به نام خدا

گزارش پیاده‌سازی ساختار Speech classifier

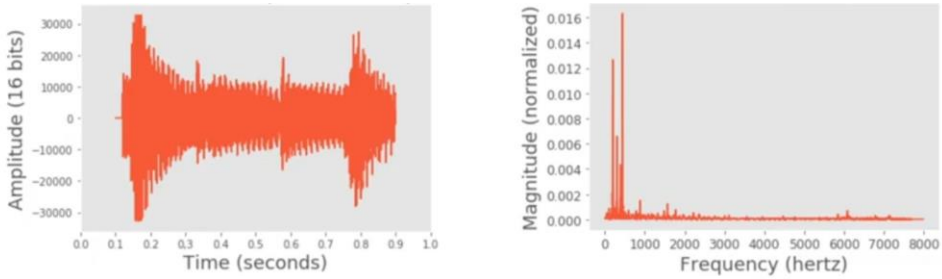
نگارش: فاطمه کربلائی

فهرست مطالب

۱- تعریف مساله و رویکرد حل پیشنهادی

یکی از رایج‌ترین فرمت‌های تعریف شده و قابل استفاده دیتا فایل صوتی است. این ساختار داده به دلایل متعددی نظیر وجود ترتیب در محتوی آن، امکان فشرده‌سازی و کاربرد بسیار آن در سیستم‌های طراحی شده جهت تسهیل امور روزمره انسان و برقراری ارتباطات، اهمیتی بیش از پیش دارد.

اولین نکته در درک ساختار فایل صوتی آن است که ساختار بررسی شکل سیگنال در حوزه زمان شاید اطلاعات کامل و واضحی جهت اعمال پردازش های لازم، به دست ندهد. از این رو، یکی از راهجل های رایج حل این مساله استفاده از تبدیلات متفاوت نظیر تبدیل فوریه جهت نگاشت سیگنال به فضای جدید است.

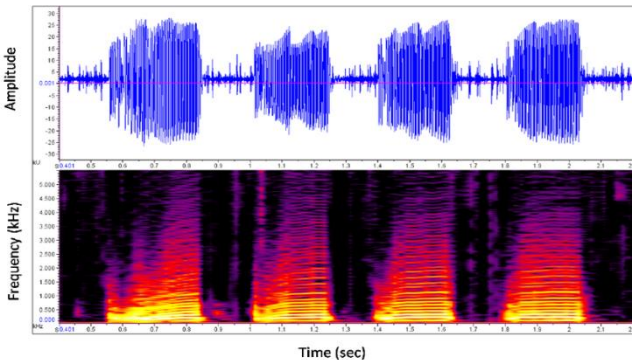


شکل۱ سیگنال صوت در حوزه زمان و تبدیل فوریه

در فایل‌های صوتی معمولا صدا با نرخ 44.1KHz ضبط می‌شود. همچنین طبق نظریه شانون، نرخ مناسب نمونه‌برداری می‌تواند عددی حدود 20KHz باشد. طبق شکل شماره ۱ و همچنین مراجع بررسی شده، در مورد صداهای ضبط شده انسان بیشتر مولفه های با magnitude قابل توجه در فرکانس‌های پایین جمع شده اند.

روش پیشنهادی جهت طرح دسته‌بند فایل‌های صوتی، از یک تبدیل فضا استفاده می‌کند. به طوری که معادل با هر سیگنال صوتی تصویری تحت عنوان spectrogram استفاده شده است.

استفاده از spectrogram یک راه مرسوم جهت مصورسازی قدرت سیگنال یا میزان loudness سیگنال صوتی است. در این نوع تصاویر می‌توان قدرت مولفه های فرکانسی متفاوت موجود در سیگنال را در طول زمان مشاهده و تغییرات آن‌ها را بررسی نمود.



شکل۲ شکل سیگنال صوتی و spectrogram آن

۲- پیاده‌سازی

برای پیاده‌سازی ساختار توصیف شده، چند بخش متفاوت هر یک به صورت یک ماژول تعریف شده اند :

پیش پردازش داده‌های صوتی : clean.py

تعریف روند آموزش مدل‌ها: train.py

تعریف مدل‌ها : models.py

استفاده از مدل جهت دسته‌بندی داده‌های تست : test.py

۲-۱ تنظیمات محیط پیاده سازی و نصب موارد مورد نیاز

به کمک دستورات زیر، ابتدا یک virtual env با نام audio ایجاد و کل مراحل نصب و پیاده‌سازی را در آن انجام می‌شود.

```
conda create -n audio python=3.7
activate audio
pip install -r requirements.txt
```

۲-۲ پیش پردازش (clean.py و wavfile_conversion.py)

یکی ازمواردی که قبل از اجرای کد باید به آن توجه شود فرمت فایل‌های صوتی دیتاست است.

mp3. یکی از فرمت‌های ذخیره‌ازی فایل صوتی است که در فایل در این حالت فشرده‌سازی نیز شده است. در نتیجه جهت اعمال پردازش و استخراج ویژگی‌ها این فرمت مناسب نبوده و سربار پردازشی را به اجرای کد اضافه می‌کند. از این رو باید تمامی فایل‌های صوتی به فرمت ساده‌تر و بدون فشرده سازی تبدیل شوند. یکی از فرمت‌های پیشنهادیwav. است. فایل‌های wav. فشرده‌سازی نشده اند و سرعت load شدن آن‌ها به نسبت بسیار بیشتر خواهد بود. پس در ابتدا و قبل از استفاده از فایل‌ها در کد و آموزش مدل توسط آن‌ها ابتدا نوع آن‌ها تغییر داده می‌شود.

ماژول wavfile conversion.py با دو کتابخانه librosa و soundfile تمامی نمونه‌های صوتی را خوانده و پس از تبدیل با rate=16000 آن‌ها را با نام و در مسیر جدید new_wavefile ذخیره می‌کند.

A. clean.py

تابع envelope :

```
def envelope(y, rate, threshold):
    mask = []
    y = pd.Series(y).apply(np.abs)
    y_mean = y.rolling(window=int(rate/20),
                       min_periods=1,
                       center=True).max()
    for mean in y_mean:
        if mean > threshold:
            mask.append(True)
        else:
            mask.append(False)
    return mask, y_mean
```

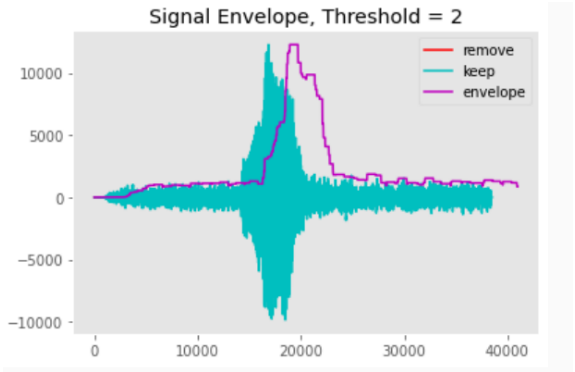
همانطور که پیشتر گفته شد، فایل‌های صوتی مخصوصا از نوع speech معمولا دارای مولفه‌های فرکانس پایین هستند.

در واقع روی محور افقی (فرکانس) از جایی به بعد اندازه مولفه های فرکانسی بسیار کوچک شده و به صفر میل می‌کند.

(اصطلاحا نواحی dead space)

تابع envelope با در نظر داشتن این نکته، و با انتخاب مقدار آستانه مناسب، فقط قسمتی از سیگنال صوتی را که مربوط به فرکانس پایین است حفظ کرده و دیگر بخش‌ها را حذف می‌کند. در نتیجه dead space های موجود در فایل‌های صوتی با بررسی یک شرط حذف می‌شود.

به عنوان مثال اگر این بخش برای یک فایل خاص اجرا شود خروجی به صورت زیر خواهد بود :



شکل ۳ سیگنال صوتی و envelope

پس از تنظیم مقدار درست threshold به صورتی که تا حد امکان مرز envelope بر سیگنال منطبق باشد، حال در انتهای فایل clean.py دستور خط اخر مربوط به split-waves باید اجرا شود. (بار اول خط ۱۳۵ اجرا شده و بار دوم این خط کامنت می‌شود و خط ۱۳۶ اجرا خواهد شد.)

```
135     test_threshold(args)
136     #split_wavs(args)
```

B. train.py

کلاس Datagenerator: برای افزایش سرعت اجرا و تطابق با multi processing دیتا به چند batch تقسیم می‌شود.
خروجی تابع __len__ : تعداد batch در هر epoch (از طریق تقسیم تعداد کل دیتا به batch-size)

خروجی تابع __getitem__ : دیتای مربوط به هر batch به صورت دو پارامتر X و Y (خروجی مورد نظر یعنی Y با فرمت one hot encoded و دارای ابعاد (batch-size,n-classes) است.)

تابع on_epoch_end : در انتهای هر epoch دیتا را shuffle می‌کند.

train : در ابتدا چند پارامتر قابل تغییر مثل sr (sample rate) و آدرس محل ذخیره سازی مدل‌ها و خود مدل‌های متفاوت به عنوان ورودی دریافت و تنظیم می‌شود. سپس لیبل هر دیتا با توجه به نام فولدر آن از مسیر ارائه شده به عنوان src_root بدست می‌آید.

tg و vg دو تولیدکننده دیتا برای train و validation هستند که با فراخوانی Datagenerator ابتدا داده‌ها را آماده می-کنند و سپس بافراخوانی train، یکی از سه مدل Conv1D ، Conv2D و LSTM انتخاب شده و آموزش مدل آغاز می‌شود

CSV_logger : اطلاعات مربوط به کل ساختار شبکه عصبی استفاده شده طبق شکل زیر در هر بار اجرای کد train به صورت یک فایل CSV برای هر مدل تولید می‌شود. این اطلاعات جهت مقایسه عملکرد مدل‌های مختلف مفید است که در ادامه با رسم نمودار مقایسه می‌شوند. در نهایت مدل روی دیتای train با epoch=30 ، fit می‌شود

C. models.py

اصلی ترین بخش این پیاده‌سازی طراحی ساختار مناسبی است که بتواند در نهایت عمل دسته‌بندی را تحقق بخشد. در روند طراحی یک دسته‌بند هرچه مرحله استخراج ویژگی بهتر و دقیق‌تر انجام شود، لیبل‌های پیش‌بینی شده توسط مدل نیز درنهایت دقیق‌تر و کم خطا تر خواهد بود. از این رو طبق بررسی‌های انجام شده از منابع مختلف، این نتیجه حاصل شد که استفاده از شبکه Con1D ، Conv2D و LSTM هر سه برای دسته‌بندی فایل‌های صوتی مرسوم بوده است. در نتیجه جهت ایجاد امکان مقایسه هر سه ساختار در این فایل پیاده‌سازی شده اند که مختصری در رابطه با لایه‌ها و تفاوت‌های ساختاری آن‌ها ارائه خواهد شد.

Con1D

در این ساختار از لایه‌های Maxpool جهت کاهش ابعاد استفاده شده است. لایه TimeDistributed در طول زمان همه فرکانس‌های مختلف را در نظر می‌گیرد. روی نمودار frequency spectrogram ممکن است به علت فشردگی ویژگی‌ها در تصویر، برخی از ویژگی‌ها از نظر دور بماند اما شبکه با استخراج الگوهایی نظیر edge، corner و ... در تصویر این ویژگی-ها را استخراج می‌کند. در واقع کنار هم قرار گرفتن و ترکیب این ویژگی‌ها در تصویر باعث معنادار شدن آن‌ها می‌شود.

لایه Dropout جهت اعمال regularization با هدف جلوگیری از overfitt (یا کاهش آن) اضافه شده است.

لایه Dense نیز به جهت تخصیص لیبل در دسته‌بندی نهایی آورده شده است.

Conv2D

این ساختار مشابه با روش‌هایی که در computer vision استفاده می‌شود، است. لایه TimeDistributed ندارد و به جای عملیات کانولوشن یک بعدی، لایه Conv2D استفاده شده است.

LSTM

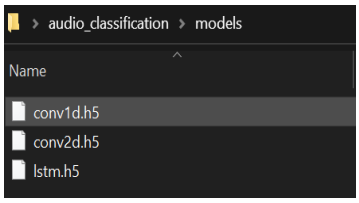
تفاوت اصلی این شبکه با دو ساختار دیگر در این است که علاوه بر featureها، تغییرات آن‌ها در طول زمان را نیز در نظر می‌گیرد. یکی از لایه‌های مهم در اینجا bidirectional LSTM است. یعنی گرادیانی که در دو جهت اعمال می‌شود و مقدار گرادیان در روند update شدن در دو جهت افزایشی و کاهشی قابل تغییر است.

D. predict.py

برای تست مدل پیاده‌سازی شده با هر فایل صوتی، پس از طی مراحل مربوط به پیش پردازش باید فایل را به عنوان ورودی این ماژول تنظیم نمود. در این پیاده سازی، بخشی از داده های train (۰.۱ از کل داده‌ها) به عنوان داده validation در نظر گرفته شده تا بتوان عملکرد predict را به صورت نمادین مشاهده و گزارش نمود. همچنین دیگر خواسته‌های صورت

سوال نظیر ماتریس درهم ریختگی و نمودار های ROC و نمودار acc (که به صورت اختیاری و برای تحلیل بهتر آورده شده اند،) قابل تعریف و مشاهده باشند.

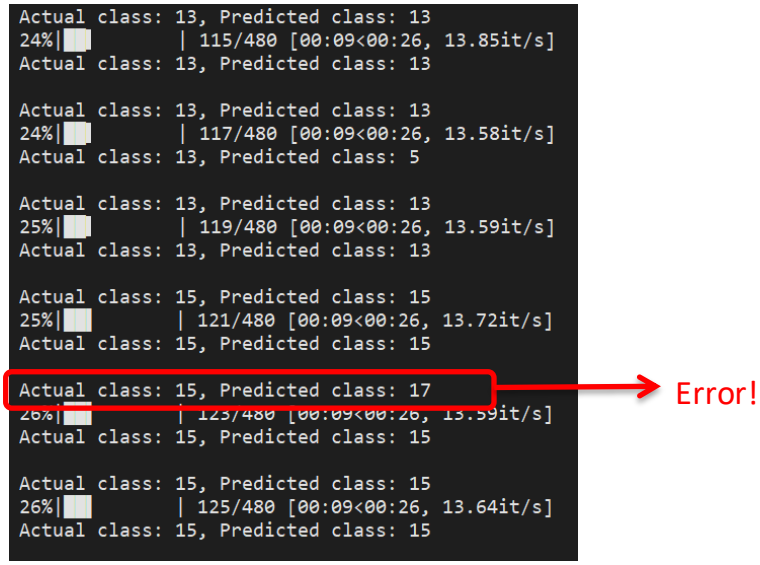
در این فایل تنها یک تابع make prediction وجود دارد. در ابتدا از نتایج و خروجی های بخش‌های گذشته، دیتای پردازش شده و همچنین یکی از سه مدل را فراخوانی می‌کند. برای انتخاب اینکه کدام مدل استفاده شود می‌توان در قسمتی از کد که در زیر آورده شده است، نام یکی دیگر از مدل‌ها که پیش از این در پوشه models با پسوند .h5 ذخیره شده اند را، انتخاب نمود.




```
54  if __name__ == '__main__':
55
56      parser = argparse.ArgumentParser(description='Audio Classification Training')
57      parser.add_argument('--model_fn', type=str, default='models/lstm.h5'
```

سپس batch های مختلف دیتا را تشکیل شده و از طریق عبارت model.predict(X_batch) خواسته اصلی مساله یعنی ارائه پیش‌بینی برای دسته‌بندی هر دیتا صورت می‌پذیرد.

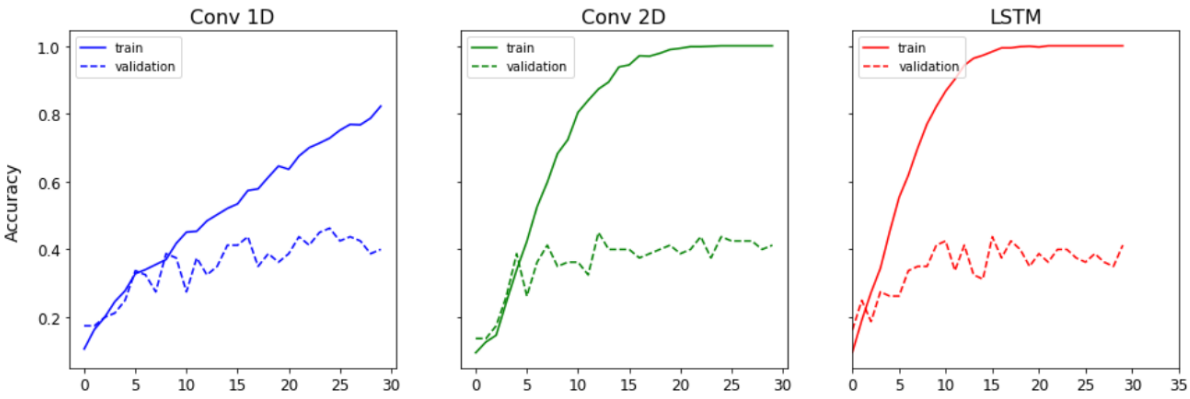
پس از اجرای این بخش، در محیط cmd لیبل درست هر نمونه و لیبل پیش‌بینی شده برای تمامی آن‌ها به صورت زیر نمایش داده خواهد شد. لازم به ذکر است که طبیعتا تعدادی از این پیش‌بینی‌ها نادرست بوده و acc مدل ۱۰۰ نیست.



۳- نتایج و نمودارها

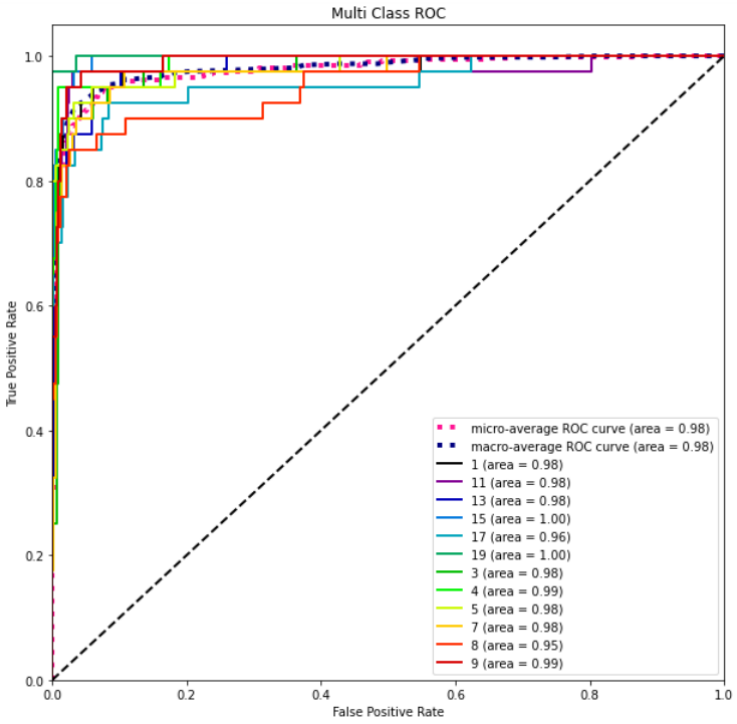
پس از اجرای فایل train برای هر سه حالت متفاوت از شبکه‌ها، با داشتن سه فایل CSV می‌توان نمودار accuracy این سه ساختار را برای هر دو دسته داده‌های آموزشی و test آورده شده است.

طبق شکل۴ ، Conv1D نسبت به دو ساختار دیگر کمترین مقدار overfit را دارد. نکته مهمی در تحلیل شکل‌ها این است که صددرصد قابل اعتماد نیستند . زیرا دیتاست استفاده شده خیلی محدود است و این مساله ممکن است نتیجه کار را تحت تاثیر قرار دهد. به عنوان یک تحلیل نظری و پیش‌بینی (شخصی) می‌شود که LSTM چون تغییرات زمانی را نیز در نظر میگیرد، احتمالا بهتر عمل خواهد کرد.



شکل۴ نمودارaccuracy برحسب epoch برای هر سه مدل

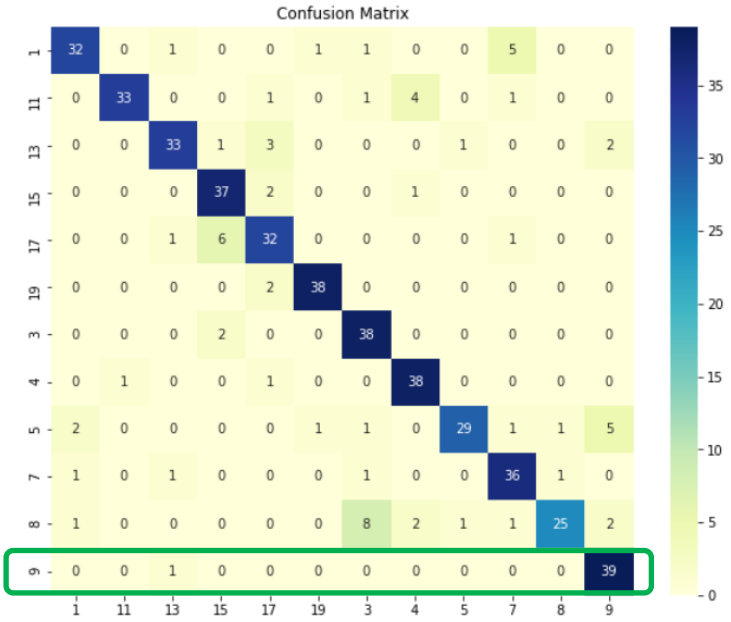
برای بررسی دقت عملکرد و کیفیت دسته بندی به ازای هر یک از لیبل‌ها به تفکیک نمودار ROC نیز رسم شده است. به طور خلاصه میتوان گفت با توجه به اینکه حالت ایده‌آل برای نمودار ROC تا حد امکان دور بودن شکل نمودار از نیمساز و نزدیک بوده آن به نقطه ایده‌آل یعنی (0,1) است، بهترین عملکرد این دسته‌بند برای لیبل ۱۹ است.



شکل۵ ROC برای ۱۲ کلاس داده‌ها

در نهایت آخرین نموداری که به طور دقیق‌تر و با تعداد نمونه‌ها عملکرد دسته‌بند به ازای هر لیبل را نشان می‌دهد، confusion matrix است. داده‌های لیبلی که با بیشترین دقت دسته‌بندی شده باشد، بزرگترین عدد روی قطر اصلی و کمترین مجموع اعداد روی سطر مربوط به همان قطر را دارد. (یعنی بیشترین تعداد داده درست بیش‌بینی شده و کمترین پیش‌بینی اشتباه)

در اینجا لیبل۹ ، بیشترین پیش‌بینی درست (۳۹) و کمترین مجموع سطری (۱) را دارد.



شکل۶ Confusion matrix