

MUSIC GENRE PREDICT MODEL

음원 데이터셋을 통해 장르를 예측하는 신경망 모델 만들어보기

경북대학교 전자전기공학부 이은찬
지능시스템설계 Term Project

목차

1. 프로젝트 소개
2. 데이터셋: 고르게 된 이유와 데이터셋 설명
3. 전처리: 음악 데이터 전처리 하는 방법과 csv 파일
4. 모델링: 신경망 모델 구현
5. 결과
6. ...후기?

프로젝트 소개

10개 장르 1000개의 음원 데이터를 제공하는
GTZAN Dataset를 사용하여 이를 분석 및 전처리하고
이를 통해 신경망 모델을 학습시켜보고

모델의 학습 후 검증과 함께

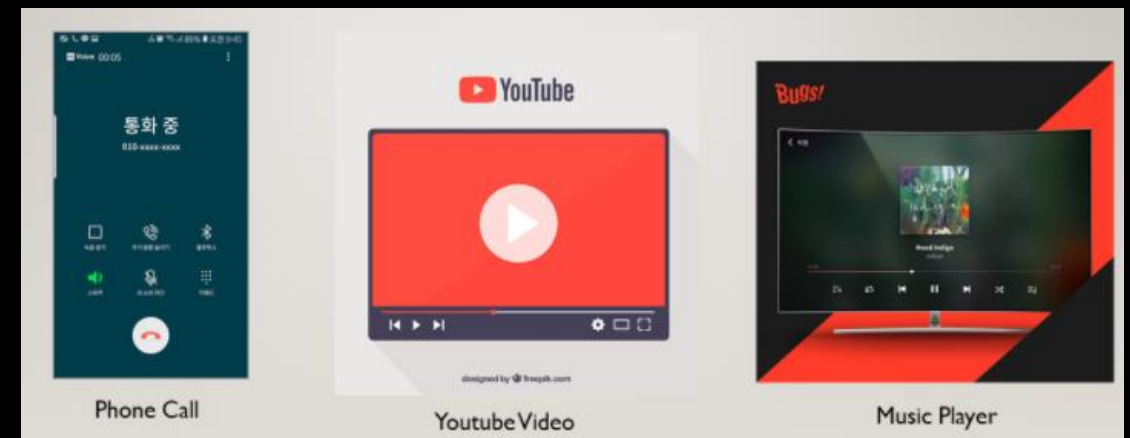
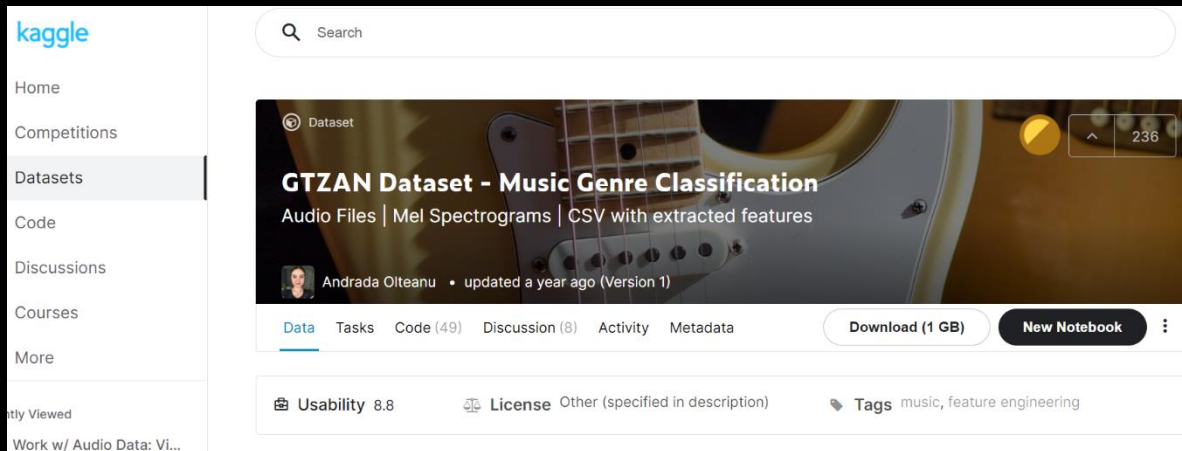
최종적으로 음원 데이터를 보고 장르를 예측하는
다양한 실전 음원 데이터에 대해서도
장르를 예측하는 신경망 모델을 만들어보는 프로젝트



2. 데이터셋

데이터셋: GTZAN DATASET

- GTZAN : MNIST of Audio, famous Audio Dataset! (Kaggle 제공)
- 이 주제를 고르게 된 이유
 - Phone Call, Music Player, Netflix/Youtube Video 등 매우 다양한 곳에서 엄청난 양의 음성 데이터가 각종 디바이스/플랫폼을 통해서 매일 쏟아지고 있습니다!
 - 음성 데이터를 다루는 딥러닝을 공부해보는 경험은 가치가 높을 것 같아서 프로젝트 시작





데이터셋: GTZAN DATASET

Data Explorer

1.31 GB

- ▼ Data
 - ▼ genres_original
 - ▶ blues
 - ▶ classical
 - ▶ country
 - ▶ disco
 - ▶ hiphop
 - ▶ jazz
 - ▶ metal
 - ▶ pop
 - ▶ reggae
 - ▶ rock
 - ▶ images_original
 - features_30_sec.csv
 - features_3_sec.csv

GTZAN Dataset 구성

① genres_original: 10개 장르 각각 100개의(총 1000개) 음원 (.wav)

② Images_original: 위 데이터의 오디오 파형 이미지 (.png)

③ 그 외 2개의 csv 파일 (features_3_sec, features_30_sec)

- 위 데이터에 대한 오디오 데이터의 다양한 특성의 값들을 가지고 있는 엑셀 파일 (1000 개의 데이터 x 60개 특성)

이번에 사용하게 된 데이터 -> ③ features_30_sec [1MB]

①, ②는 용량이 약 1.3GB로 이번 프로젝트와 부합하지 않습니다

GTZAN FEATURES_30_SEC.CSV

10개 음악 장르, 장르별 100개의 데이터(총 1000개),
각각의 데이터는 filename 포함 60가지의 특성값 가짐

특성값은 다양한 오디오 신호의 특징 값들이 존재
mean(평균), var(분산) of Chroma_stft, rms, spectral_centroid.....

	A	B	C	D	E	F	G	H	I	J
1	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var
2	blues.00000.wav	661794	0.35008812	0.088756569	0.1302279	0.002827	1784.16585	129774.0645	2002.44906	85882.76132
3	blues.00001.wav	661794	0.340913594	0.094980255	0.0959478	0.002373	1530.176679	375850.0736	2039.036516	213843.7555
4	blues.00002.wav	661794	0.363637179	0.085275196	0.1755704	0.002746	1552.811865	156467.6434	1747.702312	76254.19226
5	blues.00003.wav	661794	0.404784709	0.093999036	0.141093	0.006346	1070.106615	184355.9424	1596.412872	166441.4948
6	blues.00004.wav	661794	0.308526039	0.087840982	0.0915287	0.002303	1835.004266	343399.9393	1748.172116	88445.20904
7	blues.00005.wav	661794	0.30245629	0.087532379	0.1034936	0.003981	1831.99394	1030482.37	1729.653287	201910.5086

	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	rolloff_mean	rolloff_var	zero_crossing_rate_mean	zero_crossing_rate_var	harmony_mean	harmony_var	perceptr_mean	perceptr_var	tempo	mfcc1_mean	mfcc1_var	mfcc2_mean	mfcc2_var
2	3805.839606	901505.43	0.083044821	0.000766946	-4.53E-05	0.008172282	7.78E-06	0.005698182	123.0469	-113.571	2564.208	121.5718	295.9138
3	3550.522098	2977893.4	0.056039809	0.001447521	0.000139581	0.005099332	-0.000177608	0.003063172	67.99959	-207.502	7764.555	123.9913	560.2599
4	3042.260232	784034.46	0.076291207	0.001006829	2.11E-06	0.016341973	-1.95E-05	0.007457626	161.499	-90.7226	3319.045	140.4463	508.765
5	2184.745799	1493194.4	0.033308863	0.000422759	4.58E-07	0.019054487	-1.45E-05	0.002712198	63.02401	-199.544	5507.517	150.0909	456.5054
6	3579.757627	1572977.8	0.101460539	0.001954125	-1.76E-05	0.004814	-1.01E-05	0.003093899	135.9992	-160.338	5195.292	126.2196	853.7847
7	3481.517592	3274440	0.094041534	0.006233196	1.96E-07	0.008082612	-2.65E-05	0.003242044	69.83742	-177.774	7307.417	118.2055	3195.213
8	2795.610963	1621442	0.073052237	0.001909171	-9.67E-06	0.016923327	-2.24E-05	0.005953942	71.77734	-190.052	9656.535	130.2891	1932.796
9	2954.83676	1629130	0.061442246	0.001848816	1.94E-05	0.013223032	-4.81E-05	0.004838658	92.28516	-179.347	6573.922	136.469	1479.249
10	3782.316288	1262917	0.064024888	0.000731482	2.26E-06	0.012702081	-1.97E-05	0.003465382	83.35433	-121.364	2622.195	122.5067	414.5059

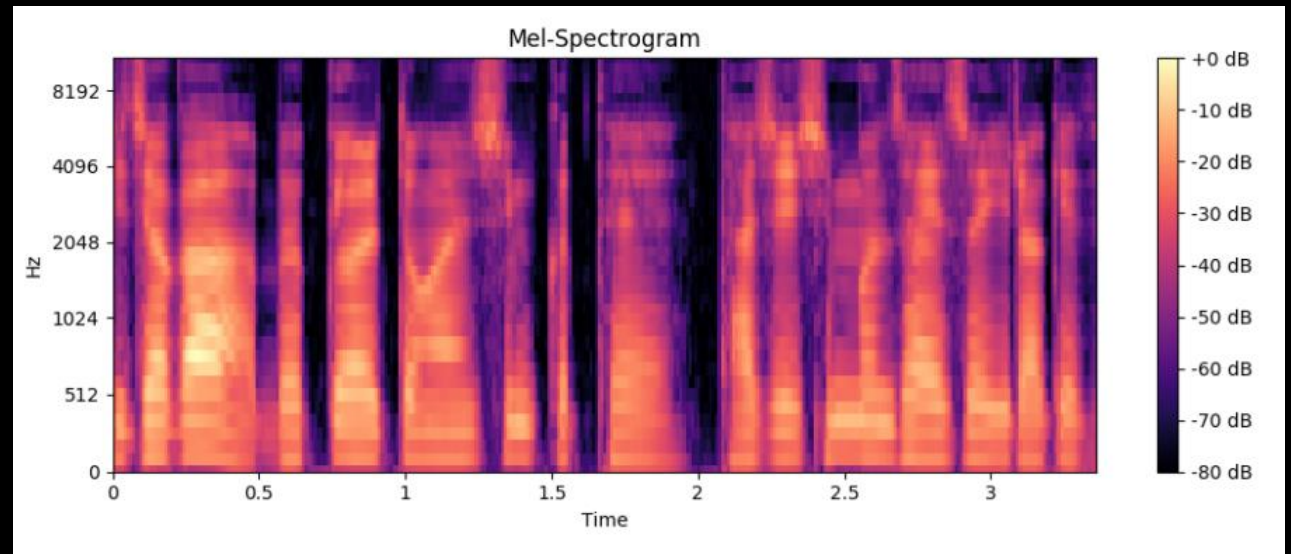
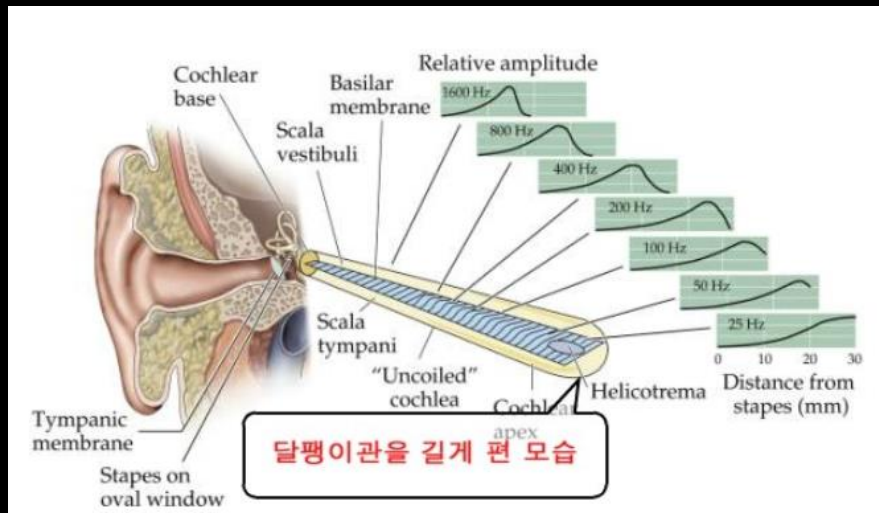


3. 데이터 전처리

음성 신호 전처리: HOW?

[오디오 전처리의 정석] 원래라면 ...

Music file → audio wave → Fourier Transform → Log-scale → Mel-Spectrogram, MFCC



사람의 청각기관 중 달팽이관이 음성 신호를 인지하는 방식과 유사한 원리의 Mel Spectrogram

음성 신호 전처리: PRE- PREPROCESSING된 CSV 파일 사용

- 그러나..

GTZAN Dataset csv 파일은 아래와 같이 이미 전처리 후의 음성 특징들을 제공하고 있으므로 할 필요는 없었습니다.

- csv 파일:

총 60개 데이터 특성 [30개의 features x 2 (평균, 분산)] 제공

- 17개 chroma_STFT, RMS, Spectral-centroid, Spectral-Bandwidth, Rolloff, zero-crossing rate, tempo ...
- 40개 MFCC1~20 데이터
- 3개 : filename, length(모두 30sec으로 동일), genre[=Label]

CSV 데이터 전처리

필요없는 columns drop,
Feature, Label로 분리시키기

```
data = pd.read_csv('data/features_30_sec.csv')  
  
x = data.drop(['filename', 'length', 'label'], axis=1)  
y = data['label'] #Label  
x = x.to_numpy()
```

Label [=genre]
one-hot encoding하기

```
genres = list(set(y))  
t = []  
for i in range(len(y)):  
    one_hot_vector = [0]*10 # [0 0 0 0 0 0 0 0 0 0]  
    one_hot_vector[genres.index(y.loc[i])] = 1  
    t.append(np.array(one_hot_vector))  
t = np.array(t)
```

데이터 정규화 시키기

각각의 특징 값들을 최댓값으로 나누어 모든
값을 0~1사이로 정규화 수행

```
x_column_len = len(x[0,:]) # (1000,57)  
for i in range(x_column_len):  
    x[:,i] = x[:,i]/max(x[:,i])
```

데이터 Train, Test 75:25 분리

```
Test_set = np.random.choice(1000,250,replace=False)  
Train_set = np.delete(np.arange(1000),Test_set)  
np.random.shuffle(Train_set)  
  
x_train = x[Train_set]  
x_test = x[Test_set]  
t_train = t[Train_set]  
t_test = t[Test_set]
```

전처리 결과

X = 정규화된 Features data [1000 x 57]

	0	1	2	3	4	5	6	7
0	0.527492	0.820973	0.327228	0.102122	0.40227	0.0427332	0.570556	0.123611
1	0.513668	0.87854	0.241091	0.085722	0.345004	0.123763	0.580981	0.307784
2	0.547907	0.788771	0.441161	0.099204	0.350107	0.0515231	0.497971	0.109752
3	0.609905	0.869464	0.354529	0.22928	0.241273	0.0607064	0.454864	0.239558
4	0.464868	0.812504	0.229987	0.0832168	0.413732	0.113078	0.498105	0.127299
5	0.455723	0.809649	0.260052	0.143807	0.413054	0.339327	0.492828	0.290609
6	0.438955	0.8693	0.356492	0.31804	0.329039	0.144182	0.395769	0.266303
7	0.464008	0.859325	0.331234	0.199828	0.327303	0.148038	0.44941	0.242107
8	0.616075	0.800216	0.357854	0.0544503	0.387661	0.0537673	0.578902	0.151907
9	0.412772	0.853896	0.204319	0.157043	0.409707	0.098206	0.562385	0.16418

```
In [5]: x.shape  
Out[5]: (1000, 57)
```

```
In [6]: y.shape  
Out[6]: (1000,)
```

Y = output data [= genre]
t = One-hot vector of Y [1000 x 10]

```
In [12]: t.shape  
Out[12]: (1000, 10)
```

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0	0	0

```
In [10]: y[1]  
Out[10]: 'blues'
```

475	1	0	0	0	0	0	0	0	0	0
476	1	0	0	0	0	0	0	0	0	0
477	1	0	0	0	0	0	0	0	0	0
478	1	0	0	0	0	0	0	0	0	0
479	1	0	0	0	0	0	0	0	0	0

```
In [11]: y[475]  
Out[11]: 'hiphop'
```



4. 모델링: 신경망 모델

모델링 : 신경망 모델

수행해야 할 큰 그림

Audio Features → 신경망 → Genre

Input
 $X: (1000, 57) \rightarrow$ output
 $Y: (1000, 1)$
Audio Features \rightarrow Genre
[57개] [10 중 1]

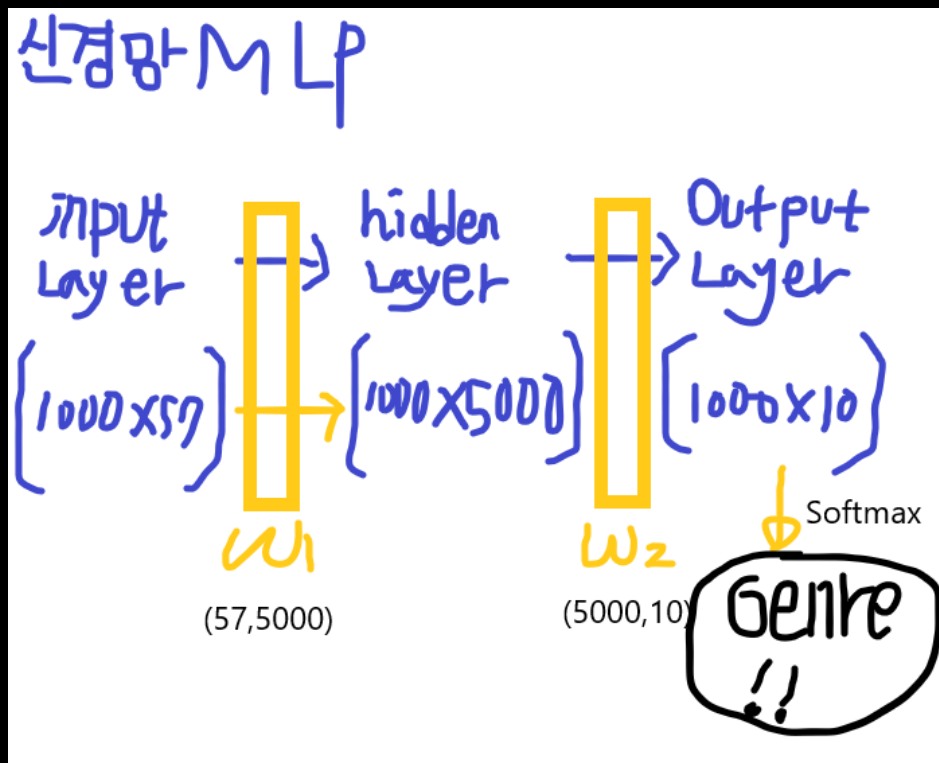
이 역할을 가장 잘 수행할 수 있는 모델은?

→ Multi Label Classification Simple MLP Model

모델링

Data Flow Diagram

INPUT → MLP → Layer OUTPUT → Softmax → Genre



Base Model : 교재 MLP 모델 응용
[TwoLayerNet class]

Structure: FCL → Sigmoid → FCL → Softmax

Size of Layers: $i=57$, $h=5000$, $o=10$

Loss Function : Cross-Entropy

Optimizer : Adam

Training : 교재 Training 코드 응용 [Trainer class]
Epochs = 300, Batch size = 32

신경망 모델링 CODE

```
class TwoLayerNet:
```

```
    self.layers = [  
        Affine(W1,b1),  
        Sigmoid(),  
        Affine(W2,b2)]
```

Structure: FCL → Sigmoid → FCL → Softmax

```
    self.loss_layer = SoftmaxWithLoss()
```

Loss Function : Cross-Entropy

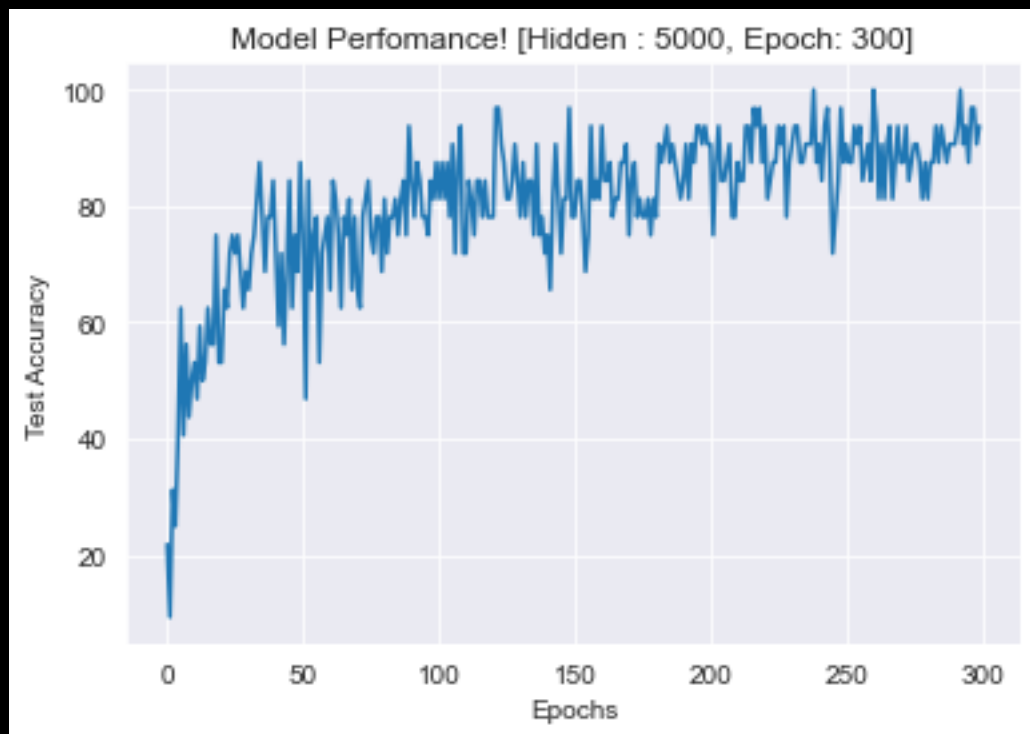
Optimizer : Adam

```
#신경망 설계 [Neuralnet Modeling]      Size of Layers: i=57, h=5000, o=10  
model = TwoLayerNet(input_size=57, hidden_size =5000, output_size =10)  
  
optimizer = Adam()  
  
#교재 제공 Trainer Class를 사용하여 Model Training 수행  
GTZAN_MLP = Trainer(model, optimizer)  
GTZAN_MLP.fit(x,t, max_epoch=300, batch_size=32)
```



5. 테스트 및 결과

TRAINING 결과



대체적으로 85~100%의 높은 정확도를 뽑아주는 것으로 확인할 수 있었다.


에폭	1	반복	1 / 31	시간	0[s]	손실	2.32	정확도	21.875
에폭	2	반복	1 / 31	시간	0[s]	손실	2.67	정확도	9.375
에폭	3	반복	1 / 31	시간	0[s]	손실	2.16	정확도	31.250
에폭	4	반복	1 / 31	시간	1[s]	손실	1.93	정확도	25.000
에폭	5	반복	1 / 31	시간	1[s]	손실	1.81	정확도	40.625

... 중략 ...

에폭	296	반복	1 / 31	시간	135[s]	손실	0.29	정확도	87.500
에폭	297	반복	1 / 31	시간	136[s]	손실	0.26	정확도	96.875
에폭	298	반복	1 / 31	시간	136[s]	손실	0.27	정확도	96.875
에폭	299	반복	1 / 31	시간	137[s]	손실	0.30	정확도	90.625
에폭	300	반복	1 / 31	시간	137[s]	손실	0.29	정확도	93.750

최종 결과 보고

- 30MB 이하의 데이터를 가지고 음원 데이터를 분석하기 위해 음원 데이터와 스펙트로그램 파형 이미지 데이터가 아닌 음원의 특징 값들을 지니는 .csv 정형 데이터로 타겟 데이터를 잡았고 이를 멀티 레이블 분류 MLP 모델을 통해 분석해 보았다.
- 그 결과 뚜렷한 결과는 아니더라도 전체 25%의 TEST Data에 대해서 훈련 결과 최소 85% 이상~100%까지의 Accuracy를 달성할 수 있었다.
- 그러나 위 모델로 실제 음원을 통한 장르를 예측시키고 싶을 때, CSV 파일처럼 음원의 57가지 feature를 직접 featurizing 해 주어야하며 이는 오디오 전문가가 아니고서야 범용성이 0에 가깝다.
- 나중에 기회가 된다면 CNN + 음원 파형 이미지 분석과 음악 데이터 그 자체의 파형만으로 NLP based 모델로 더욱 범용성 높은 모델을 구성해볼 계획이다.



감사합니다!

이번 프로젝트의 더 자세한 내용은 제 깃허브 [@Github/Purang2](https://github.com/Purang2)에서 보실 수 있습니다.