)
31     if n is None:
32         n = tmp.shape[axis]
33     elif n != tmp.shape[axis]:
34         tmp, copy_made2 = _fix_shape(tmp, n, axis)
35         copy_made = copy_made or copy_made2
36     if n < 1:
```

Usage

Here you can get help of any object by pressing **Ctrl+H** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our [tutorial](#)

Variable explorer | File explorer | Help

Python console

Console 7/A

In [15]: runfile('C:/Users/Lenovo/Desktop/Prediction_Simchi_Yazdani.py', wdir='C:/Users/Lenovo/Desktop')

the top 2 emotion is:('boredom', 0.29157877)

the top 2 emotion is:('anger', 0.70842123)

In [16]:

با دقتی نزدیک به ۷۸ درصد، صدای ورودی را عصبی تشخیص می دهد که کاملاً صحیح است.

قسمت دوم

(بررسی کد دوم)

کد دوم

در ادامه، روش دوم پیاده سازی را بررسی خواهیم کرد. این روش شامل دو فایل مربوط به پیش پردازش داده ها و همچنین معماری شبکه عصبی مبتنی بر مکانیزم attention است، که به ترتیب در فایل های preprocessing.py و model.py پیاده سازی شده اند.

بخش اول

پیش پردازش داده ها

در این بخش ابتدا نیاز است مقادیر ثابتی که در طول اجرای الگوریتم مورد نیاز هستند را مشخص کنیم. به عنوان مثال می توان به فرکانس نمونه برداری که برابر با ۱۶ کیلوهرتز است اشاره کرد. لیست کامل این مقادیر در تصویر زیر مشخص شده است.

```
sr = 16000
duration = 5
frame_length = 512
N_FRAMES = math.ceil(sr * duration / frame_length)
N_FEATURES = 46
N_EMOTIONS = 6
emo_codes = {"A": 0, "W": 1, "F": 2, "H": 3, "S": 4, "N": 5}
emo_labels = ["anger", "surprise", "fear", "happiness", "sadness", "neutral"]
path = "ShEMO"
```

بهتر است مدت زمان هر فریم بین ۲۰ تا ۳۰ میلی ثانیه باشد، اما به دلیل محدودیت سخت افزاری این مقدار بر روی ۵۱۲ تنظیم شده است. همچنین تعداد ویژگی هایی که از هر فریم استخراج خواهد شد، ۴۶ عدد است.

تابع feature_extraction

این تابع که به منظور استخراج ویژگی ها از سیگنال گفتار مورد نظر پیاده سازی شده است، خود دارای چند بخش میباشد.

۱- خواندن فایل های صوتی و ذخیره سری زمانی آنها

۲- padding هر کدام از آنها در جهت یکسان شدن از نظر زمانی (به مدت زمان ۵ ثانیه)

۳- استخراج و ذخیره ویژگی های مربوط به هر فریمبه وسیله کتابخانه librosa

۴- ذخیره و بازگرداندن بردار های ویژگی هر فریم و همچنین برچسب مورد نظر آنها به عنوان خروجی

تابع get_train_test

این تابع با توجه به بردارهای ویژگی استخراج شده و همچنین برچسب های ذخیره شده هر کدام از آنها در مرحله قبل، اقدام به ساختن مجموعه داده های آموزشی و تست کرده و هر یک را به عنوان خروجی بازمیگرداند. گفتنی است که برچسب های مورد نظر با روش one-hot encoding به صورت بردار هایی که فقط یکی از مقادیر آنها ۱ و بقیه مقادیر ۰ هستند درآمده است.

در ادامه تصویری از ویژگی های استخراج شده برای هر فریم و نحوه ذخیره آنها را خواهیم دید.

```
frames = []
spectral_centroid = librosa.feature.spectral_centroid(y=y, sr=sr, hop_length=frame_length)[0]
spectral_contrast = librosa.feature.spectral_contrast(y=y, sr=sr, hop_length=frame_length)[0]
spectral_bandwidth = librosa.feature.spectral_bandwidth(y=y, sr=sr, hop_length=frame_length)[0]
spectral_rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr, hop_length=frame_length)[0]
zero_crossing_rate = librosa.feature.zero_crossing_rate(y, hop_length=frame_length)[0]
S, phase = librosa.magphase(librosa.stft(y=y, hop_length=frame_length))
rms = librosa.feature.rms(y=y, hop_length=frame_length, S=S)[0]
mfcc = librosa.feature.mfcc(y=y, sr=sr, hop_length=frame_length)
mfcc_der = librosa.feature.delta(mfcc)
for i in range(N_FRAMES):
    f = [spectral_centroid[i], spectral_contrast[i], spectral_bandwidth[i], spectral_rolloff[i],
         zero_crossing_rate[i], rms[i]]
    for m_coeff in mfcc[:, i]:
        f.append(m_coeff)
    for m_coeff_der in mfcc_der[:, i]:
        f.append(m_coeff_der)
    frames.append(f)
```


بخش دوم

پیاده سازی، آموزش و تست مدل

پیاده سازی بخش دوم که مبتنی بر کتابخانه keras می باشد، شامل بخش های مختلفی از جمله ایجاد مدل مورد نظر، آموزش آن با توجه به مجموعه داده های آموزشی که در بخش قبل ایجاد شد و همچنین تست مدل با توجه به مجموعه داده تست است.

تابع create_model

به وسیله این تابع، مدل مورد نظر خود که از مکانیزم توجه استفاده میکند را، ایجاد میکنیم. تعداد واحدها یا نورون های هر لایه به طور پیش فرض، ۲۵۶ عدد در نظر گرفته شده است. سایر جزئیات این مدل به شرح زیر است:

Model: "Attention_BLSTM"			

Layer (type)	Output Shape	Param #	Connected to

input_1 (InputLayer)	[(None, 157, 46)]	0	

bidirectional (Bidirectional)	[(None, 157, 512), (620544	input_1[0][0]

dense (Dense)	(None, 157, 256)	131072	bidirectional[0][0]

dense_1 (Dense)	(None, 157, 1)	256	dense[0][0]

flatten (Flatten)	(None, 157)	0	dense_1[0][0]

lambda (Lambda)	(None, 157)	0	flatten[0][0]

alpha (Softmax)	(None, 157)	0	lambda[0][0]

dot (Dot)	(None, 512)	0	bidirectional[0][0] alpha[0][0]

concatenate (Concatenate)	(None, 512)	0	bidirectional[0][1] bidirectional[0][3]

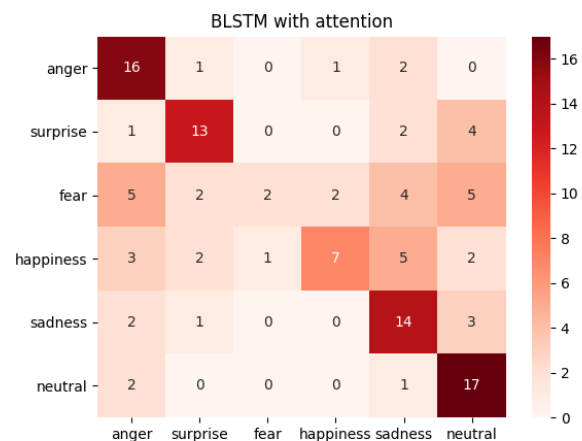
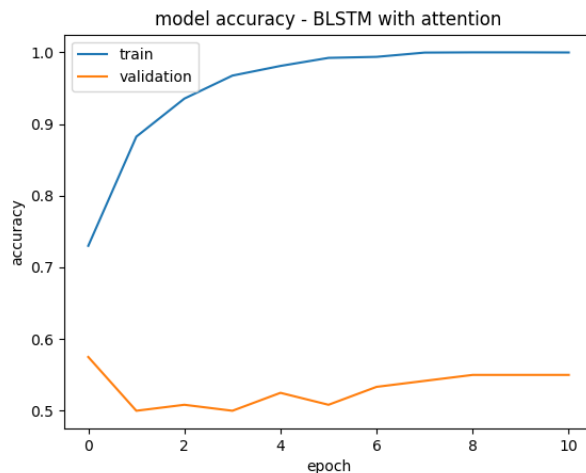
concatenate_1 (Concatenate)	(None, 1024)	0	dot[0][0] concatenate[0][0]

dense_2 (Dense)	(None, 6)	6150	concatenate_1[0][0]

Total params: 758,022			
Trainable params: 758,022			
Non-trainable params: 0			

تابع train_and_test_model

در این تابع، مجموعه داده های آموزشی و تست که در بخش اول تعبیه شد را روی مدل مورد نظر آموزش داده و تست میکنیم. برای آموزش مدل از بهینه ساز Adam و همچنین روش Early Stopping برای جلوگیری از بیش برآزش استفاده شده است. در نهایت نمودار دقت و ماتریس کوواریانس مربوط به این آزمایش در تصاویر زیر آمده است.



در تصویر زیر نیز بخشی از مراحل آموزش مدل و همچنین دقت نهایی مدل روی مجموعه داده ShEMO مشخص شده است.

```
Epoch 8/30
195/195 - 19s - loss: 0.0096 - accuracy: 0.9995 - val_loss: 2.8466 - val_accuracy: 0.5417

Epoch 00008: val_loss did not improve from 1.63058
Epoch 9/30
195/195 - 19s - loss: 0.0024 - accuracy: 0.9998 - val_loss: 2.9945 - val_accuracy: 0.5500

Epoch 00009: val_loss did not improve from 1.63058
Epoch 10/30
195/195 - 20s - loss: 0.0017 - accuracy: 0.9998 - val_loss: 2.9959 - val_accuracy: 0.5500

Epoch 00010: val_loss did not improve from 1.63058
Epoch 11/30
195/195 - 24s - loss: 0.0019 - accuracy: 0.9997 - val_loss: 3.0648 - val_accuracy: 0.5500

Epoch 00011: val_loss did not improve from 1.63058
Epoch 00011: early stopping
model saved
4/4 [=====] - 2s 29ms/step - loss: 1.2961 - accuracy: 0.6362

test accuracy: 57.50%
```

قسمت سوم

(شرح نحوه ی اجرای کد اول)

Second part of report on the final project of the Digital Speech Processing

Hossein Simchi, Ali Yazdani, Dr. Yasser Shekofteh
Computer Science and Engineering , Shahid Beheshti University of Tehran , Iran
[h.simchi@mail.sbu.ac.ir]

February 2, 2021

Abstract

In the second part, we try to give a complete description of the project in various stages. Also in each stage we try to express the explanations related to that stage in very short sentences

1 Automatic SER using recurrent neural networks with local attention

To implement the algorithm described in the article mentioned, we must perform the steps stated at the time of presentation, ie extracting the feature and then categorizing them correctly.

1.1 Steps and How the code works

- 1) Specify the name of the dataset, dataset type = 'shEMO'
 - 2) **Call dataset class** : The class of the data is called, and we give it the name and path of the data.
 - 3) **dataset class** : When we call the dataset class, we also need to get a label for our data. The fourth letter of our data name also specifies their label. In this project we have used the Librosa library to read audio files.
 - 4) Then, using the **cPickle command**, we put each data along with its label inside 'shEMO db.p'
 - 5) **Feature Extraction Function** : Using the cPickle command, we read the data set stored in the previous step and pass it to the feature extraction function to extract the feature.
- We will discuss feature extraction functions in detail in a separate section, and here we will discuss the general process of written code.
- 6) **feature extract function** : We do this for every data in the database
- The class of the stFeatureExtraction is called.

- **Harmonic ratio and pitch** : Using stFeatureSpeed Function and then Then we add its output to the feature vector obtained from the previous step.
 - **Z-normalized** : We normalize the feature vector
 - Add the obtained feature vector to the total feature vector.
- In fact, in this step we try to get its features for each data and add it to the total Feature vector.
- Finally, we put the total feature vector inside '**shEMO features.p**' using the **cPickle command** to use it later. Our final vector contains a number of feature vectors, which is equal to the number of data in the dataset.

1.2 stFeatureExtraction

This function implements the shor-term windowing process. For each short-term window a set of features is extracted. This results to a sequence of feature vectors, stored in a numpy matrix.

ARGUMENTS :

- signal: the input signal samples
- Fs: the sampling freq (in Hz)
- Win: the short-term window size (in samples)
- Step: the short-term window step (in samples)

RETURNS :

- stFeatures: a numpy array (numOfFeatures x numOfShortTermWindows)

1.2.1 stFeatureExtraction Steps

- 1) Signal normalization
- 2) Compute total number of samples
- 3) Compute the triangular filter banks used in the **mfcc** calculation

We must note that the final vector of features is equal to the sum of TimeSpectralFeatures and ChromaFeatures. For each window we have to do the following calculations :

- Get current window : $x = \text{signal}[\text{curPos}:\text{curPos}+\text{Win}]$
- Update window position
- get Fast Fourier Transform (FFT) magnitude : $X = \text{abs}(\text{fft}(x))$
- Normalize FFT
- To extract the Spectral Flux feature, we need to save the Fast Fourier Transform (FFT) magnitude of the previous frame as well.
- We need to define a list named '**curFV**' with the same number of rows as the number of attributes and one column.

Now, we extract the following features and enter the features in the list of Hussein from zero to the end, respectively

- 1) Zero crossing rate : stZCR Function
- 2) Short-term energy : stEnergy Function
- 3) Short-term entropy of energy : stEnergyEntropy Function
- 4) Spectral centroid and spread : stSpectralCentroidAndSpread Function
- 5) Spectral entropy : stSpectralEntropy Function

- 6) Spectral Flux : stSpectralFlux Function
 - 7) spectral Rolloff : stSpectralRollOff Function
 - 8) MFCC : stMFCC Function
 - 9) Chroma : stChromaFeatures
- As a result, we have extracted nine features for each window.

1.2.2 Feature Extraction Functions

Zero crossing rate (stZCR Function)

- applying the sign function to each window.
- calculating the difference in value at each point from the previous point (Numpy.diff())
- calculating the absolute value of the values obtained
- Return the sum of all the resulting values as the final value

Zero crossing rates are also used for Voice activity detection (VAD), which determines whether human speech is present in an audio segment or not.

Short-term energy (stEnergy Function)

Sum values of all frequencies in each window to the power of two and then divide by the number of frequencies in each window. Thus short term energy can be used for voiced, unvoiced and silence classification of speech

Short-term entropy of energy (stEnergyEntropy Function)

To compute entropy of energy. Spectral entropy features, used for voice activity detection and the entropy is a measure of disorganization and it can be used to measure the peakiness of a distribution.

Spectral centroid and spread (stSpectralCentroidAndSpread Function)

To compute spectral centroid and spread of frame (given abs(FFT))

The spectral centroid is a measure used to characterise a spectrum. It indicates where the center of mass of the spectrum is located. Perceptually, it has a robust connection with the impression of brightness of a sound. Noise-like signals have usually a large spectral spread, while individual tonal sounds with isolated peaks will result in a low spectral spread

Spectral entropy (stSpectralEntropy Function)

To compute the spectral entropy.

Spectral Flux (stSpectralFlux Function)

To Compute the spectral flux feature of the current frame.

ARGUMENTS:

- X: the abs(fft) of the current frame
- Xpre: the abs(fft) of the previous frame

spectral Rolloff (stSpectralRollOff Function)

To compute spectral roll-off

MFCC (stMFCC Function)

To compute the MFCCs of a frame, given the fft magnitude.

ARGUMENTS:

- X: fft magnitude abs(FFT)
- fbank: filter bank (mfccInitFilterBanks Function)

RETURN:

- ceps: MFCCs (13 element vector)

Note that to calculate the MFCCs Features, we must obtain the MFCC filter bank and DCT matrix.

Chroma (stChromaFeatures)

To Compute the Chroma feature of the current frame.

1.3 Create Model (Bidirectional LSTM)

After extracting the desired features and then saving the features in different files (according to what was stated in the previous section), in this section, the model is designed to be trained on the data collected from audio files.

1.4 Prediction on Model

After training the model, in this step we need to give data to the trained model in the previous section, to receive the desired label.