The Project name is Music type classification. Basically, audio processing is kind of similar as image processing and image classification. The project aims to classify the audio files in certain categories of sound to which they belong. As for the dataset, I use the GTZAN dataset from Kaggle. As for the implementation of the project, I downloaded the dataset and first read the data from the dataset. Then preprocessing the data by removing any unrelated things such as labels and if we load the audio files, we can find that they are types of NumPy arrays with float64 object. Also, the packages and APIs from python we used includes Librosa, TensorFlow and Keras.

```
df=pd.read_csv('/Users/hejingze/Desktop/CS448 final_project/Data/features_3_sec.csv')
df.head()
```

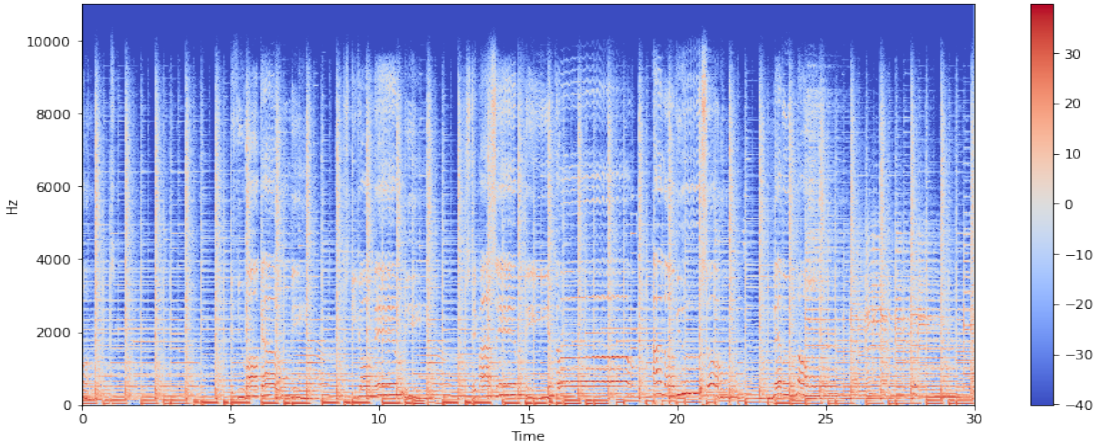
✓ 0.2s Python

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_
0	blues.00000.0.wav	66149	0.335406	0.091048	0.130405	0.003521	1773.065032	167541.630869	1972.744388	
1	blues.00000.1.wav	66149	0.343065	0.086147	0.112699	0.001450	1816.693777	90525.690866	2010.051501	
2	blues.00000.2.wav	66149	0.346815	0.092243	0.132003	0.004620	1788.539719	111407.437613	2084.565132	
3	blues.00000.3.wav	66149	0.363639	0.086856	0.132565	0.002448	1655.289045	111952.284517	1960.039988	
4	blues.00000.4.wav	66149	0.335579	0.088129	0.143289	0.001701	1630.656199	79667.267654	1948.503884	

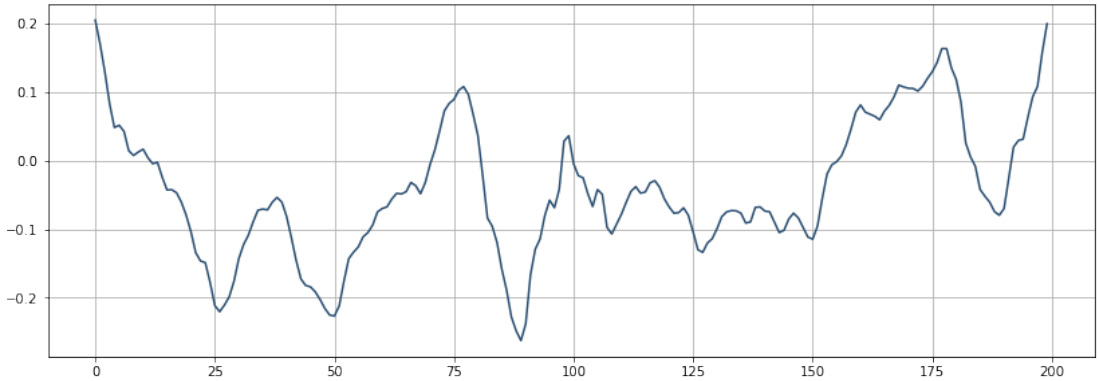
5 rows × 60 columns

Then we load and decodes the audio arrays as time series with 20 kHz and sample rate 45600Hz. The visualization of spectrogram is shown below:

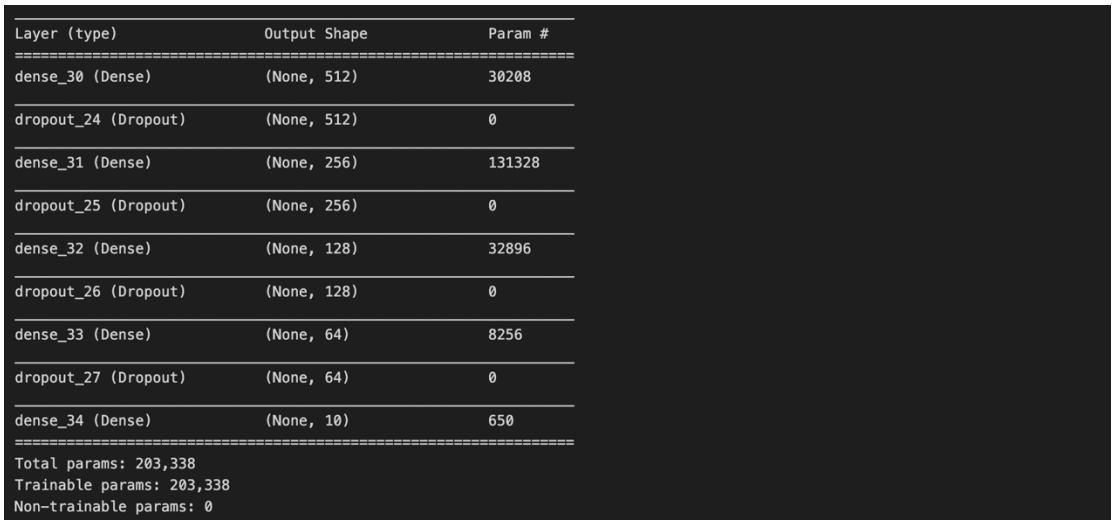
With x-label time per seconds and y-label frequency (Hz).



And the rate at which zero-crossings occur is shown below:



Then extracting the features and building the model, here we use the Convolutional Neural Network since they can learn patterns that are translation invariant and have spatial hierarchies. The overall accuracy by using CNN is higher and it is an efficient tool in classifying music types. Therefore, we build and train the model by using Adam optimizer with over 500 epochs. Finally, evaluating the model gives us a test loss of 0.6356 and the best accuracy of 91.932%.



Layer (type)	Output Shape	Param #
dense_30 (Dense)	(None, 512)	30208
dropout_24 (Dropout)	(None, 512)	0
dense_31 (Dense)	(None, 256)	131328
dropout_25 (Dropout)	(None, 256)	0
dense_32 (Dense)	(None, 128)	32896
dropout_26 (Dropout)	(None, 128)	0
dense_33 (Dense)	(None, 64)	8256
dropout_27 (Dropout)	(None, 64)	0
dense_34 (Dense)	(None, 10)	650

Total params: 203,338
Trainable params: 203,338
Non-trainable params: 0