Task 3

本实验框架仍采用项目一二所用的框架[[Drawing 2023-01-29 23.35.06.excalidraw]]

Quick Start

与实验一不同的地方在于这次需要处理语音，思路为处理音频，可视化出一张图，然后接着用CNN做分类，所以只需要在处理数据和提取特征的地方进行一些修改即可

题目一为对比拓展音频到更高领域与否进行分类的结果对比
不拓展领域为机器学习做法时提取声学特征进行分类
拓展领域为用了Spectrogram等音频特征提取技术提取高维数据进行音频分类

处理数据

Resample

音频的采样率可能不同，读取到大多音频的sample_rate 都为16000, 因此将所有音频一个一个读取，调整采样率到 16000.

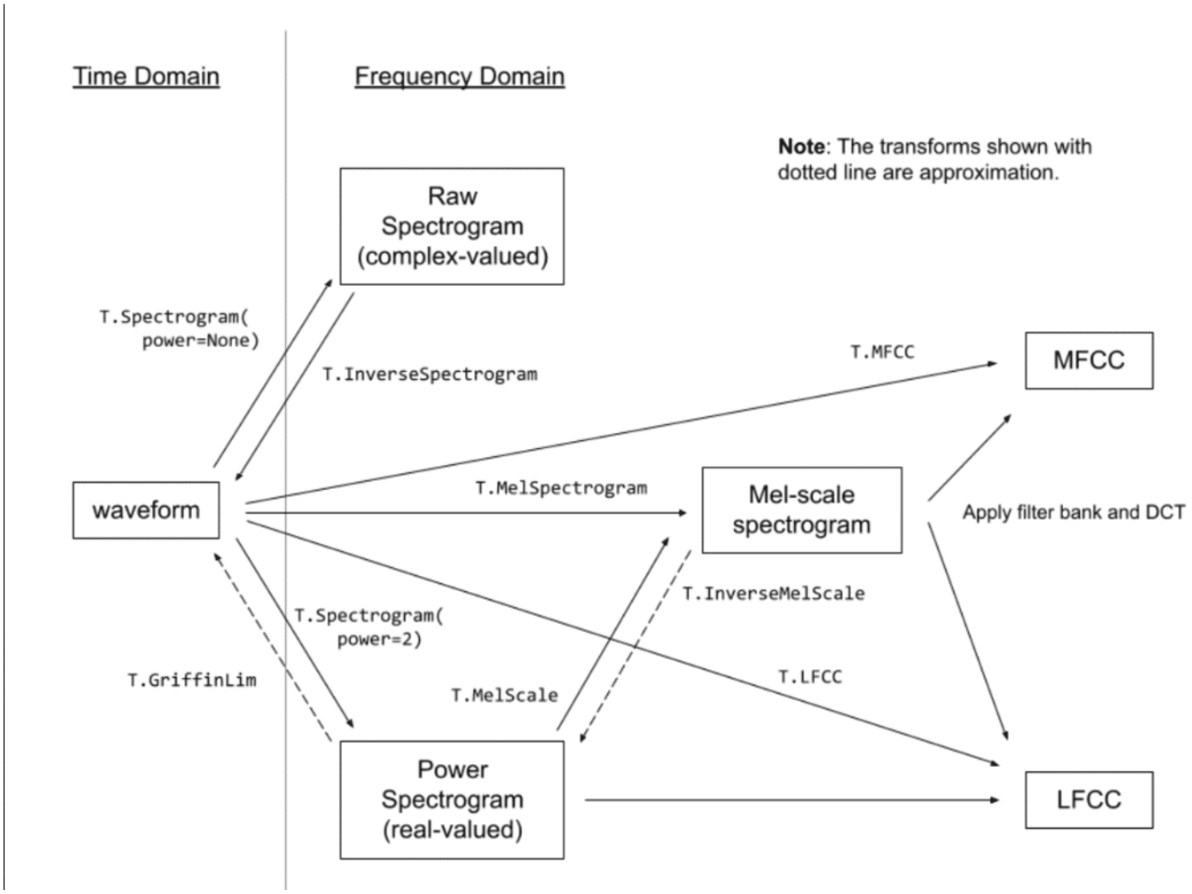
Reshape

为了让音频统一长度，我对音频采取了一些裁剪的动作，如果超过了一定范围则裁剪掉多余的部分，如果不足则用样本作padding补足。

Mix down channels

音频的channels 数可能不同，如果音频的channel 大于1， 我们把取其所有 channel的平均数来作为输入.

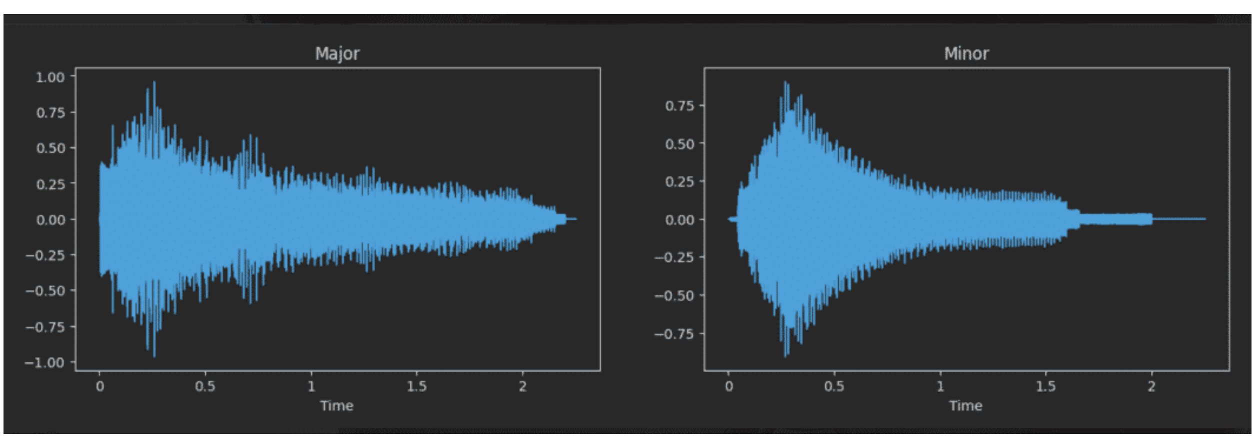
提取特征



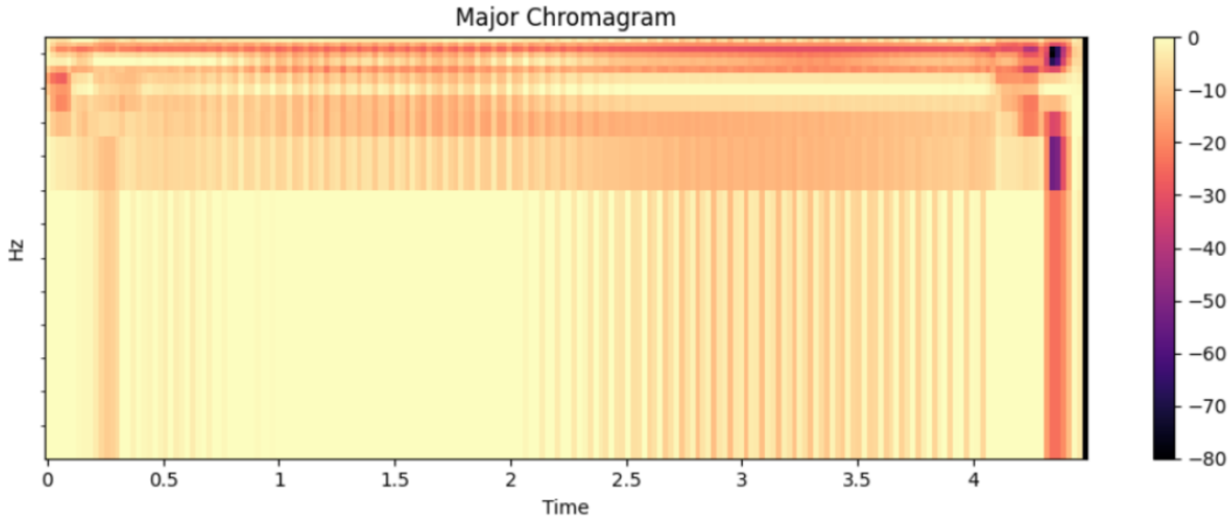
这里我用了Spectrogram, Mel_Spectrogram, MFCC, STFT四种特征，保证n_fft=512, sample_rate = 16000, n_mel=40(如果存在此参数)的情况下，用相同的结构相同的网络进行分类任务.

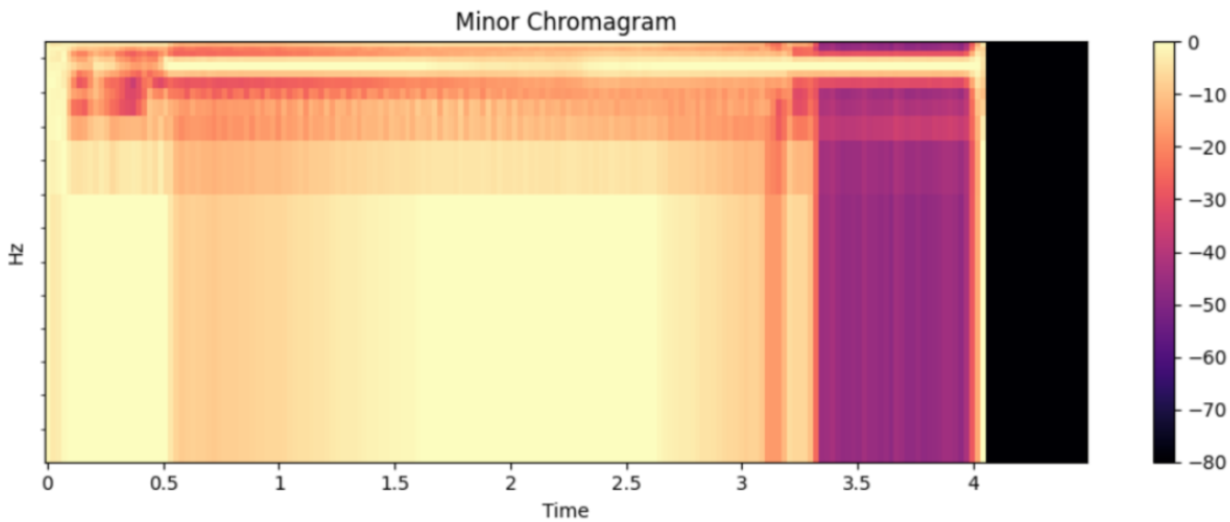
可视化展示

- Raw data

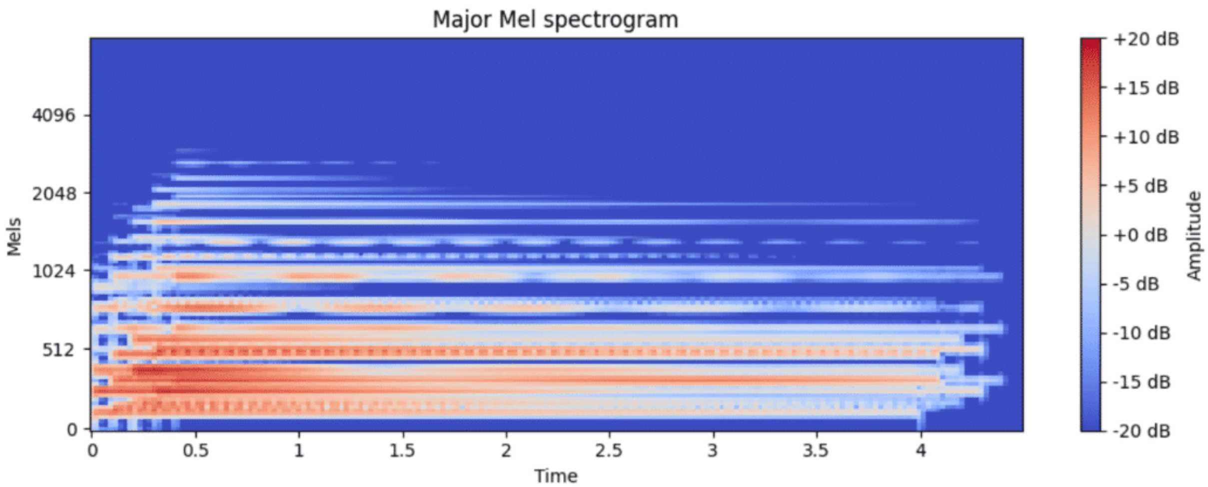


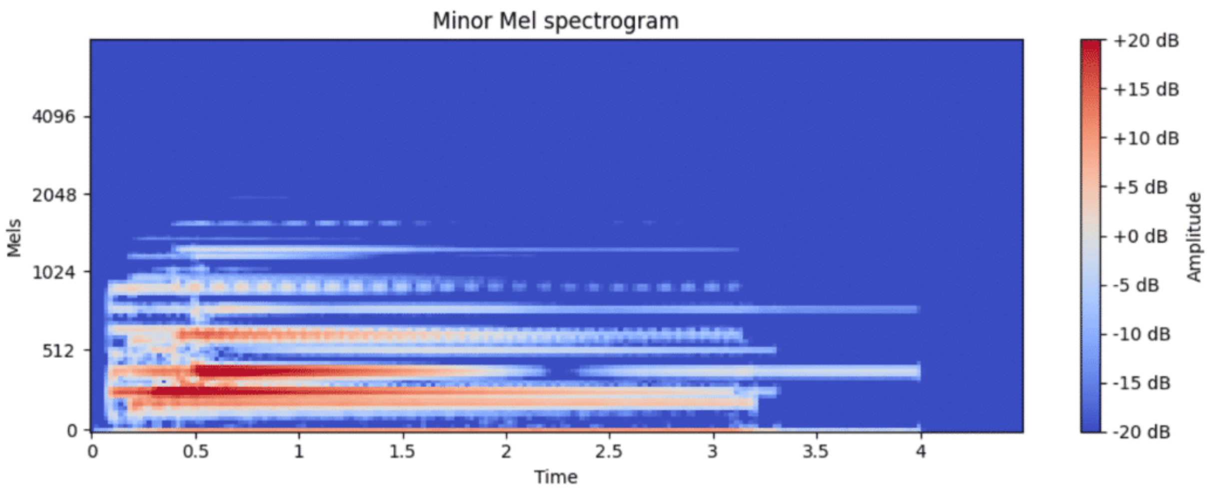
- Chroma_stft



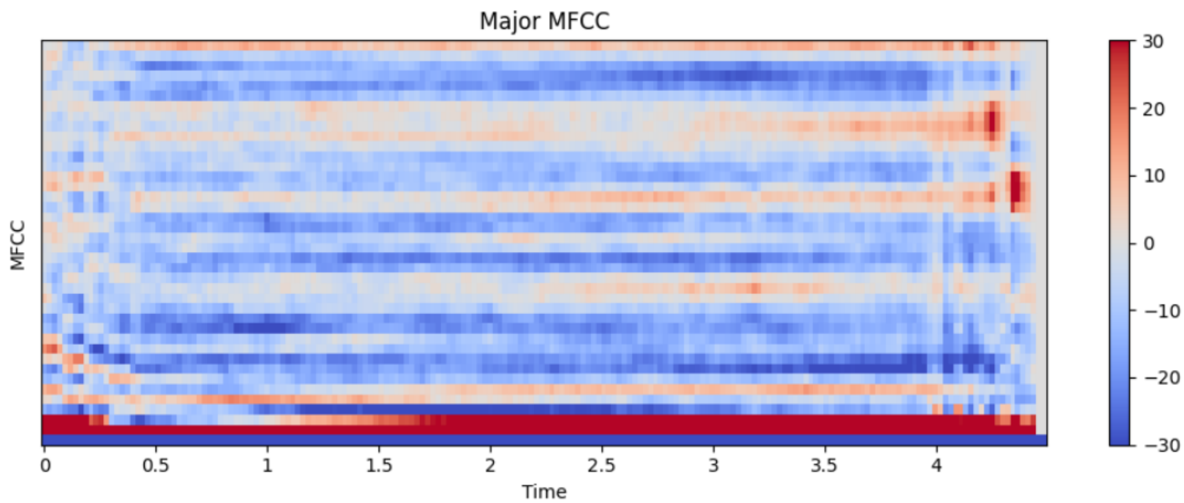


- Mel_Spectrogram

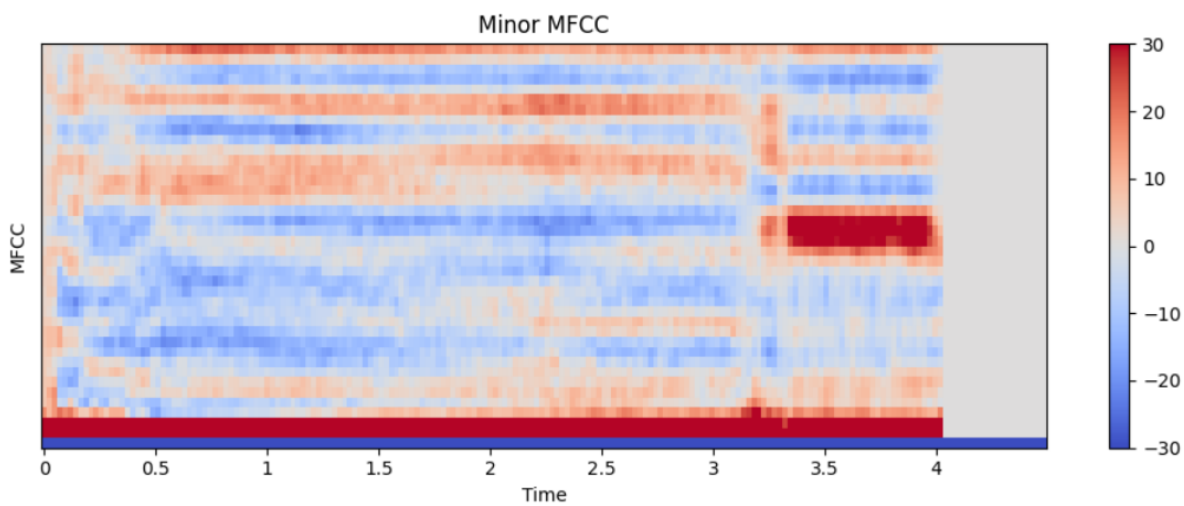

Major Mel spectrogram
4096
2048
1024
512
0
Mels
0
0.5
1
1.5
2
2.5
3
3.5
4
Time
+20 dB
+15 dB
+10 dB
+5 dB
+0 dB
-5 dB
-10 dB
-15 dB
-20 dB
Amplitude


Minor Mel spectrogram
4096
2048
1024
512
0
Mels
0
0.5
1
1.5
2
2.5
3
3.5
4
Time
+20 dB
+15 dB
+10 dB
+5 dB
+0 dB
-5 dB
-10 dB
-15 dB
-20 dB
Amplitude

- MFCC

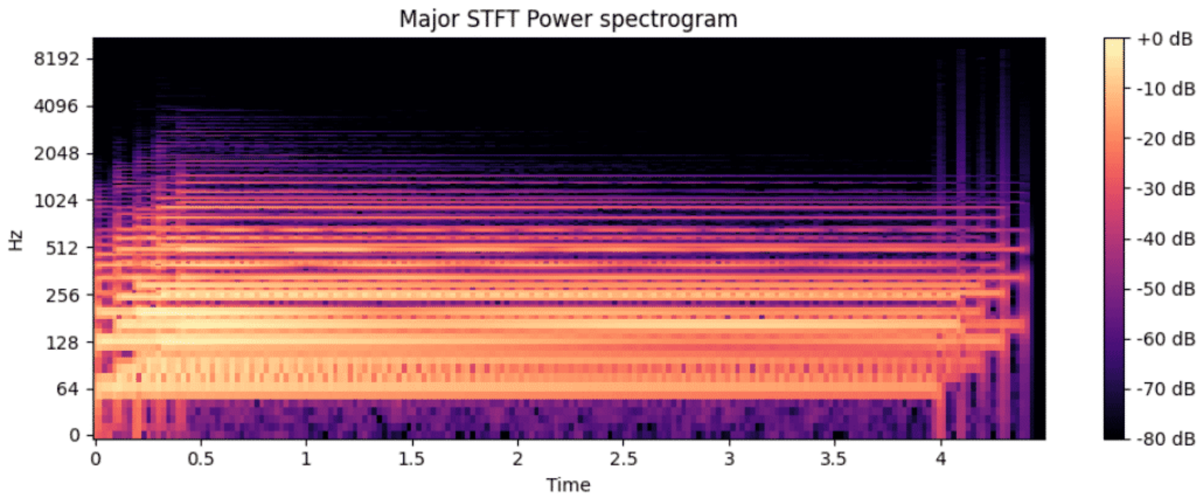


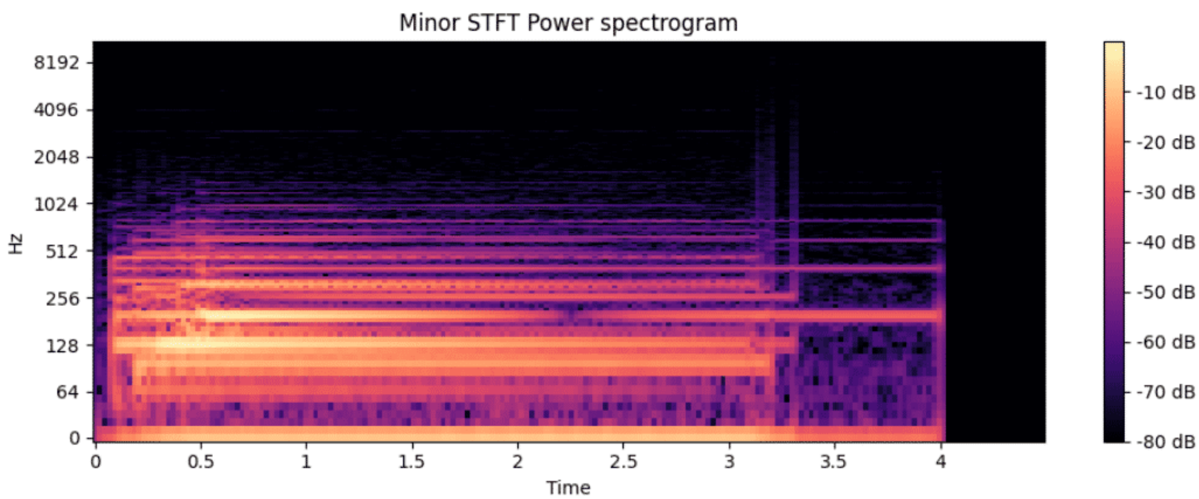
Major MFCC
MFCC
Time
0
0.5
1
1.5
2
2.5
3
3.5
4
30
20
10
0
−10
−20
−30



Minor MFCC
MFCC
Time
0
0.5
1
1.5
2
2.5
3
3.5
4
30
20
10
0
−10
−20
−30

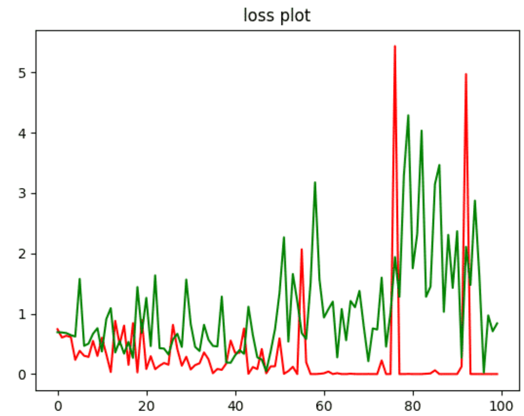
- STFT

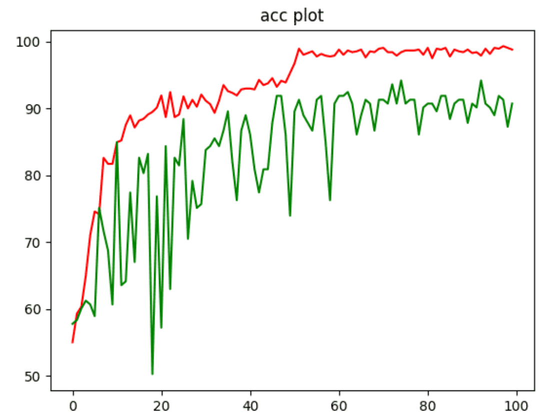




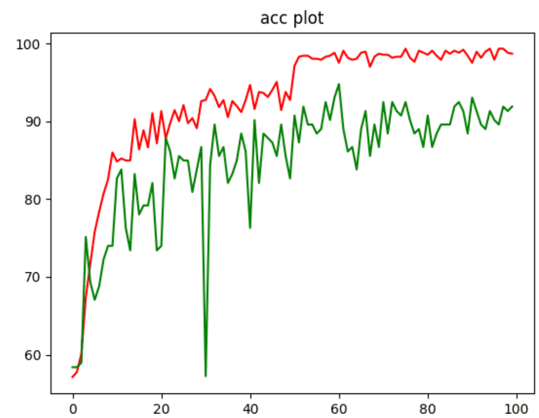
结果比较

Mel_Spectrogram

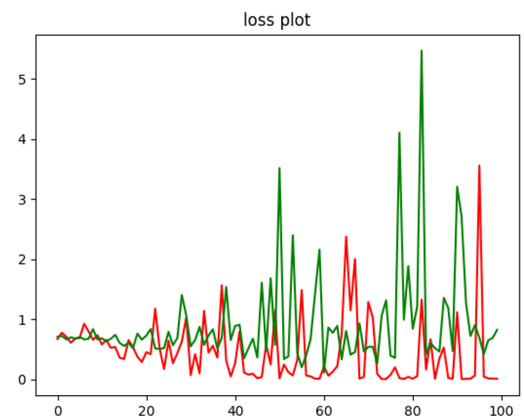


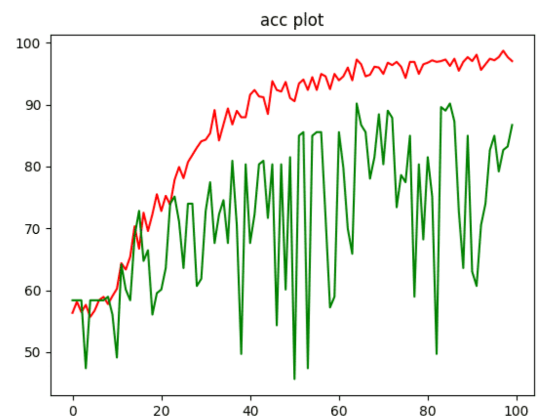


LFCC

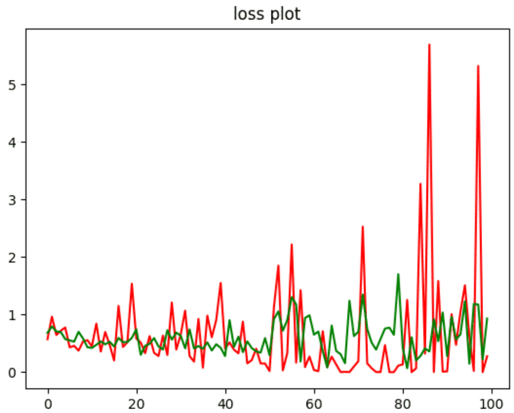


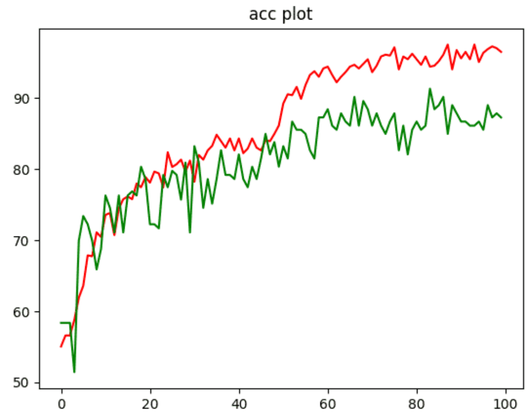
MFCC





Spectrogram





机器学习做法

> 如果用机器学习，我们可以同时将Mel_Spectrogram, MFCC, STFT, chromagram四种特征同时提取出来做一个表格来作为特征，再用机器学习的多层感知机去实现和深度学习一样的功能.

- 提取特征

用librosa库里的函数, 我们可以和torchaudio一样提取到特征

In [8]:
```
def get_features(file):
    # 提取特征函数
     with soundfile.SoundFile(file) as audio:
        waveform = audio.read(dtype="float32")
        sample_rate = audio.samplerate
        # 得到特征
        chromagram = feature_chromagram(waveform, sample_rate)
        melspectrogram = feature_melspectrogram(waveform, sample_rate)
        mfc_coefficients = feature_mfcc(waveform, sample_rate)

        feature_matrix=np.array([])
        # 用hstack来堆叠数据
        feature_matrix = np.hstack((chromagram, melspectrogram, mfc_coefficients))

        return feature_matrix
```

提取到特征以后我们可以将其组合到一个表格中

In 27

```
feature_df = pd.DataFrame(features)
feature_df
```

Out 27

11 rows 859 rows × 180 columns

	0	1	2	3	4	5	6	7	8	9	...	170	171	172	173	174	175
0	0.924185	0.566611	0.563148	0.524787	0.833386	0.526825	0.319134	0.707793	0.432539	0.197384	...	3.148947	1.753475	1.575011	-0.238072	-10.714950	-16.731562
1	0.753709	0.549417	0.485294	0.444800	0.763372	0.552030	0.378476	0.511528	0.431547	0.296792	...	0.327301	-0.532426	0.851971	3.164714	2.816711	0.977635
2	0.465204	0.493697	0.355373	0.319128	0.599958	0.638060	0.682794	0.469525	0.274859	0.507984	...	-13.742491	-11.423026	-5.474628	4.675347	8.974694	5.322316
3	0.551677	0.944659	0.824853	0.645967	0.514380	0.676053	0.408774	0.572577	0.778662	0.413241	...	7.368774	3.836713	-2.565101	-7.358634	-13.468405	-14.565556
4	0.487197	0.697362	0.627283	0.658380	0.555918	0.631397	0.595703	0.650935	0.478512	0.362763	...	7.557004	2.375077	-2.622767	-5.221120	-6.796598	-6.360038
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
854	0.408793	0.379007	0.496257	0.547768	0.710774	0.527925	0.586890	0.679098	0.706606	0.590805	...	-2.735215	0.035742	3.881609	6.081942	4.124215	0.171305
855	0.614545	0.938543	0.785523	0.441923	0.344460	0.507025	0.675125	0.618293	0.772003	0.566146	...	13.882571	8.055911	-4.439617	-11.116182	-12.908711	-10.641177
856	0.598626	0.638431	0.539260	0.348475	0.267030	0.363734	0.660357	0.647699	0.677750	0.565003	...	8.004548	1.764342	-1.705480	-1.702653	1.133152	3.015473
857	0.320032	0.153663	0.323911	0.483517	0.821091	0.610079	0.342374	0.541358	0.379161	0.465834	...	-13.993570	-12.849380	-2.431386	12.852025	17.460995	8.705842
858	0.308530	0.174627	0.291490	0.456683	0.800791	0.559830	0.453053	0.755300	0.527458	0.339609	...	-7.672117	-6.325659	0.019569	7.312313	8.835259	4.042197

有**12 chromagram特征 + 128 mel spectrogram特征 + 40 MFC coefficients特征** 总共180个columns, sklearn调用分类器的包很方便，所以我们可以先用机器学习的方法做一下分类任务

In 29

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

classification_models = [
    KNeighborsClassifier(),
    SVC(kernel='linear'),
    SVC(kernel='rbf'),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    AdaBoostClassifier(),
    GaussianNB(),
    QuadraticDiscriminantAnalysis()]

scores = []
for model in classification_models:
    model.fit(X_train, y_train)
    score = model.score(X_test, y_test)
    model_name = type(model).__name__
    if model_name=='SVC' and model.kernel=='rbf': model_name+=' RBF kernel'
    scores.append((model_name,(f'{100*score:.2f}%')))
# Make it pretty
scores_df = pd.DataFrame(scores,columns=['Classifier','Accuracy Score'])
scores_df.sort_values(by='Accuracy Score',axis=0,ascending=False)
```

Out 29

8 rows 8 rows × 2 columns

	Classifier	Accuracy Score
4	RandomForestClassifier	74.42%
7	QuadraticDiscriminantAnalysis	72.09%
3	DecisionTreeClassifier	70.93%
0	KNeighborsClassifier	63.95%
5	AdaBoostClassifier	63.95%
1	SVC	60.47%
2	SVC RBF kernel	55.23%
6	GaussianNB	50.58%

sklearn里也有MLP包，所以我们可以用sklearn搭建一个简易的多层感知机网络

```python
from sklearn.neural_network import MLPClassifier
model = MLPClassifier(
    activation='logistic',
    solver='adam',
    batch_size=256,
    epsilon=1e-08,
    hidden_layer_sizes=(300,),
    learning_rate='adaptive',
    max_iter=1000, # I've found for this task, loss converges at ~1000 iterations
    random_state=69,
)
model.fit(X_train, y_train)

print(f'MLP Model\'s accuracy on training set is {100*model.score(X_train, y_train):.2f}%')
print(f'MLP Model\'s accuracy on test set is {100*model.score(X_test, y_test):.2f}%')
```

```
MLP Model's accuracy on training set is 99.56%
MLP Model's accuracy on test set is 79.07%
```

结论

题目一：机器学习做法时尝试了多种Classifier, 最后也搭了一个多层感知机来进行分类任务，最终的表现(acc<=80%)仍然不如高维特征提取出来的数据分类结果(acc sometimes > 90%)

题目二：控制网络和参数的情况下Mel_Spectrogram的表现更稳定更好，如果我在最后加一个early stopping, acc可以很好的达到90%以上