음악과 딥러닝의 만남

도승헌
KAIST GSCT | Music and Audio Computing Lab

Github : https://github.com/dohppak
E-mail : seungheondoh@kaist.ac.kr

오늘 튜토리얼의 내용

CONNECT THE PYTHONISTAS

- Sound Data의 이해와 전처리와 시각화
  - Signal Processing
  - Sound Visualization
- Music-Deep Learning 의 주요 Task Review
  - 학계에서 주요하게 무엇이 사용되는가?
  - 산업에서는 어떻게 쓰인가?
  - What is MIR?
- Music Gerne Classification 실습!
  - CNN Architecture 구현!
  - 일단 데이터셋부터 받자

오늘 발표에 중점인 부분

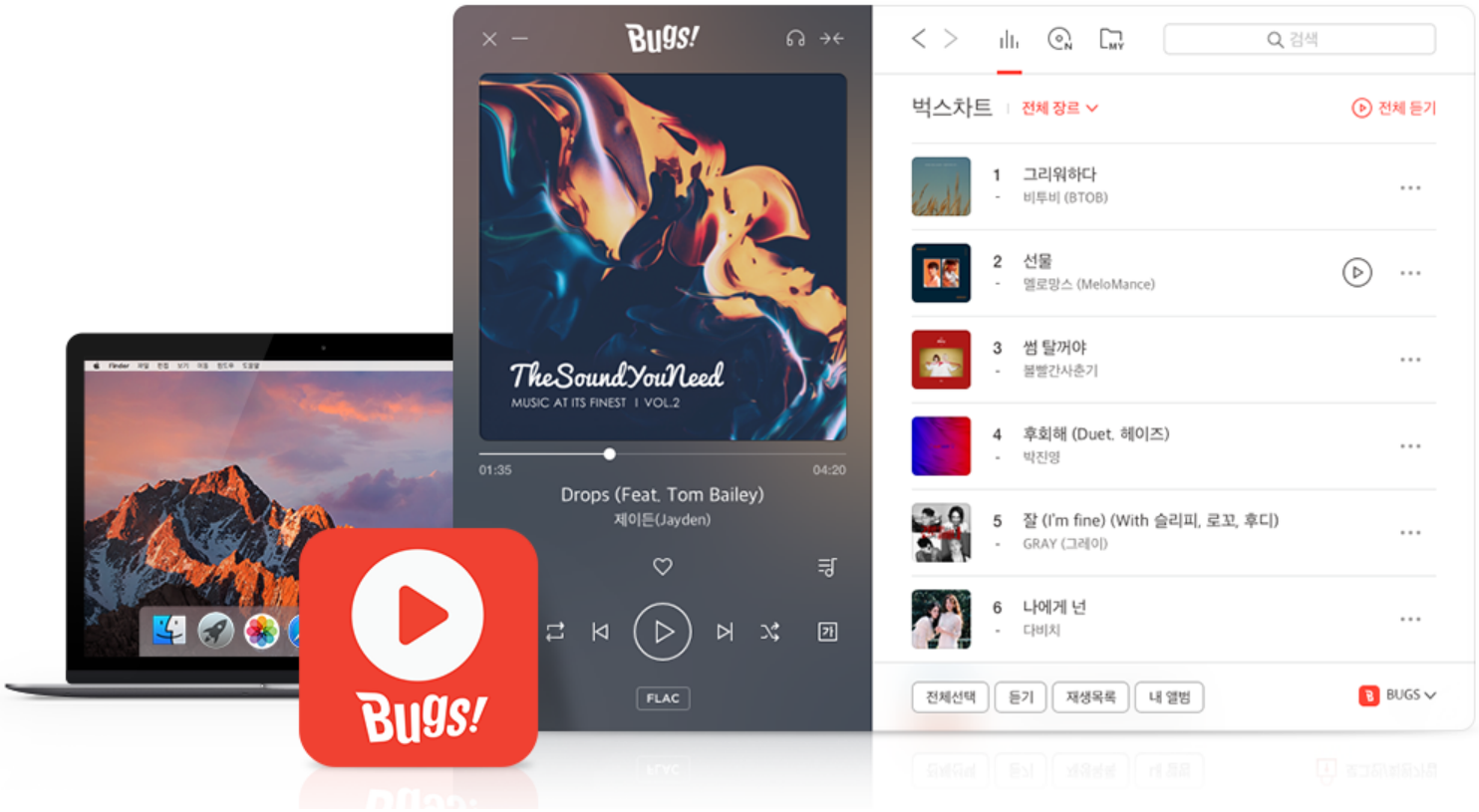
CONNECT THE PYTHONISTAS

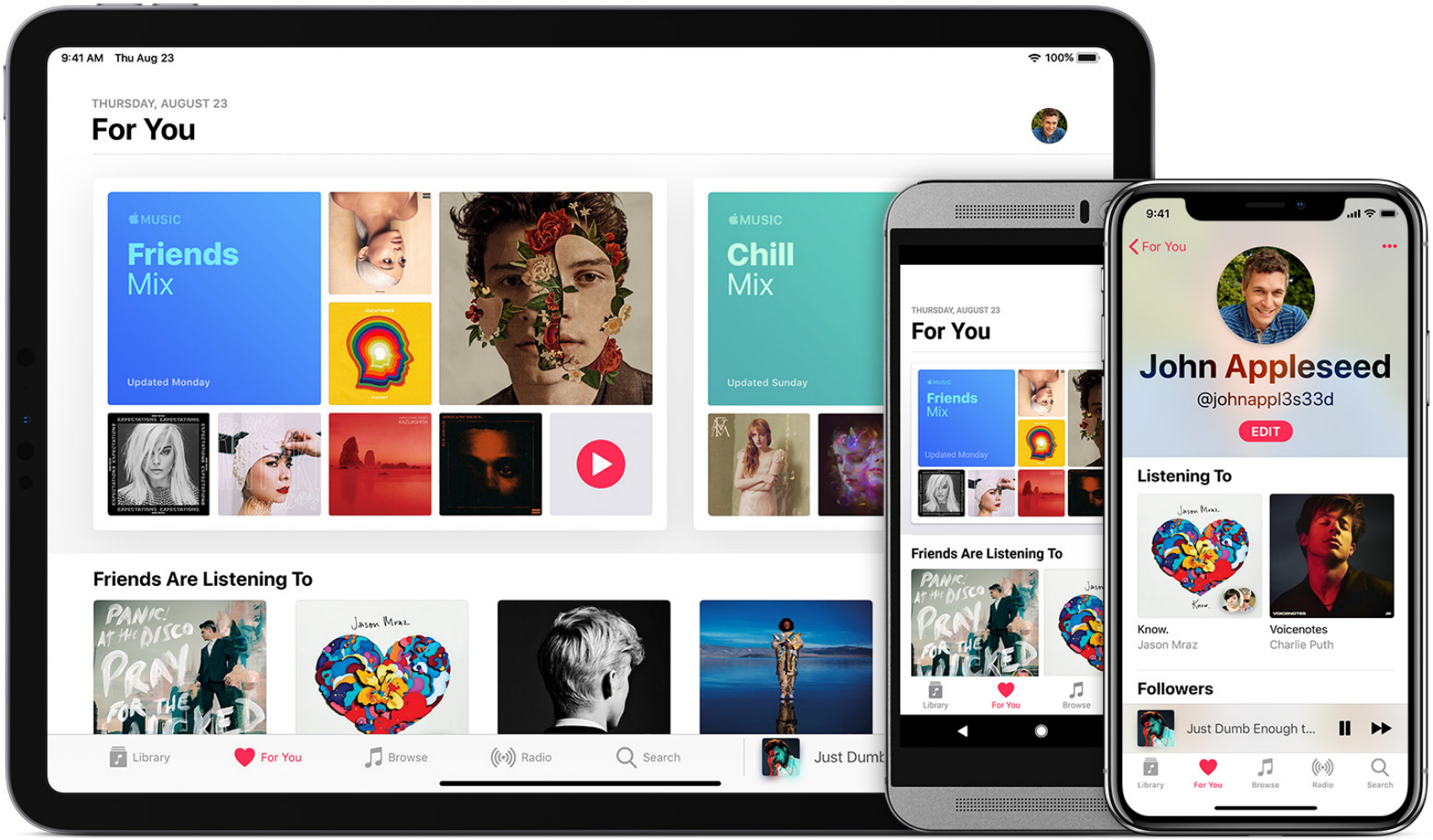
- 튜토리얼 답게 실습을 하자!
- Import 해서 한 줄이면 되는 코드를 실제로 타이핑해본다!
- Notation을 하나씩 분석해보면서 우리가 생각없이 호출하던 함수들의 파라미터를 정확히 이해해보자!
- 딥러닝이 해결해주는 Task를 살펴본다

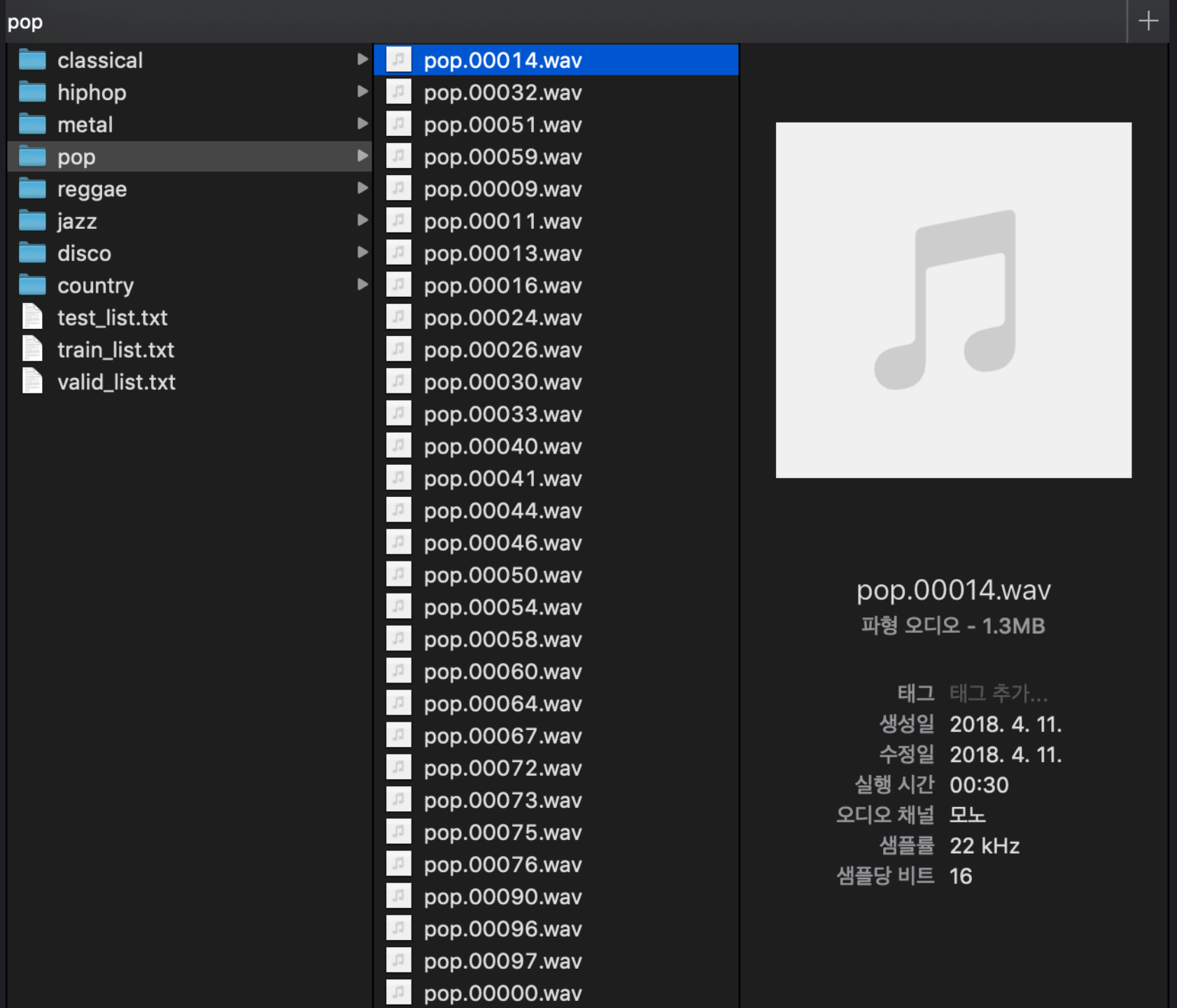
다들 어떻게 음악을 들으시나요?

Mac에서도 벅스

국내 음악 서비스 최초 Mac용 음악 플레이어



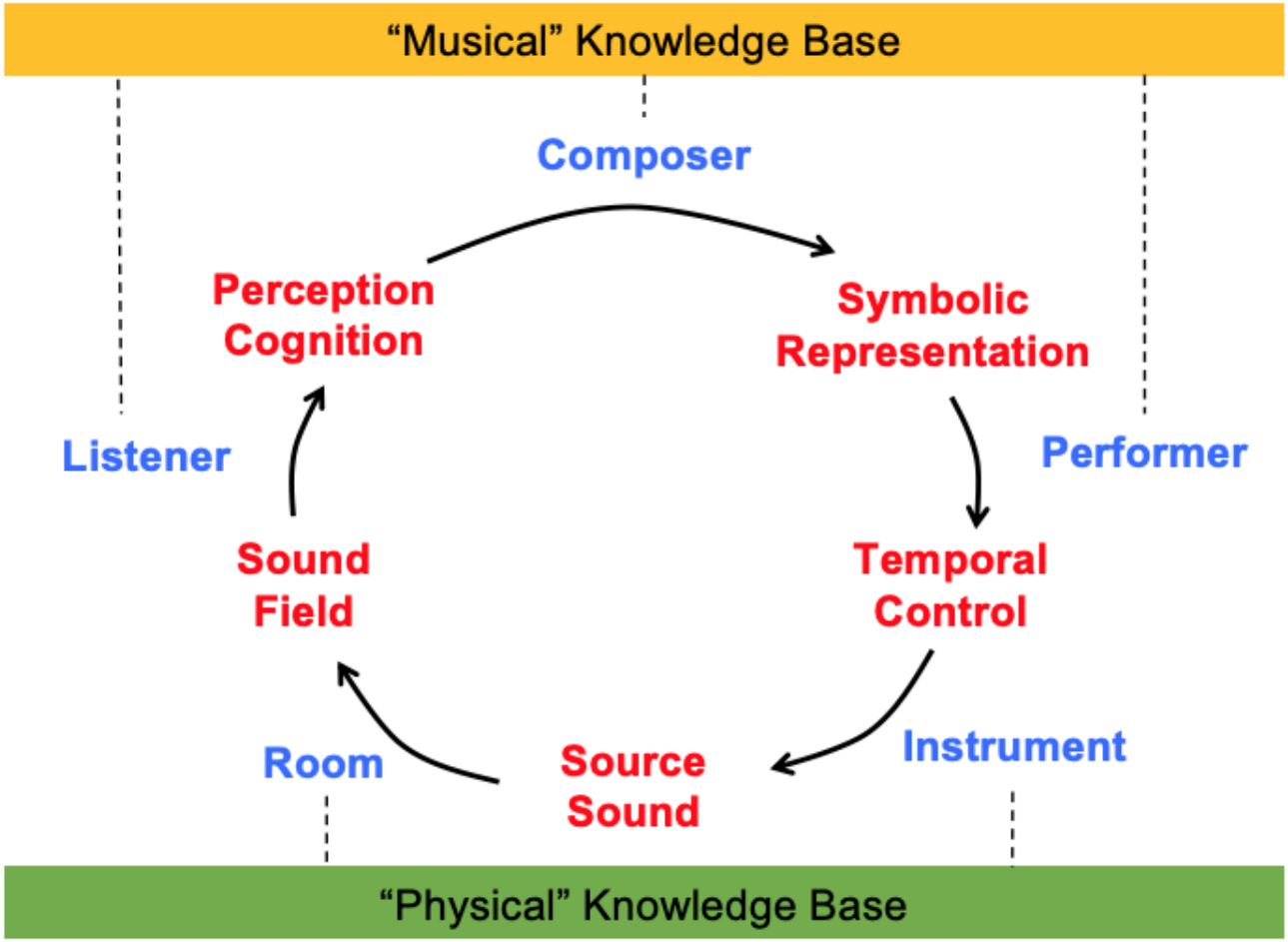

9:41 AM Thu Aug 23
100%
THURSDAY, AUGUST 23
For You
MUSIC
Friends
Mix
Updated Monday
MUSIC
Chill
Mix
Updated Sunday
Friends Are Listening To
Library
For You
Browse
Radio
Search
Just Dumb
9:41
For You
John Appleseed
@johnappl3s33d
EDIT
Listening To
Know.
Jason Mraz
Voicenotes
Charlie Puth
Followers
Just Dumb Enough t...


pop
classical
hiphop
metal
pop
reggae
jazz
disco
country
test_list.txt
train_list.txt
valid_list.txt
pop.00014.wav
pop.00032.wav
pop.00051.wav
pop.00059.wav
pop.00009.wav
pop.00011.wav
pop.00013.wav
pop.00016.wav
pop.00024.wav
pop.00026.wav
pop.00030.wav
pop.00033.wav
pop.00040.wav
pop.00041.wav
pop.00044.wav
pop.00046.wav
pop.00050.wav
pop.00054.wav
pop.00058.wav
pop.00060.wav
pop.00064.wav
pop.00067.wav
pop.00072.wav
pop.00073.wav
pop.00075.wav
pop.00076.wav
pop.00090.wav
pop.00096.wav
pop.00097.wav
pop.00000.wav
pop.00014.wav
파형 오디오 - 1.3MB
태그 태그 추가...
생성일 2018. 4. 11.
수정일 2018. 4. 11.
실행 시간 00:30
오디오 채널 모노
샘플률 22 kHz
샘플당 비트 16

음악은 어떻게 만들어지는 걸까요?

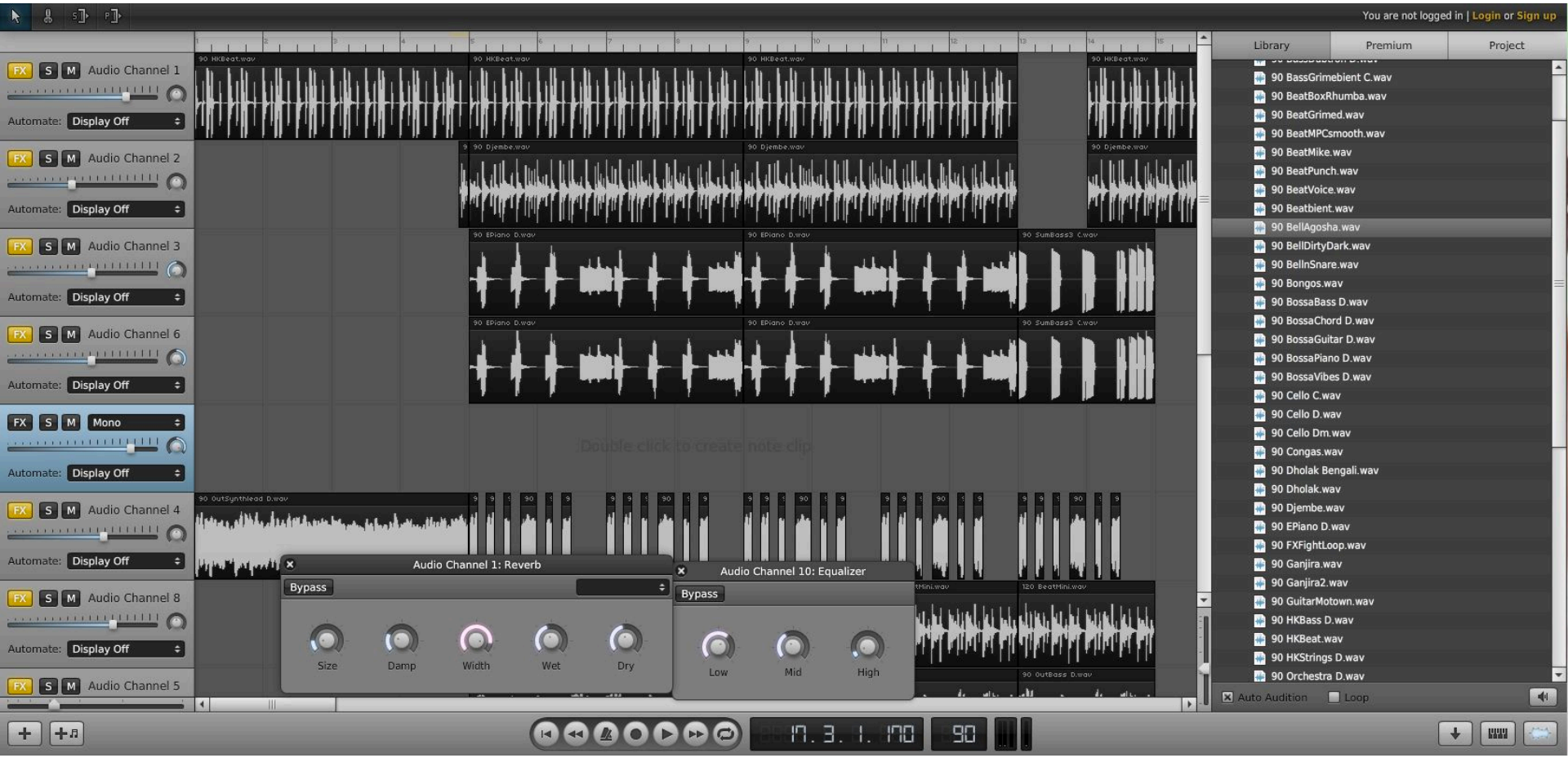
Music Data Process

음악은 Composer, Performer, Instrument, Room(Recoding)을 지나서 **Listener** 에게 전달됩니다.



Music Data Process

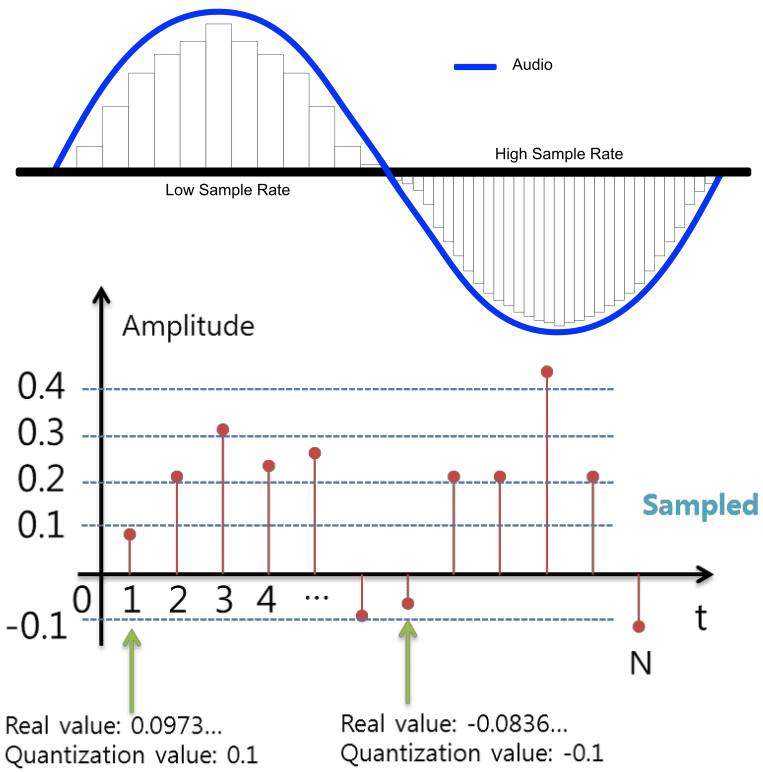
그리고 음성과 다르게 다양한 악기들이 **Layer**처럼 쌓여서 만들어지는 조금 더 복잡한 소리의 형태이죠



Computer가 소리를 이해하는 과정을 살펴봅시다.

Computer가 소리를 인식하는 방식

연속적인 아날로그 신호를 **표본화(Sampling), 양자화(Quantizing), 부호화 (Encoding)을** 거쳐 이진 **디지털 신호(Binary Digital Signal)**로 변화시켜서 인식하게 됩니다.



http://www.ktword.co.kr/abbr_view.php?m_temp1=978

샘플링 단계에서 초당 샘플링 횟수를 정하는데, 이를 Sampling rate라고 합니다.

1초의 연속적인 시그널을 몇개의 숫자로 표현 할 것인가?

양자화 단계에서는 amplitude의 real valued 를 기준으로 시그널의 값을 조절합니다

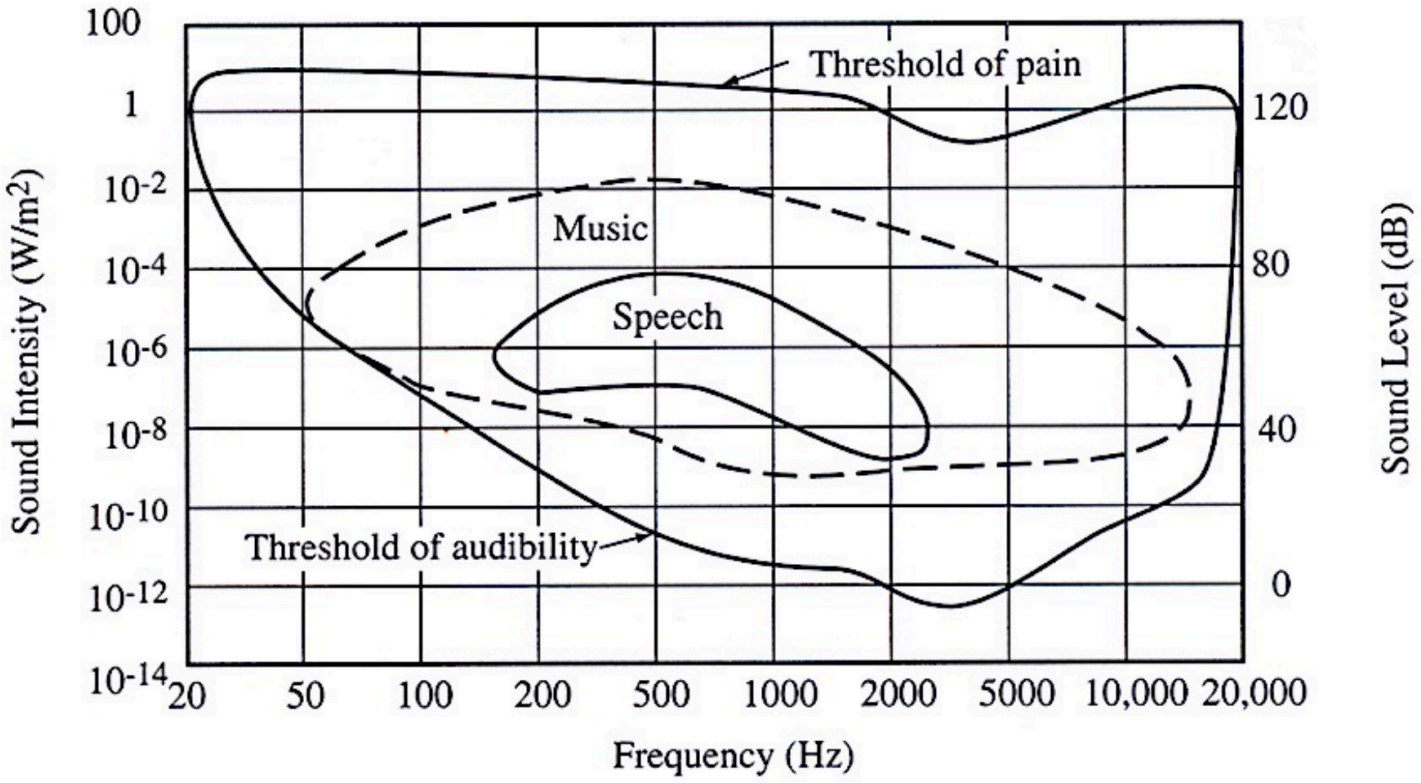
Amplitude를 이산적인 구간으로 나누고, signal 데이터의 Amplitude를 반올림하게 됩니다.

- B bit의 Quantization : $-2^{B-1} \sim 2^{B-1} - 1$
- Audio CD의 Quantization (16 bits) : $-2^{15} \sim 2^{15} - 1$
- 위 값들은 보통 -1.0 ~ 1.0 영역으로 scaling되기도 합니다.

Sampling Rate

그렇다면 어떻게 **Sampling Rate**를 설정할 것인가?

나이키스트-섀넌 표본화 에 따르면 A/D를 거치고, D/A로 복원하기 위해서는 표본화 된 신호의 최대 주파수가 두배 보다 더 클 때 가능하다고 합니다.
일반적으로 사용되는 주파수 영역대는 16KHz(Speech), 22.05KHz 44.1KHz (Music)입니다.



Tutorial

CONNECT THE PYTHONISTAS

scipy.io의 librosa.load 함수의 경우, 파일이름을 인자로 받아서 sample rate와 data를 return 해줍니다.

이때의 데이터 타입은 float 32이며 sampling rate의 디폴트 값은 22050입니다.

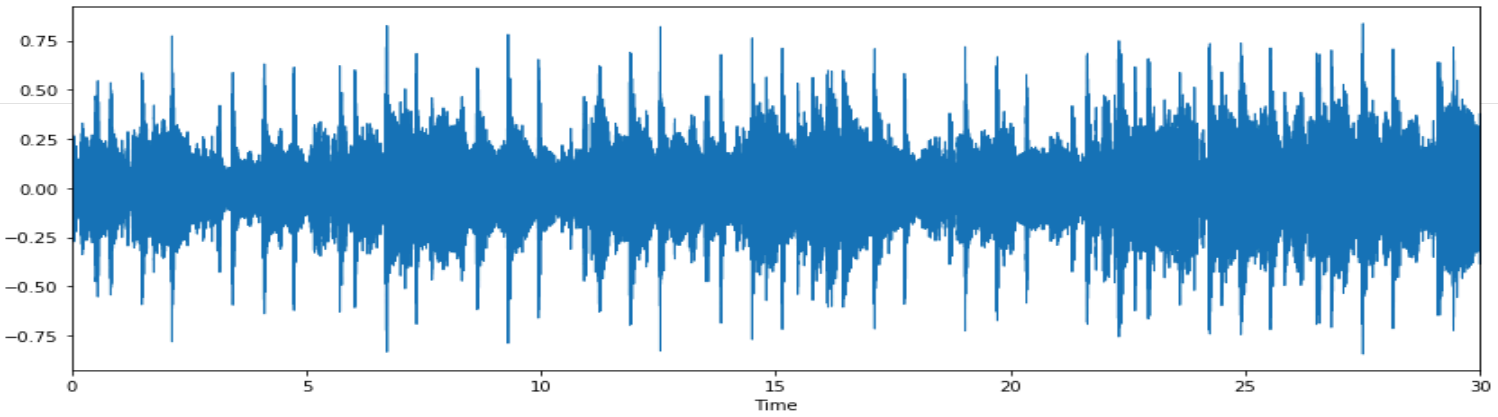
```
import librosa
import librosa.display

train_audio_path = 'input/'
file_path = 'pop.wav'
y, sr = librosa.load(str(train_audio_path)+file_path, sr=22050)
y, sr

(array([-0.0249939 , -0.01815796, -0.02023315, ..., -0.36505127,
        -0.4237976 , -0.42385864], dtype=float32), 22050)
```

Tutorial

CONNECT THE PYTHONISTAS

Waveform. 시간이 x축으로 그리고 amplitude가 y축으로 표현

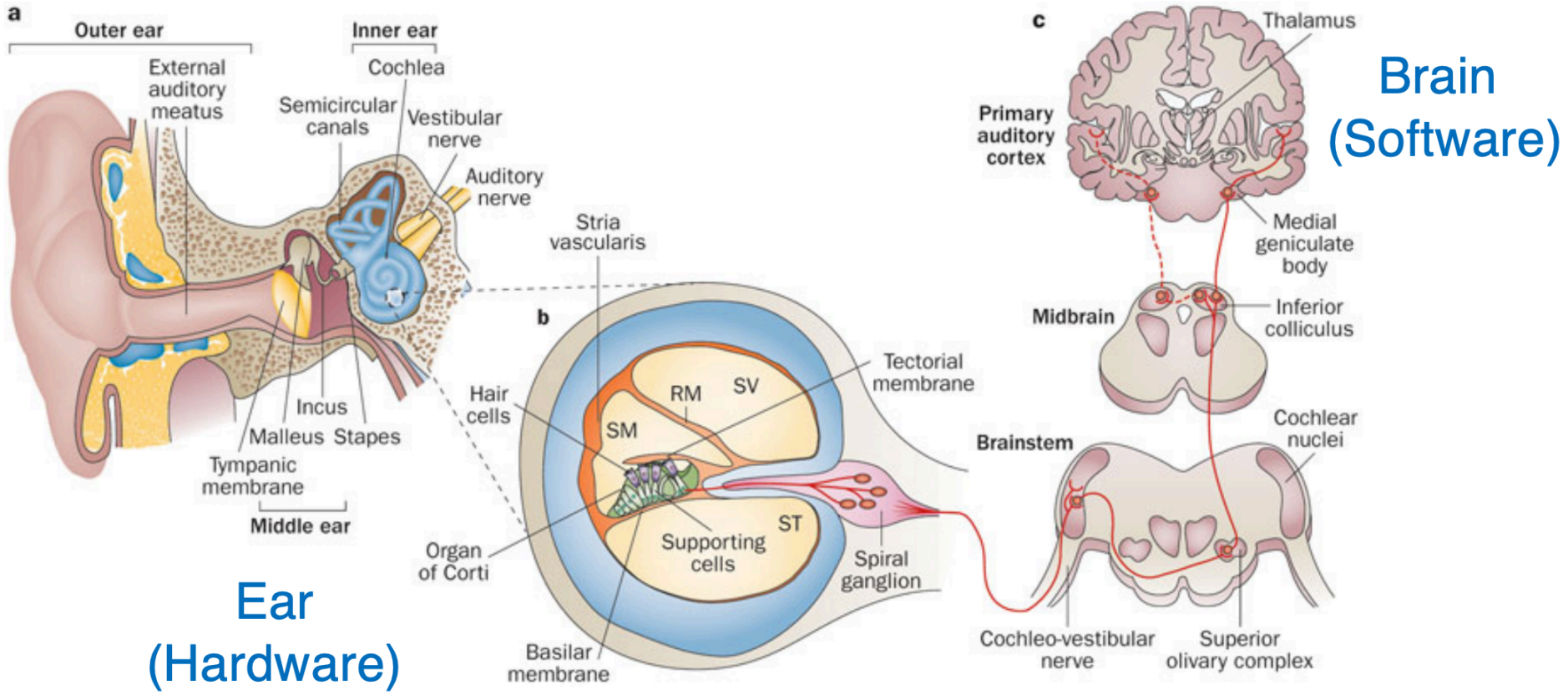
```
# Jupyter에서 듣고 싶다면!
import IPython.display as ipd
ipd.Audio(y, rate=sr)

# 시각화해서 보고싶다면!
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

fig = plt.figure(figsize = (14,5))
librosa.display.waveplot(y, sr=sr)
```



그렇다면 인간은 이러한 소리를 어떻게 인지할까요?

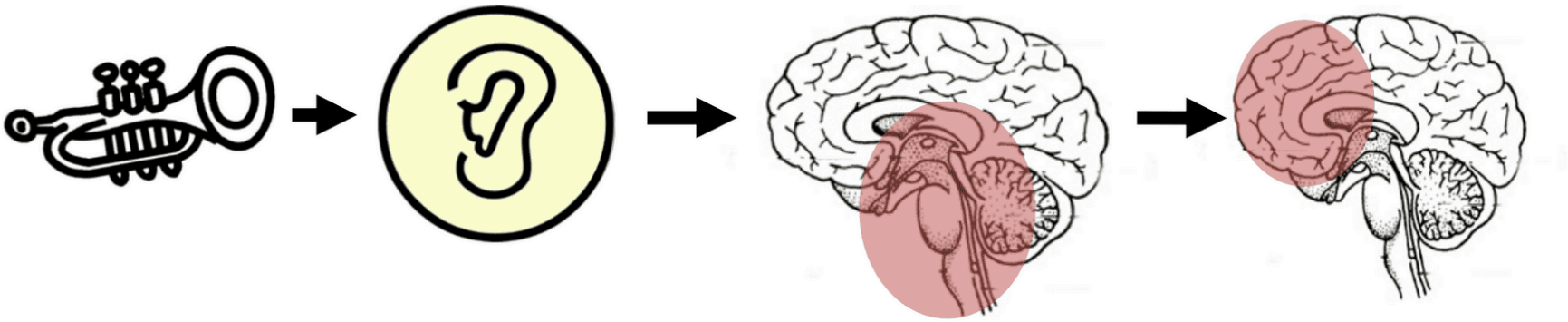
Cognition?

Outer ear - Middle ear - Inner ear - Brain
인간의 소리 인지를 **정보처리**라고 생각한다면



Cognition?

Sensing -> Perception -> Cognition
우리는 **컴퓨터도** 이와 같은 과정을 하길 원한다!



1. Sensation

Sound encoded by ear

2. Perception

Pitch
Timbre
Loudness
Duration

3. Cognition

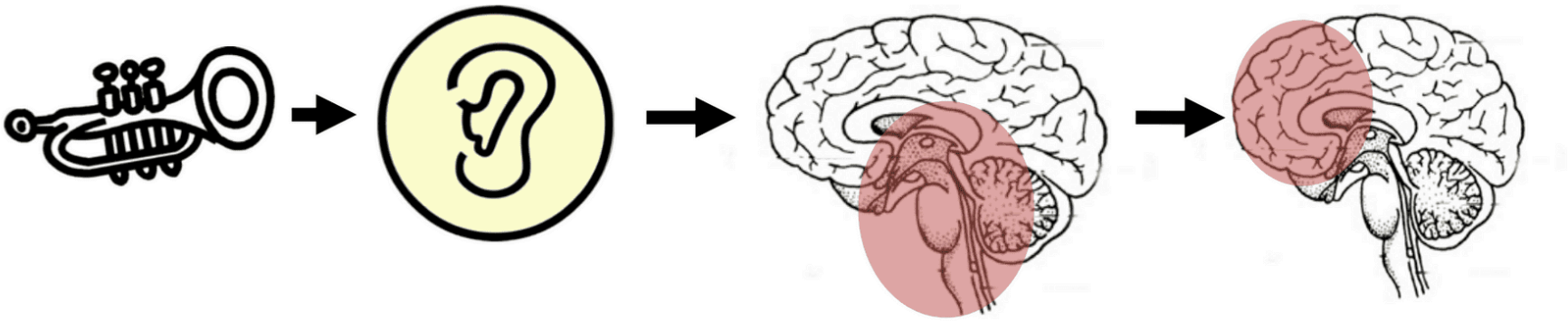
Scale, Tonality Rhythm
Musical Structure
Musical memory
Emotion

Cognition?

마이크
A/D Converter

Signal
Processing

Signal
Processing +
Deep Learning



1. Sensation

Sound encoded by ear

2. Perception

Pitch
Timbre
Loudness
Duration

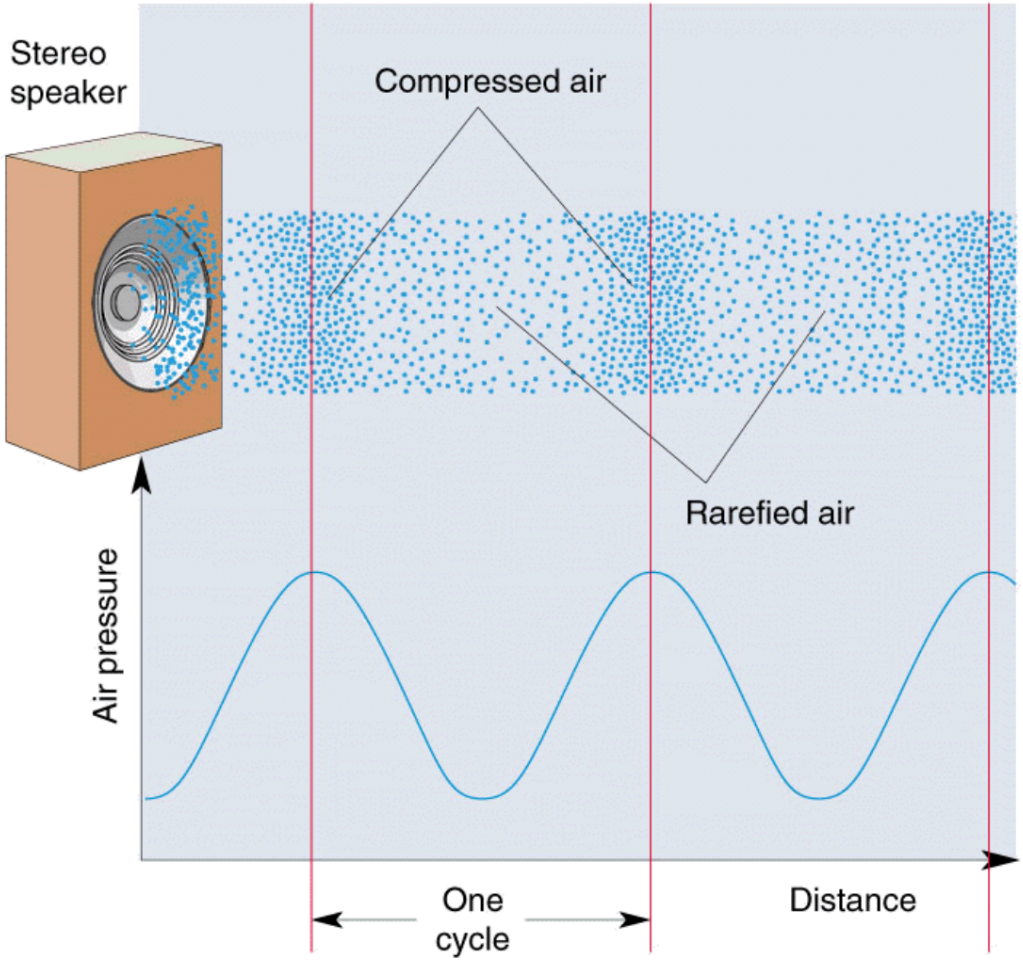
3. Cognition

Scale, Tonality Rhythm
Musical Structure
Musical memory
Emotion

Computer가 Music Cognition을
인간처럼 잘하고 음악에서 다양한
정보를 알려주면 좋겠다.

Sound?

소리 = 진동으로 인한 **공기의 압축**
압축이 얼마나 됐느냐 = Wave(파동)

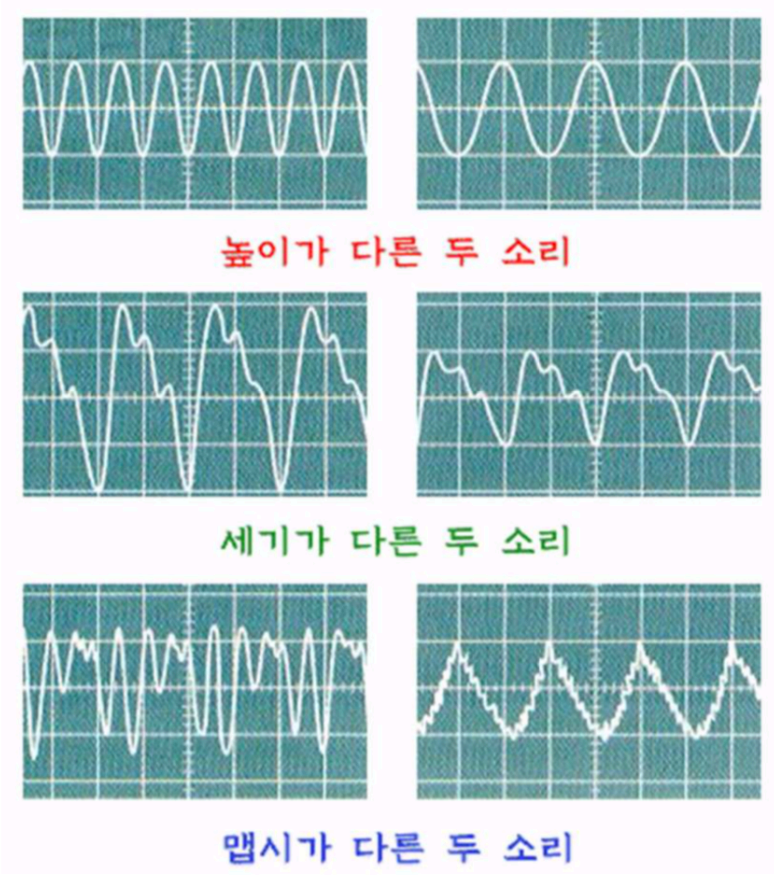


파동은 진동하며 공간/매질을 전파해 나가는 현상입니다.

질량의 이동은 없지만 에너지/운동량의 운반은 존재합니다.

소리에서 얻을 수 있는 물리량

- Amplitude(Intensity) : **진폭**
- Frequency : **주파수**
- Phase(Degree of displacement) : **위상**



물리 음향

- Intensity : 소리 진폭의 세기
- Frequency : 소리 떨림의 빠르기
- Tone-Color : 소리 파동의 모양

심리 음향

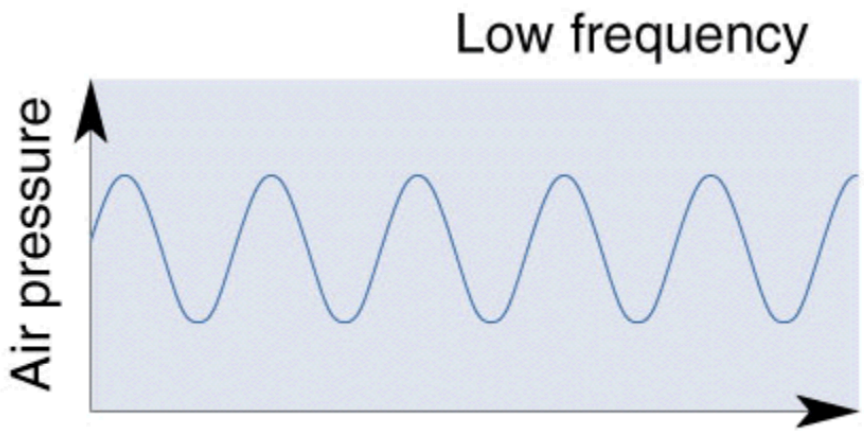
- Loudness : 소리 크기
- Pitch : 음정, 소리의 높낮이 / 진동수
- Timbre : 음색, 소리 감각

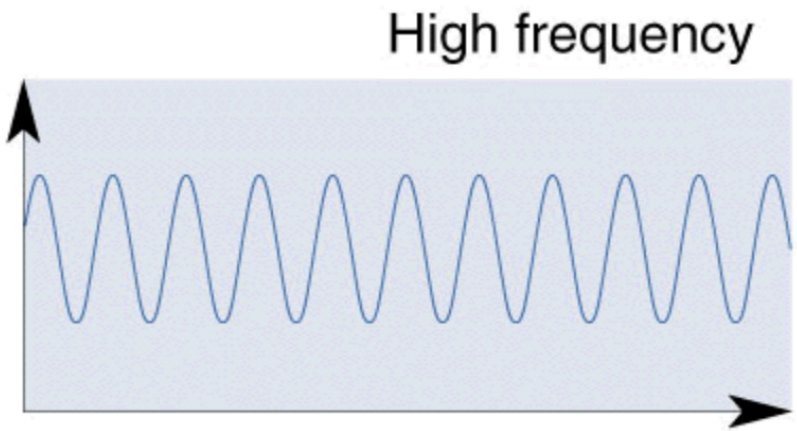
Frequency?

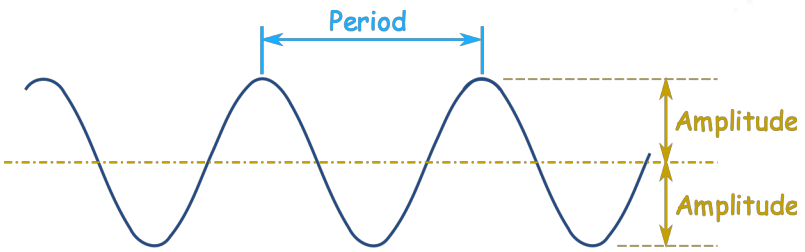
Frequency는 The number of compressed

단위는 Hertz를 사용하며, 1Hertz는 1초에 한번 Vibration을 의미합니다.

- 주기(period): 파동이 한번 진동하는데 걸리는 시간, 또는 그 길이,
  일반적으로 sin함수의 주기는 $2\pi/w$ 입니다.
- 주파수(frequency): 1초 동안의 진동 횟수입니다.



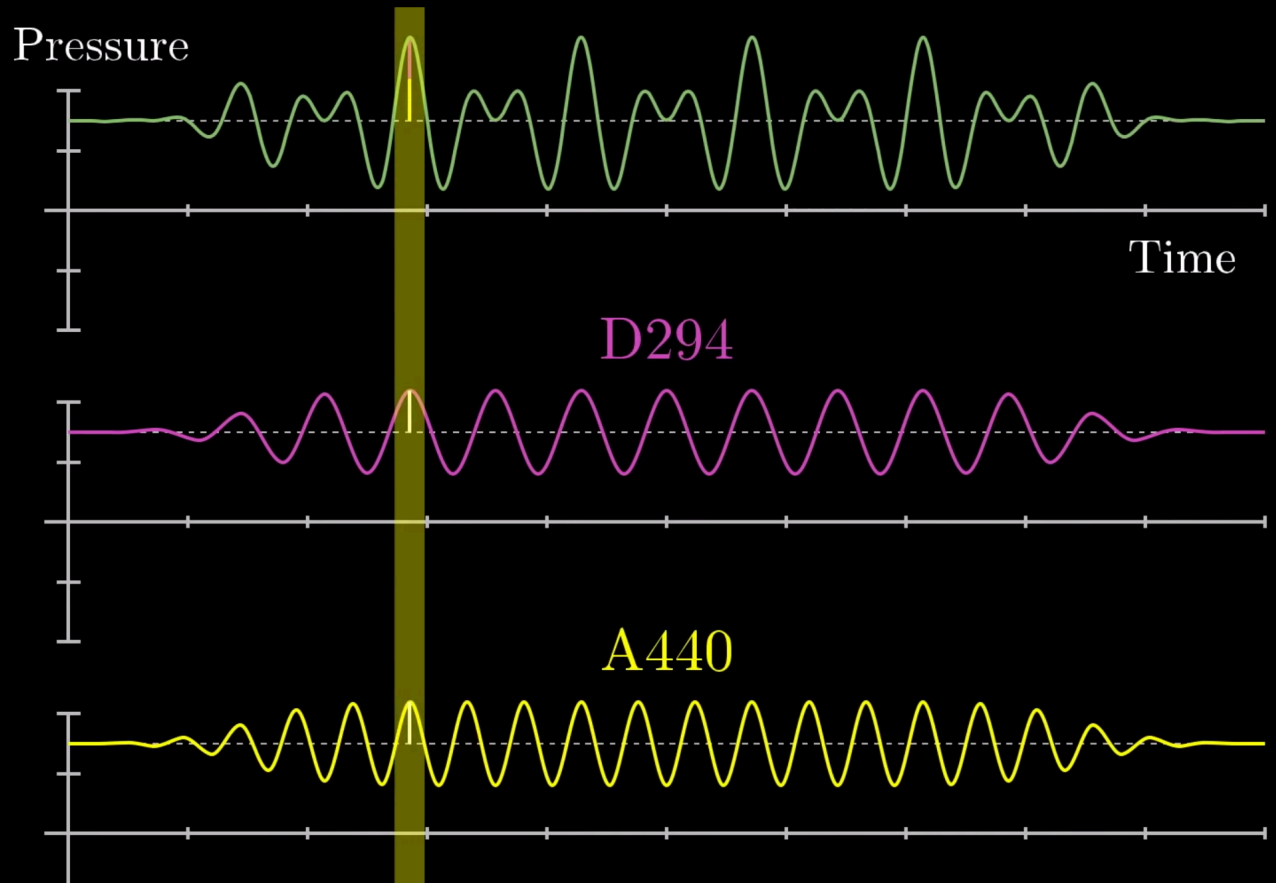




소리의 높낮이는 음원의 주파수에 의해 결정
주파수가 높으면 높은 소리가, 낮으면 낮은 소리가 난다.

Complex Wave?

우리가 사용하는 대부분의 **소리**들은 **복합파입니다**.
복합파(Complex wave)는 복수의 서로 다른 정현파들의 합으로 이루어진 파형(Wave)입니다.



Sinusoidal Wave?

정현파? 일종의 복소 주기함수라고 생각하면 좋습니다.

$$x(n) \approx \sum_{k=0}^{K} a_k(n) cos(\phi_k(n)) + e(n)$$

where,

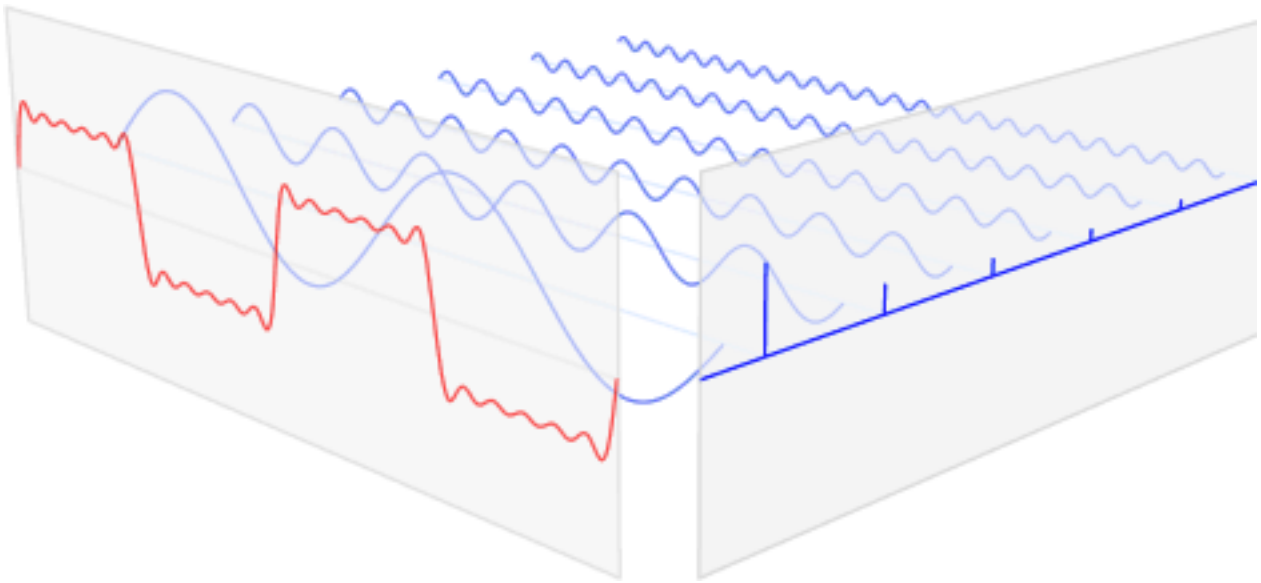
a_k = instantaneous amplitude
ϕ_k = instantaneous phase
$e(n)$ = residual (noise)

푸리에 변환 (Fourier transform)

CONNECT THE PYTHONISTAS

임의의 **입력 신호**를 다양한 주파수를 갖는
주기함수(복수 지수함수)들의 합으로 분해하여 표현하는 것

$$A_k = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\,\exp\left(-i\cdot 2\pi\frac{k}{T}t\right)dt$$



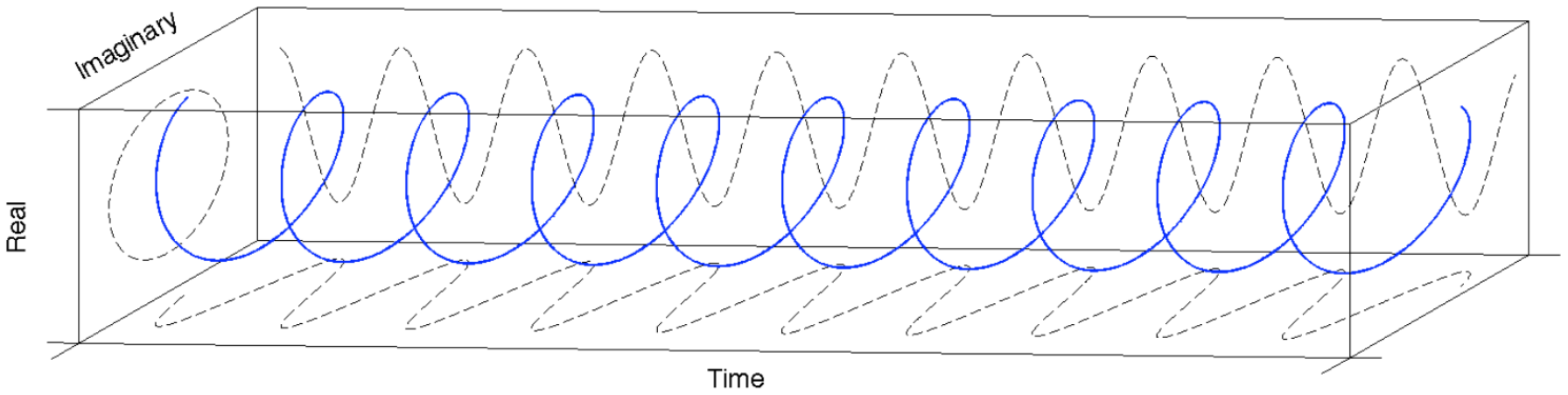
오일러 공식

지수함수와 **주기함수**와의 관계란?

여기서 $\cos(2\pi kT)$, $i \sin(2\pi kT)$함수는 주기와 주파수를 가지는 주기함수입니다.
즉 퓨리에 변환은 입력된 Audio를 sin, cos과 같은 주기함수들의 합으로
분해 가능하다는 것입니다.

$$e^{i\theta} = \cos\theta + i\sin\theta$$

$$\exp\left(i \cdot 2\pi \frac{k}{T} t\right) = \cos\left(2\pi \frac{k}{T}\right) + i \sin\left(2\pi \frac{k}{T}\right)$$



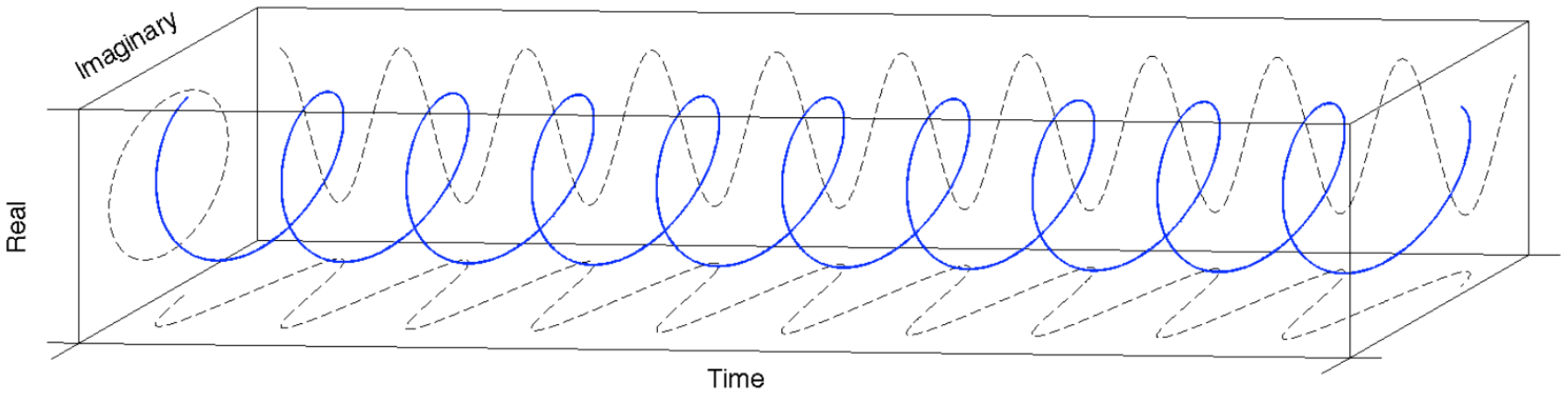
푸리에 변환의 결과

Spectrum magnitude, Phase spectrum

퓨리에 변환이 끝나면, 이제 실수부와 허수부를 가지는 복소수가 얻어집니다. 이러한 복소수의 절대값은 Spectrum magnitude라고 부르며 (주파수의 강도), 복소수가 가지는 phase를 phase spectrum (주파수의 위상)이라고 부르게 됩니다.

$$|X(k)| = \sqrt{X_R^2(k) + X_I^2(k)}$$

$$\angle X = \phi(k) = tan^{-1} \frac{X_I(k)}{X_R(k)}$$



DFT (Discrete Fourier Transform)

CONNECT THE PYTHONISTAS

Continuous to Discrete

컴퓨터가 인식하는 신호는 Discrete하게 sampling된 값입니다. 따라서 input signal이 연속적이지 않은 상황을 고려해야합니다.

만약에 우리가 수집한 데이터 y_n에서, 이산 시계열 데이터가 주기 N으로 반복한다고 할때, DFT는 주파수와 진폭이 다른 N개의 사인 함수의 합으로 표현이 가능합니다.

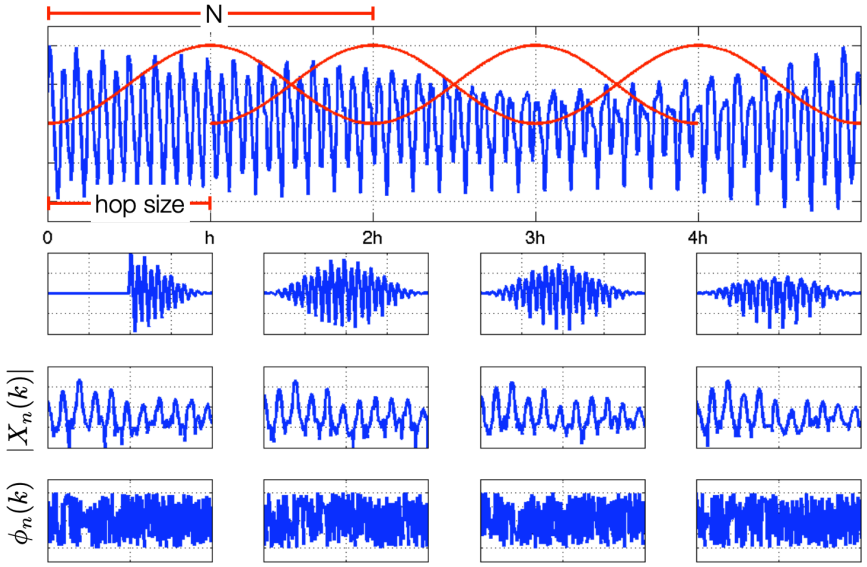
$$y_n = \frac{1}{N}\sum_{k=0}^{N-1} Y_k \cdot \exp\left(i \cdot 2\pi \frac{k}{N} n\right) \qquad Y_k = \sum_{n=0}^{N-1} y_n \cdot \exp\left(-i \cdot 2\pi \frac{k}{N} n\right)$$

- y_n : input signal
- n : Discrete time index
- k : discrete frequency index
- Y_k : k번째 frequeny에 대한 Spectrum의 값

STFT (Short Time Fourier Transform)

FFT는 시간에 흐름에 따라 신호의 주파수가 변했을때,
어느 시간대에 주파수가 변하는지 모르게 됩니다.
이러한 한계를 극복하기 위해서, STFT는 시간의 길이를 나눠서 이제 퓨리에 변환을 하게 됩니다.

$$X(l,k) = \sum_{n=0}^{N-1} w(n)x(n+lH)\exp^{\frac{-2\pi kn}{N}}$$



N : FFT size
- Window를 얼마나 많은 주파수 밴드로 나누는가 입니다.

Duration
- 샘플링 레이트를 window로 나눈 값입니다.
- $T = window/SR$
- T(Window) = 5T(Signal), duration은 신호주기보다 5배 이상 길게 잡아야한다.
- 440Hz 신호의 window size는 5*(1/440)이 됩니다.

$w(n)$: Window function
- 일반적으로 Hann window가 쓰입니다.

n : Window size
- Window 함수에 들어가는 Sample의 양입니다.
- 작을수록 Low-frequency resolution을 가지게 되고, high-time resolution을 가집니다
- 길수록 High-frequency, low time resolution을 가집니다.

H : Hop size
- 윈도우가 겹치는 사이즈입니다. 일반적으로는 1/4정도를 겹치게 합니다.

Tutorial (STFT)

CONNECT THE PYTHONISTAS

STFT

STFT는 time-domain의 정보가 날라가 버리는 문제를 해결하기 위해서 짧은 시간의 축을 기반으로 FFT를 진행하게 됩니다.

그 결과로는 Frequency Bin 수 x Time 구간수 크기의 복소수 매트릭스가 형성됩니다.
513 (Frequency), 44 (Time)

일반적으로 복소수 정보를 빼고 연산을 위해 제곱을 취해 power spectrogram을 만듭니다.

```
# STFT
S = librosa.core.stft(samples, n_fft=1024, hop_length=512, win_length=1024)
print(S.shape, len(S[0]), S[0][0])

((513, 44), 44, (-0.2504628+0j))
```

Tutorial (STFT)

librosa.core.stft(*y, n_fft=2048, hop_length=None, win_length=None, window='hann', center=True, dtype=<class 'numpy.complex64'>, pad_mode='reflect'*) [source]

Short-time Fourier transform (STFT)

Returns a complex-valued matrix D such that

np.abs(D[f, t]) is the magnitude of frequency bin *f* at frame *t*

np.angle(D[f, t]) is the phase of frequency bin *f* at frame *t*

Parameters:

y:np.ndarray [shape=(n,)], real-valued

the input signal (audio time series)

n_fft:int > 0 [scalar]

FFT window size

hop_length:int > 0 [scalar]

number audio of frames between STFT columns. If unspecified, defaults *win_length / 4*.

win_length:int <= n_fft [scalar]

Each frame of audio is windowed by *window()*. The window will be of length *win_length* and then padded with zeros to match *n_fft*.

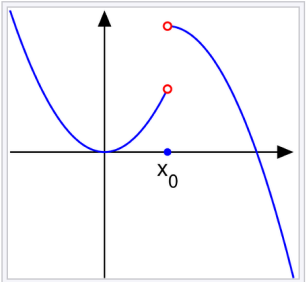
If unspecified, defaults to `win_length = n_fft`.

https://librosa.github.io/librosa/generated/librosa.core.stft.html

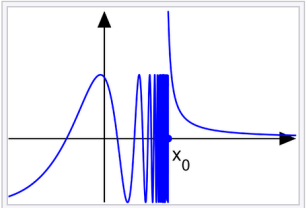
Tutorial (STFT)

Window Function

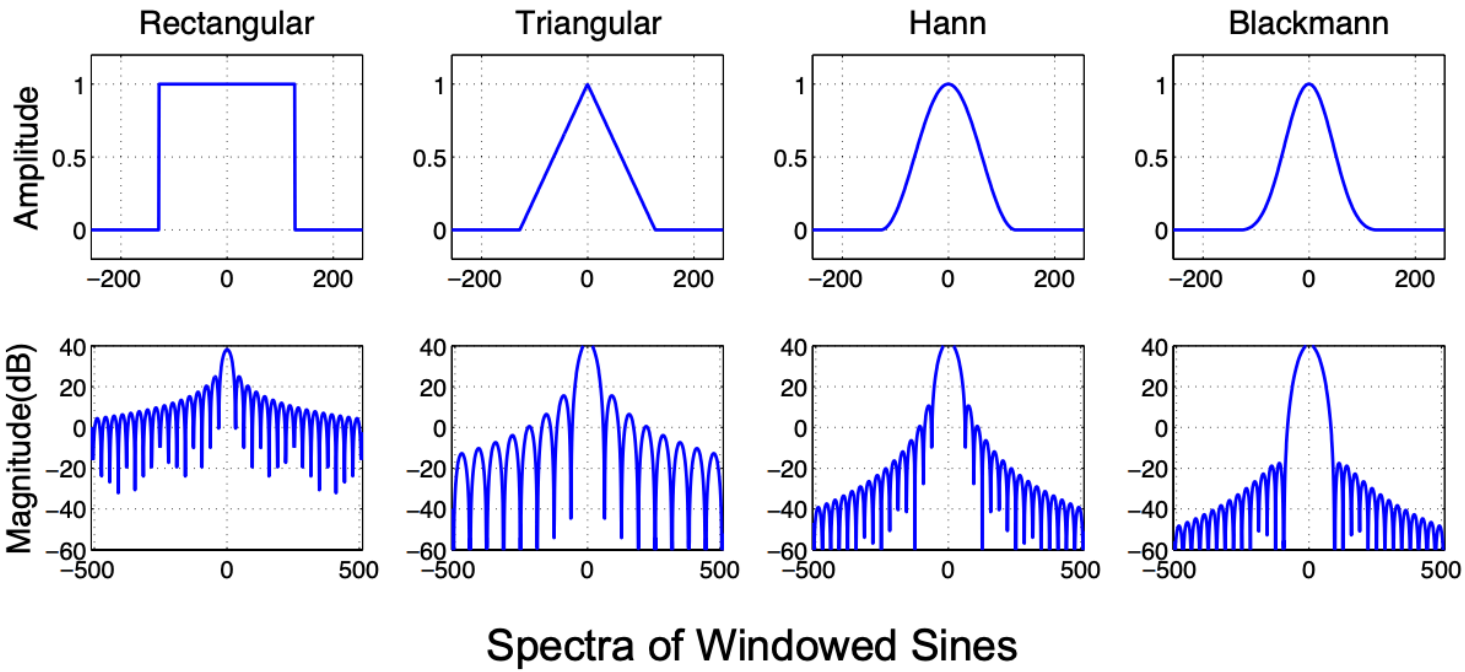
Window function의 주된 기능은 main-lobe의 width와 side-lobe의 레벨의 Trade-off 를 제어해 준다는 장점이 있습니다. 쉽게 말해서, Time bin 분리에 따른 불 연속성을 잡아주는 Window function입니다.



The function in example 2, a jump discontinuity



The function in example 3, an essential discontinuity

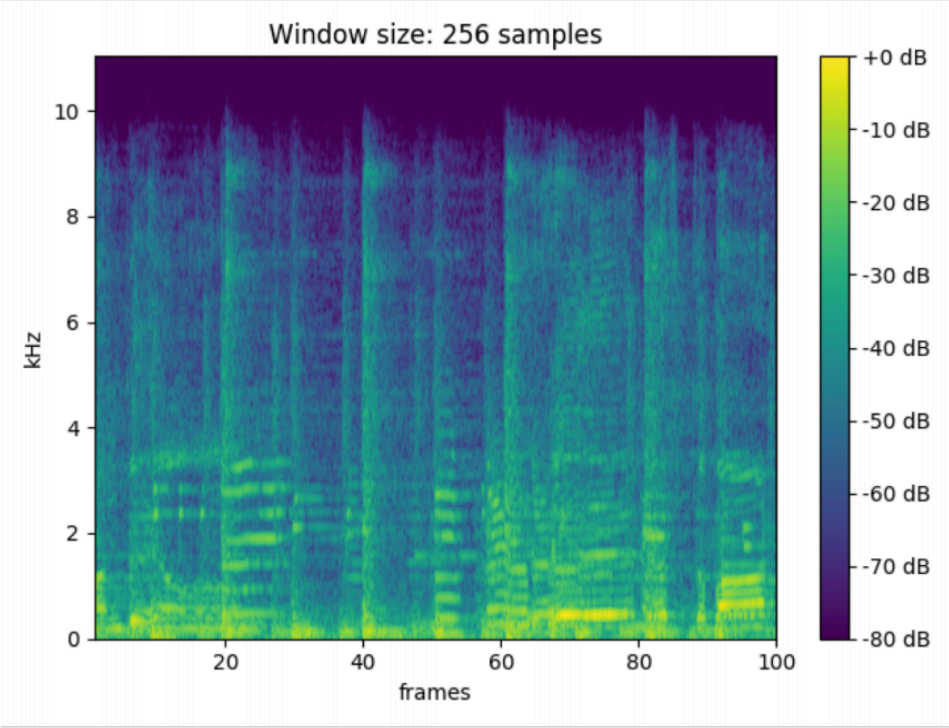


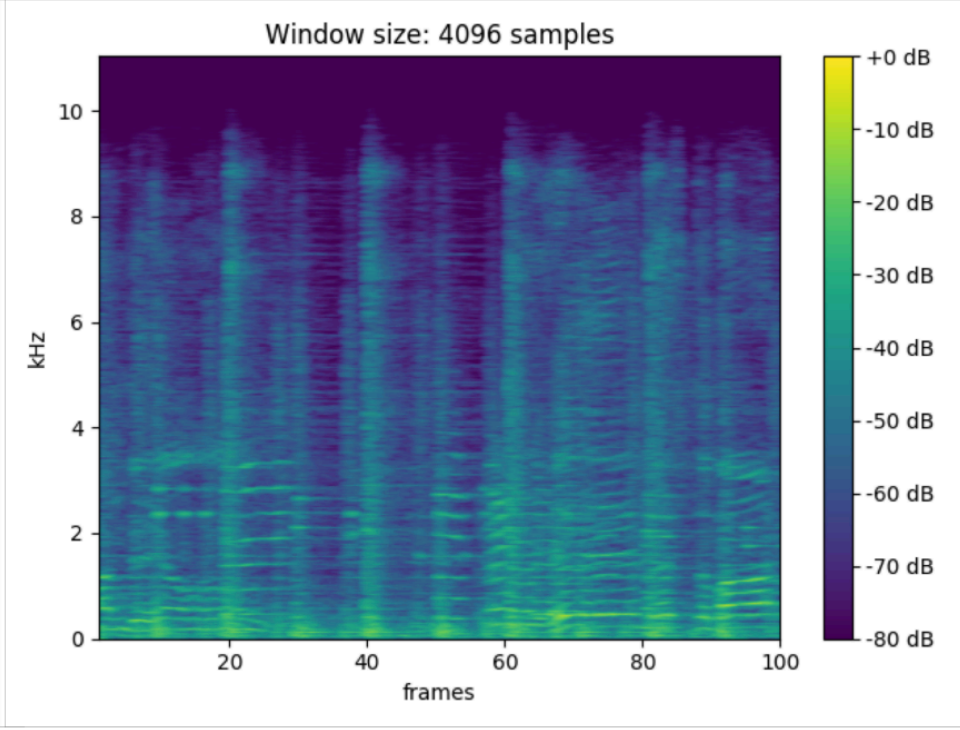
https://en.wikipedia.org/wiki/Classification_of_discontinuities#Jump_discontinuity

Tutorial (STFT)

Window Size

Window size에 따라서 Frequency와 Time Resolution에 대한 Trade-off가 발생합니다.





http://mac.kaist.ac.kr/~juhan/gct634/slides/02-audio_representations.pdf

주파수 영역은 이해가 되는데 무엇을 Model의 Input으로 넣어야하지?

Data Processing for Music

CONNECT THE PYTHONISTAS

Frequency-domain과 Time-domain을 어떻게 고려할 것인가?

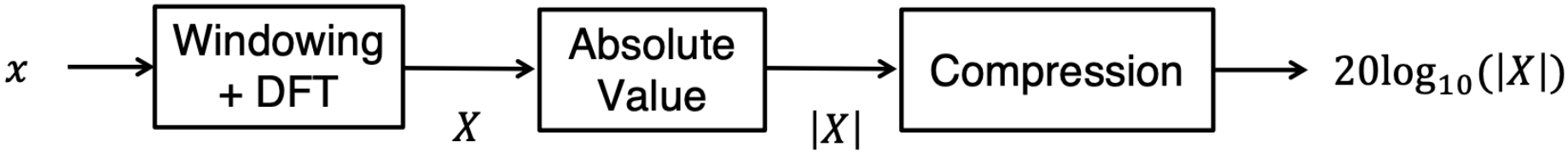
- Linear-Scale Spectrogram
- Log-Scale Spectrogram
- Mel Frequency Spectrogram
- Mel Frequency Cepstral Coefficients (MFCC)
- Constant Q-Transform (CQT)

Scaling with Frequency

Linear to Logarithmic Scale

사실상 복소수 영역에 있는 정보는 사용하기 힘들기 때문에 (사실 이것도 중요한 정보!, 나중에 Sample CNN이 등장한 배경이기도 합니다.)
우리는 STFT의 결과에 절대값을 취해주고 그것을 Log scale로 압축하게 됩니다.

Linear Scale은 그들이 가지고 있는 Amplitude를 그대로 나타내는 것이고,
반면에 Decibel Scale은 Linear Scale을 Logarithmic Scale로 변환 시킨 것 입니다.



- $20\log_{10}(|X|)$: decibel unit

db	Power Ratio	Voltage Ratio
+40	10000	100
+20	100	10
+6	4	2
+3	2	1.4
0	1	1
-3	1/2	1/1.4
-6	1/4	1/2
-20	1/100	1/10
-40	1/10000	1/100

http://digital.ni.com/public.nsf/allkb/8A93302DFBA0654D86256F32000C9CCB

Scaling with Frequency

Note	No. of semitones above C'	Multiplier of C' frequency	F (Hz)	$\log_{10}$ F
C'''	24	4	1056	3.0237
B	23	3.75	990	2.9956
A	21	3.333	880	2.9445
G	19	3	792	2.8987
F	17	2.667	704	2.8476
E	16	2.5	660	2.8195
D	14	2.25	594	2.7738
C''	12	2	528	2.7226
B	11	1.875	495	2.6946
A	9	1.667	440	2.6435
G	7	1.5	396	2.5977
F	5	1.333	352	2.5465
E	4	1.25	330	2.5185
D	2	1.125	297	2.4728
C'	0	1	264	2.4216

http://digital.ni.com/public.nsf/allkb/8A93302DFBA0654D86256F32000C9CCB

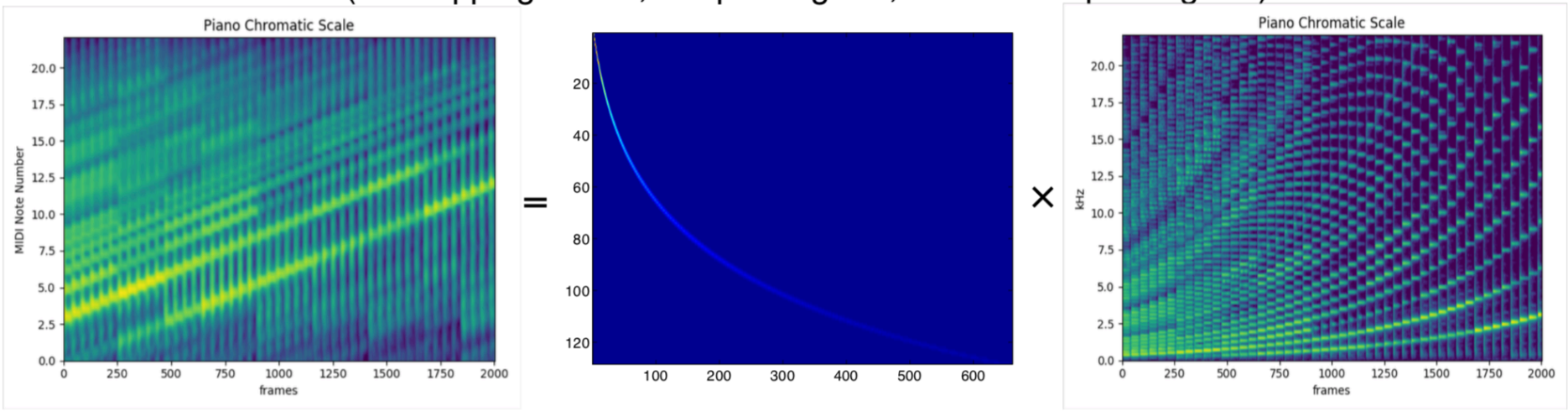
Scaling with Frequency

Linear to Logarithmic Scale

음악의 옥타브의 변화를 보면 C` 와 C`` 차이는 264 Hz. C`` 와 C``` 의 차이는 528Hz, 입니다 거의 둘의 차이가 2배 발생하죠. 이러한 Musical Scale을 반영하기 위해서는 일반적으로 Log scale을 사용합니다.

$$Y = M \cdot X$$

(M: mapping matrix, X: spectrogram, Y: scaled spectrogram)



- $20\log_{10}(|X|)$: decibel unit

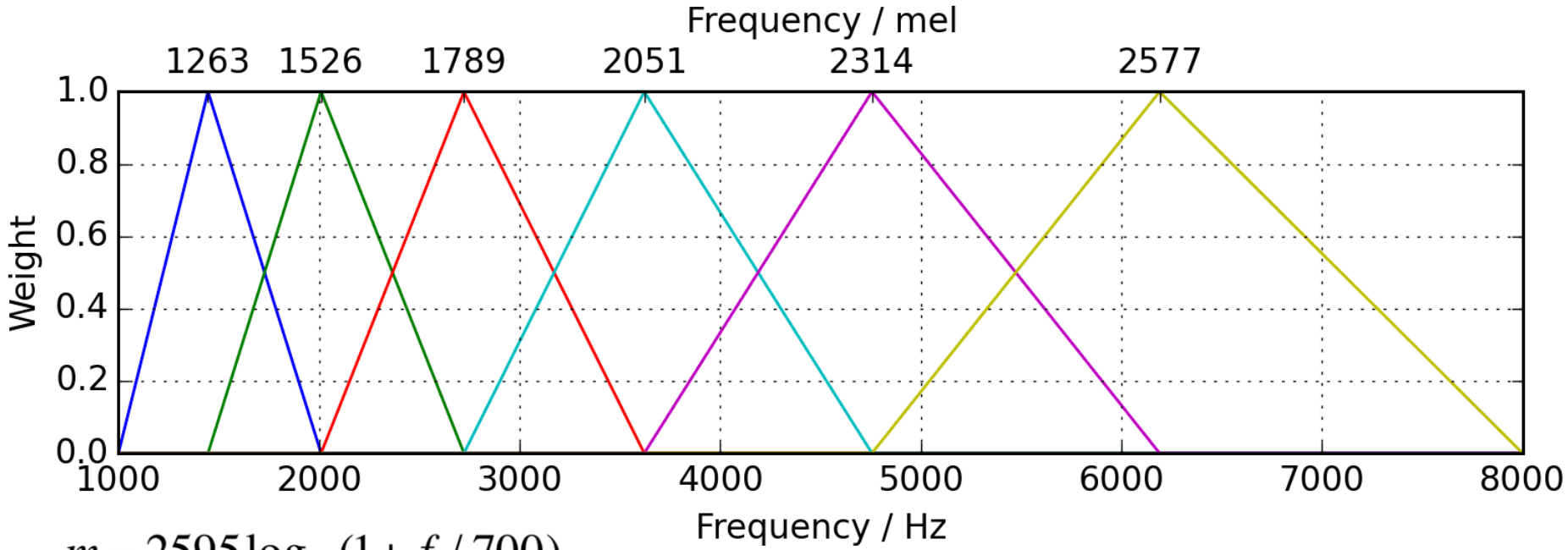
http://mac.kaist.ac.kr/~juhan/gct634/slides/02-audio_representations.pdf

Mel-Scale

사람은 인접한 주파수를 크게 구별하지 못한다!

우리의 인지기관이 categorical한 구분을 하기 때문입니다

멜 스펙트럼은 주파수 단위를 다음 공식에 따라 멜 단위로 바꾼 것을 의미합니다.
이 멜 필터는 인간의 인지방식를 고려하여 나왔으며, 우리가 낮은 frequency 영역대에 더 큰 정보를 받아드리는 것과 비슷합니다.

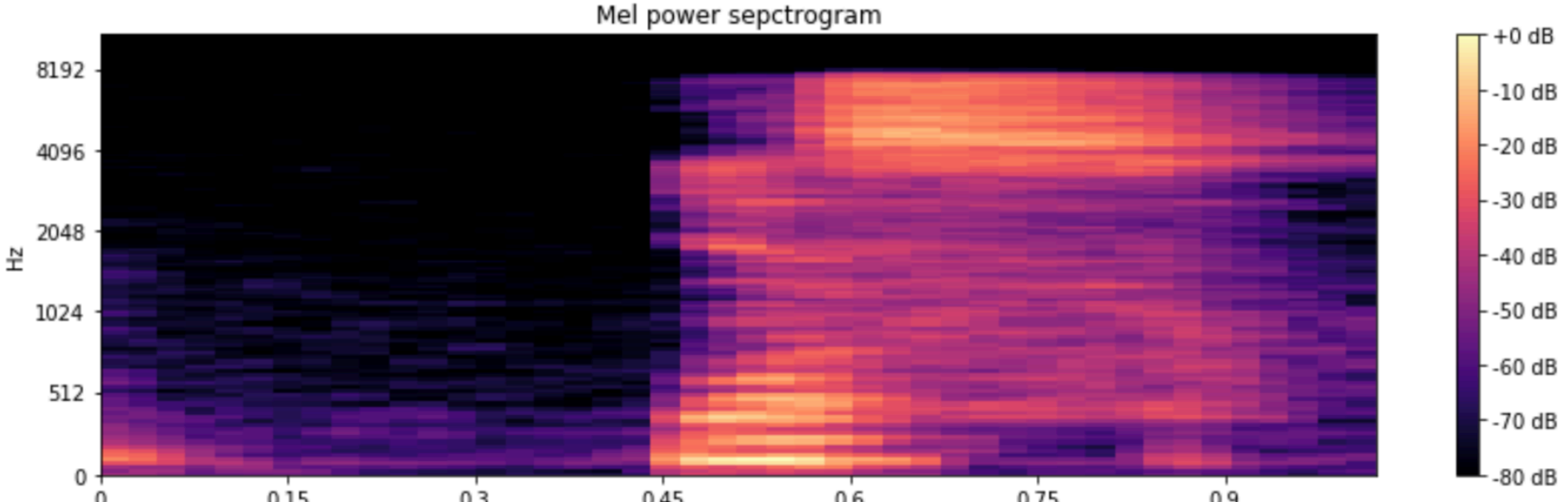


$$m = 2595 \log_{10}(1 + f / 700)$$

http://mac.kaist.ac.kr/~juhan/gct634/slides/02-audio_representations.pdf

Tutorial (Mel)

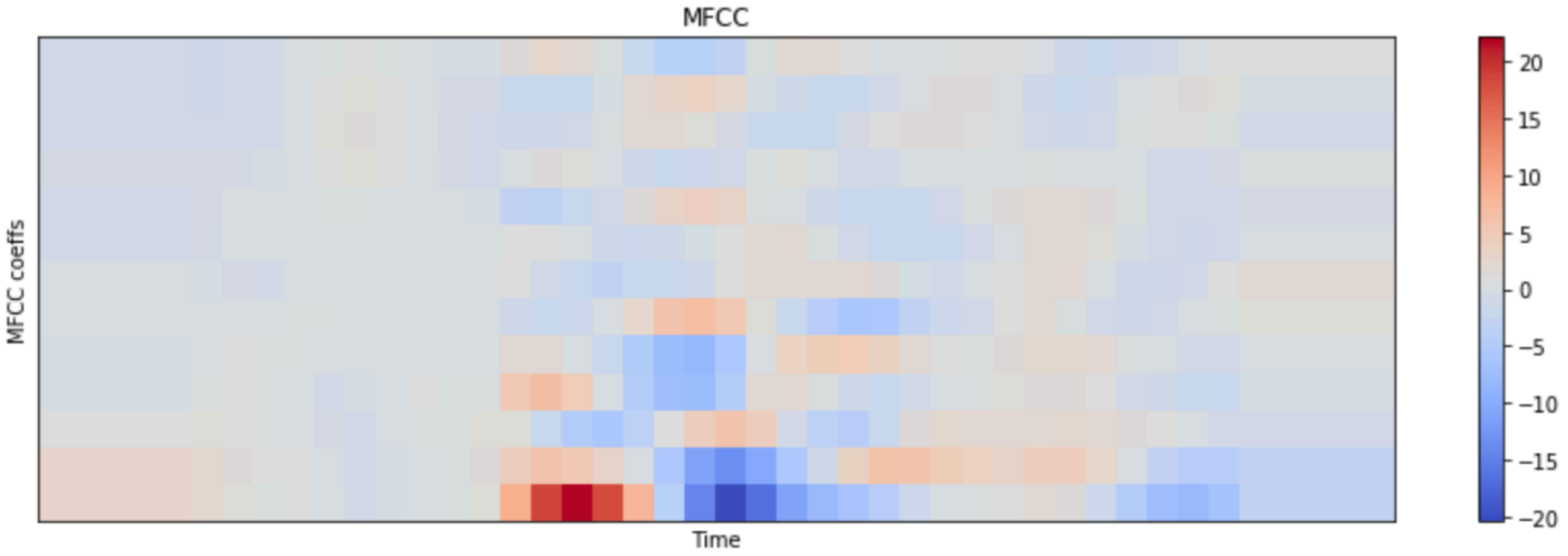
```
import librosa.display

S = librosa.feature.melspectrogram(samples, sr=sample_rate, n_mels = 128)
log_S = librosa.power_to_db(S, ref=np.max)

plt.figure(figsize=(12,4))
librosa.display.specshow(log_S, sr=sample_rate, x_axis='time', y_axis='mel')
plt.title('Mel power sepctrogram')
plt.colorbar(format='%+02.0f dB')
plt.tight_layout()
```



MFCC

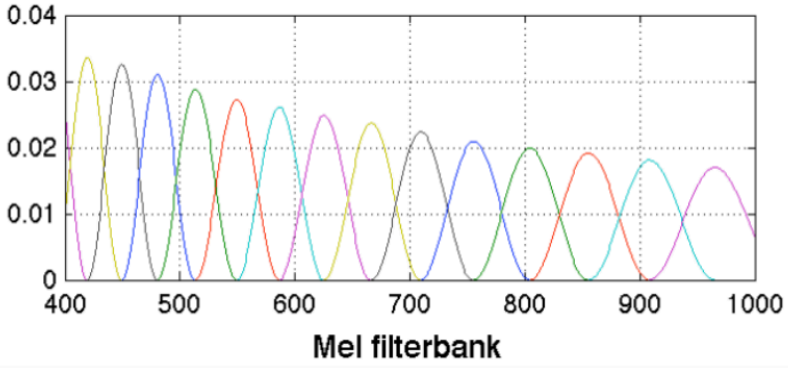
Filter Bank는 모두 Overlapping 되어 있기 때문에 Filter Bank 에너지들 사이에 상관관계가 존재

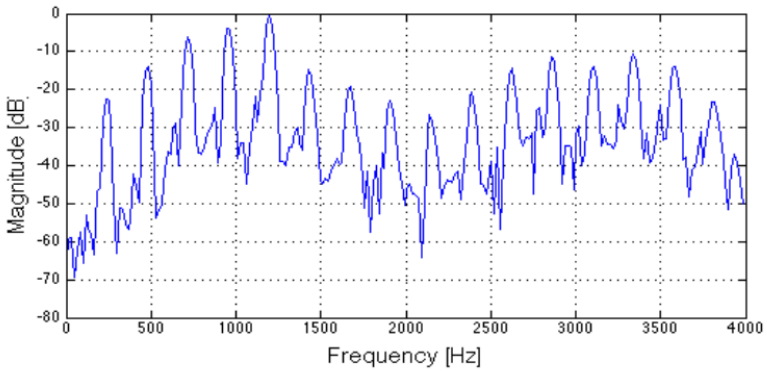
- Mel-Spectrum 에 log 적용
- Mel-log-Spectrum list 전체에 DCT(Discrete Cosine transform) 적용
- 얻어진 Coefficients 에서 앞에서부터 N개만 남기고 버리게됩니다.



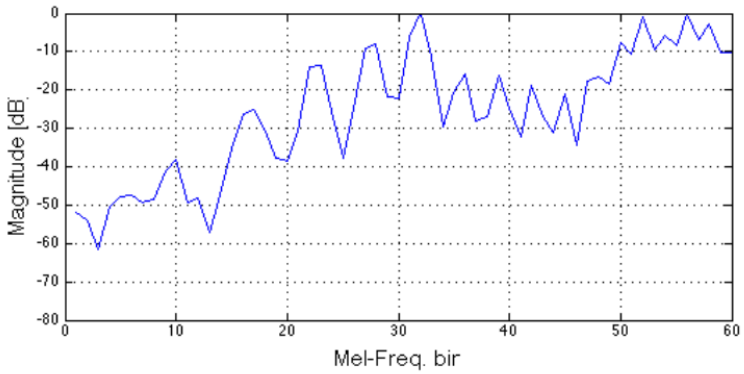
MFCC

STFT이후에
각 Spectrum에 mel-filter bank를 적용 → Mel-Spectrum
Mel-Spectrum 에 log 적용





Spectrum



Spectrum (mel-scaled)

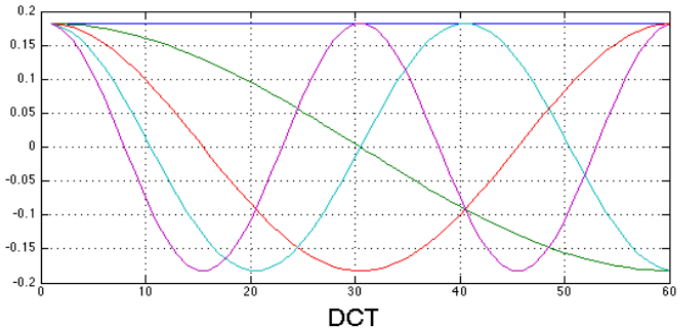
$$m = 2595 \log_{10}(1 + f / 700)$$

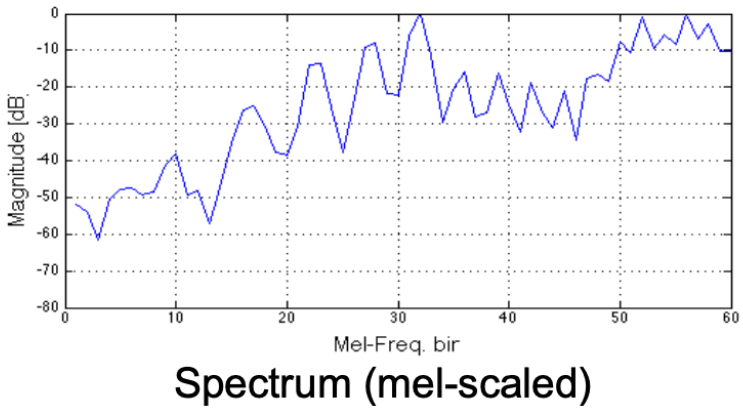
http://mac.kaist.ac.kr/~juhan/gct634/slides/02-audio_representations.pdf

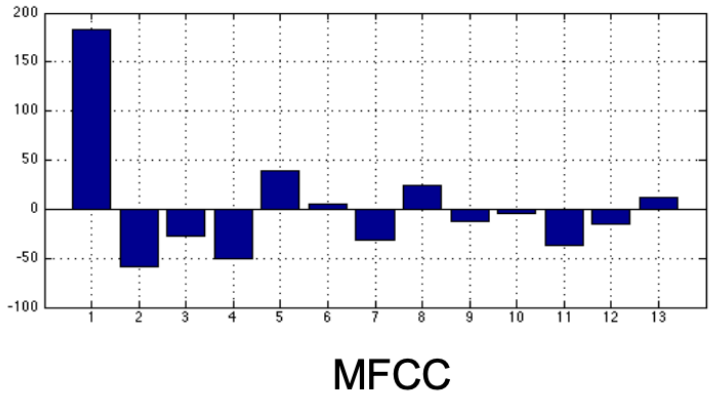
MFCC

DCT(Discrete Cosine transform) 적용
얻어진 Coefficients 에서 앞에서부터 N개만 남기고 버린다.
결과적으로 Input의 차원을 낮춰주는데 사용된다!

$$X_{DCT}(k) = \sqrt{\frac{2}{N}} \sum_{n=1}^{N-1} x(n) \cos\left(\frac{\pi k}{N}(n - 0.5)\right)$$







$$m = 2595 \log_{10}(1 + f / 700)$$

http://mac.kaist.ac.kr/~juhan/gct634/slides/02-audio_representations.pdf

MFCC

CONNECT THE PYTHONISTAS

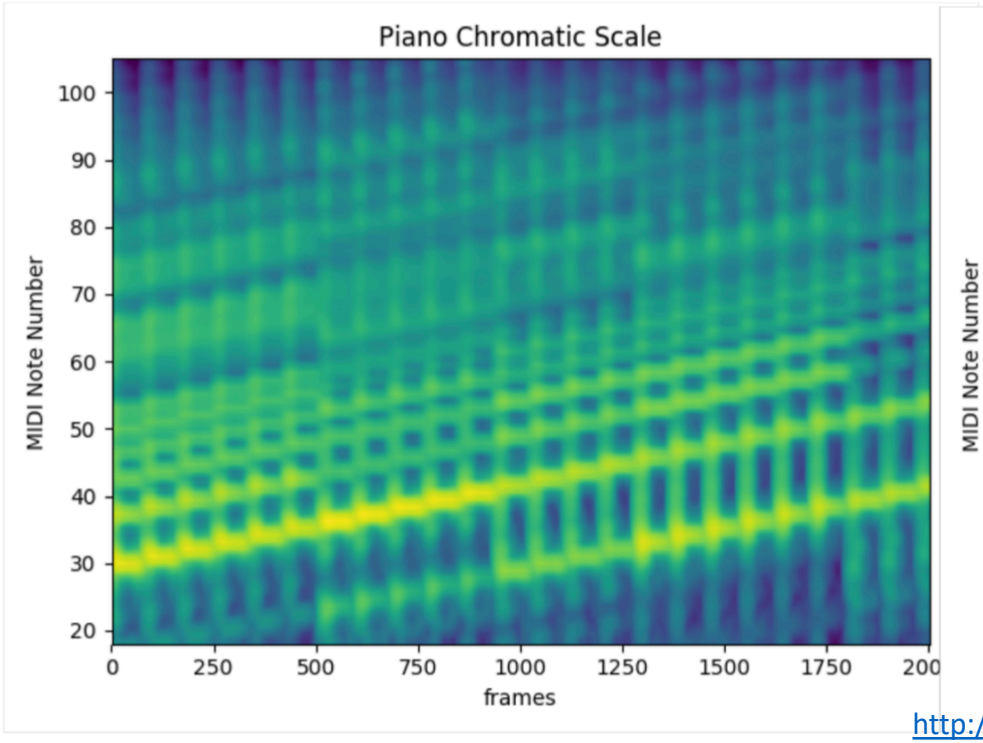
```
# mfcc (DCT)
mfcc = librosa.feature.mfcc(S=log_mel_S, n_mfcc=13)
mfcc = mfcc.astype(np.float32)    # to save the memory (64 to 32 bits)
mfcc[0]

array([-352.24487, -366.56326, -374.9281 , -390.60297, -385.9468 ,
       -392.3261 , -398.55487, -396.6922 , -395.0417 , -398.3241 ,
       -400.8156 , -399.5283 , -400.10876, -405.3044 , -407.25635,
       -410.77112, -409.39105, -405.62488, -400.28088, -386.88754,
       -256.587  , -184.99794, -167.65366, -155.98474, -151.11472,
       -136.15807, -139.98293, -145.12239, -152.257  , -160.66386,
       -164.44904, -166.04997, -172.92793, -178.01353, -180.90056,
       -187.92828, -179.37764, -190.39357, -217.45445, -235.33223,
       -262.61517, -282.51657, -303.91086, -313.25308], dtype=float32)
```

Constant Q-bank

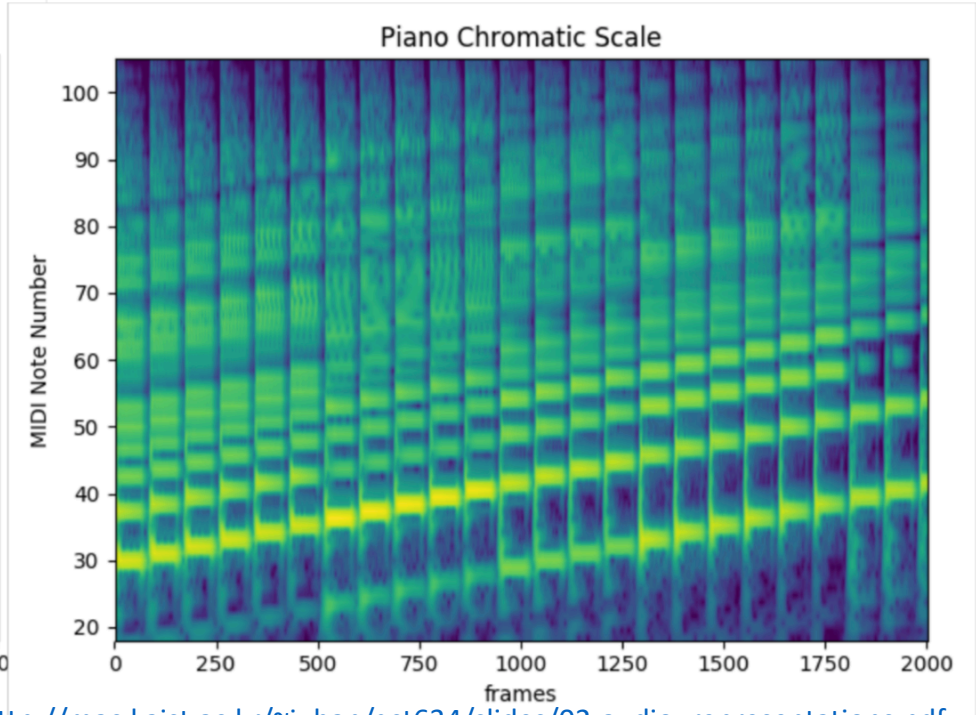
조금더 확실하게 음악적인 Scale을 적용해보자

1/24-oct filter bank를 적용한 것입니다.. 어떠한 2개의 주파수영역대를 선택하더라도 동일한 패턴을 보이며 화성적인 요소를 잘 표현하기 위해서 사용되고 있습니다.

- Logarithmically spaced frequencies
- Constant Q = frequency/bandwidth



Log-Frequency Spectrogram (mapping)



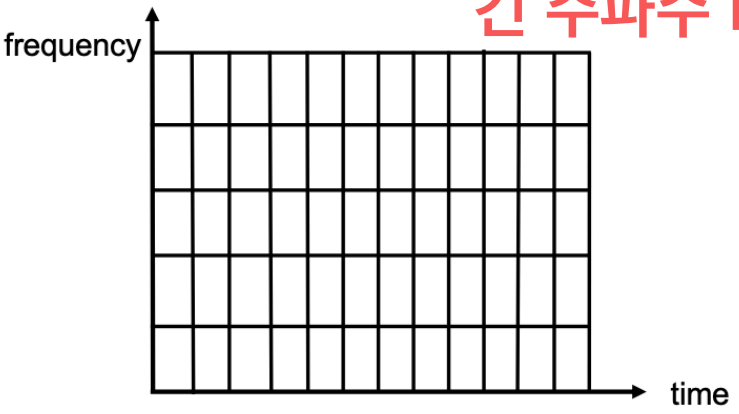
http://mac.kaist.ac.kr/~juhan/gct634/slides/02-audio_representations.pdf

Log-Frequency Spectrogram (Constant-Q transform)

Conclusion

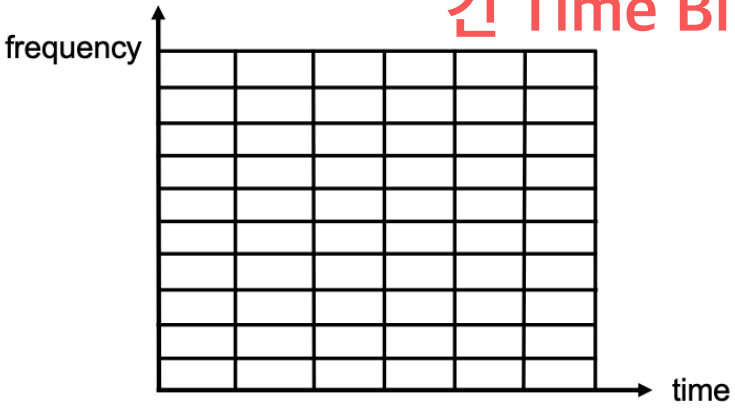
Frequency-domain과 Time-domain을 어떻게 고려할 것인가?

긴 주파수 Bin



Spectrogram (short window)

긴 Time Bin

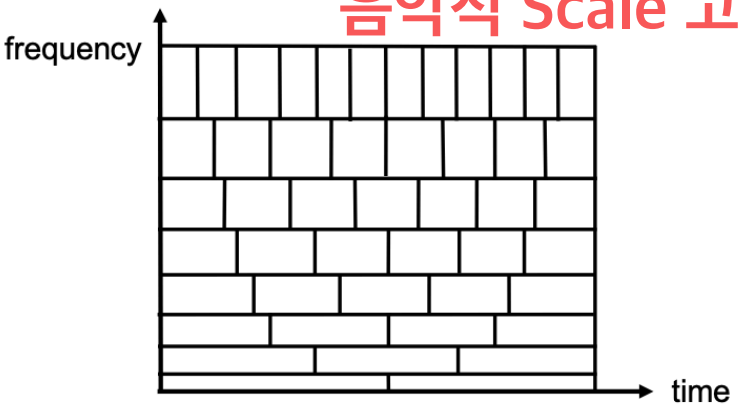


Spectrogram (long window)

인간의 청각인지 고려

Mel Spectrogram

음악적 Scale 고려

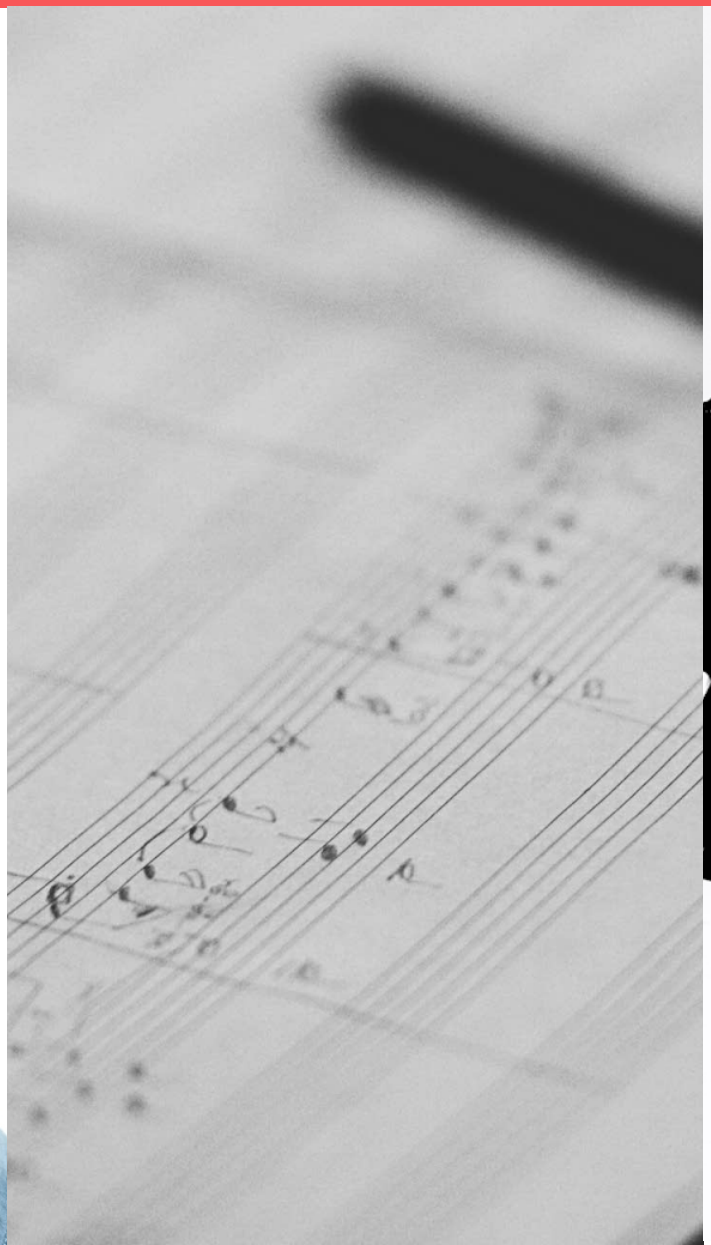


Constant-Q transform

http://mac.kaist.ac.kr/~juhan/gct634/slides/02-audio_representations.pdf

10 Min
Time Break

Deep Learning for music?



Deep Learning for music?

CONNECT THE PYTHONISTAS

Music Classification (음악 검색)

- Music Instrument Classification
- Music Genre Classification
- Music Singer Classification
- Music Emotion Classification

= **Music Auto-tagging**

SIMILARITY (음악 추천)

- Similarity measurement
- Cover song identification
- Query by humming
- Music recommendation
- Playlist generation

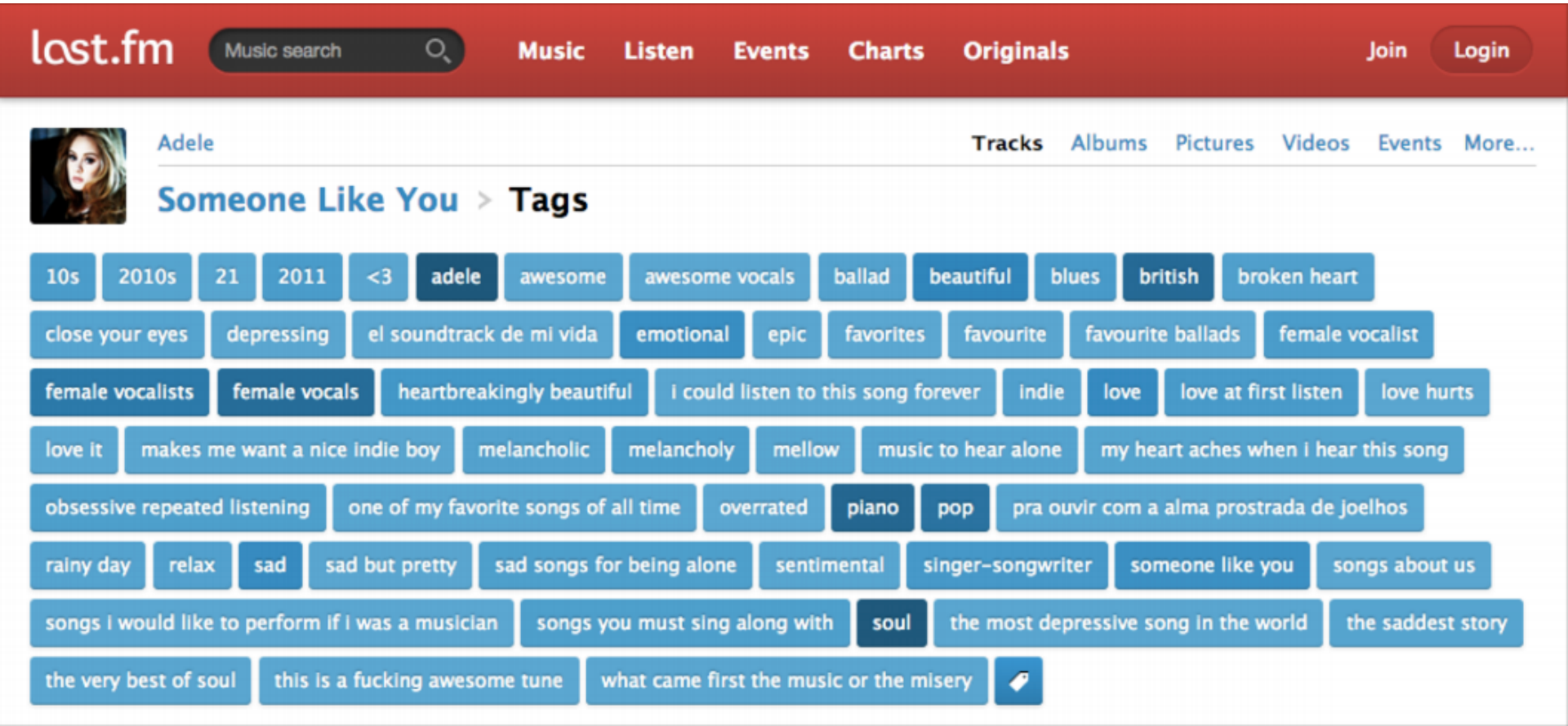
Music Transcription (작곡, 연주)

- Language Model of Music
- Human like performance
- Timbre description
- Melody extraction
- Onset detection, Beat tracking, and tempo estimation
- Tonality estimation: chroma, chord, and key
- Structural analysis, segmentation and summarization

Music Classification

Music Genre Classification 이란?

CV도메인에서 하는 Object detection 처럼, 음악 역시 여러 Label이 담기게 됩니다.
Genre, Mood, Instrument, Artist등등으로 나눌 수 있습니다.



Music Classification

CONNECT THE PYTHONISTAS

Music (Multi-Label) Classification = Auto tagging

This is a [ ] song that is [ ],
[ ] and [ ]. It features [ ] and [ ]
vocal. It is a song with [ ] and [ ] that you might like
to listen to while [ ].

Music Classification

CONNECT THE PYTHONISTAS

Music (Multi-Label) Classification = Auto tagging

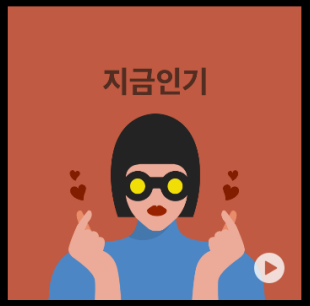
This is a [ ] song that is [ ], [ ] and [ ]. It features [ ] and [ ] vocal. It is a song with [ ] and [ ] that you might like to listen to while [ ].

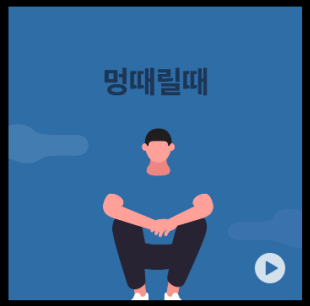
This is a [**very danceable**] song that is [**arousing/awakening**], [**exciting/thrilling**] and [**happy**]. It features [**strong**] and [**fast tempo**] vocal. It is a song with [**high energy**] and [**high beat**] that you might like to listen to while [**at a party**].

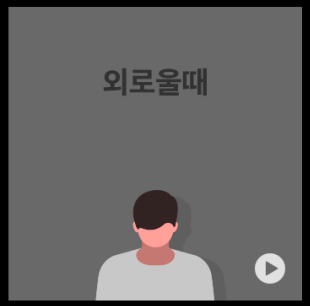
DJ

테마

느낌별 스테이션







Similarity-based Retrieval

query

0 (1.0) Eminem	HIP-HOP
1 (0.9182) OutKast	
2 (0.9027) Spezializtz	
3 (0.9012) Dino MC 47	
4 (0.8878) Bone Thugs-N-Harmony	
5 (0.8840) Method Man	
6 (0.8814) Big Punisher	
7 (0.8761) Cor Veleno	
8 (0.8642) Lil' Wayne	
9 (0.845) Obie Trice	
10 (0.8301) mcenroe & Birdapres	

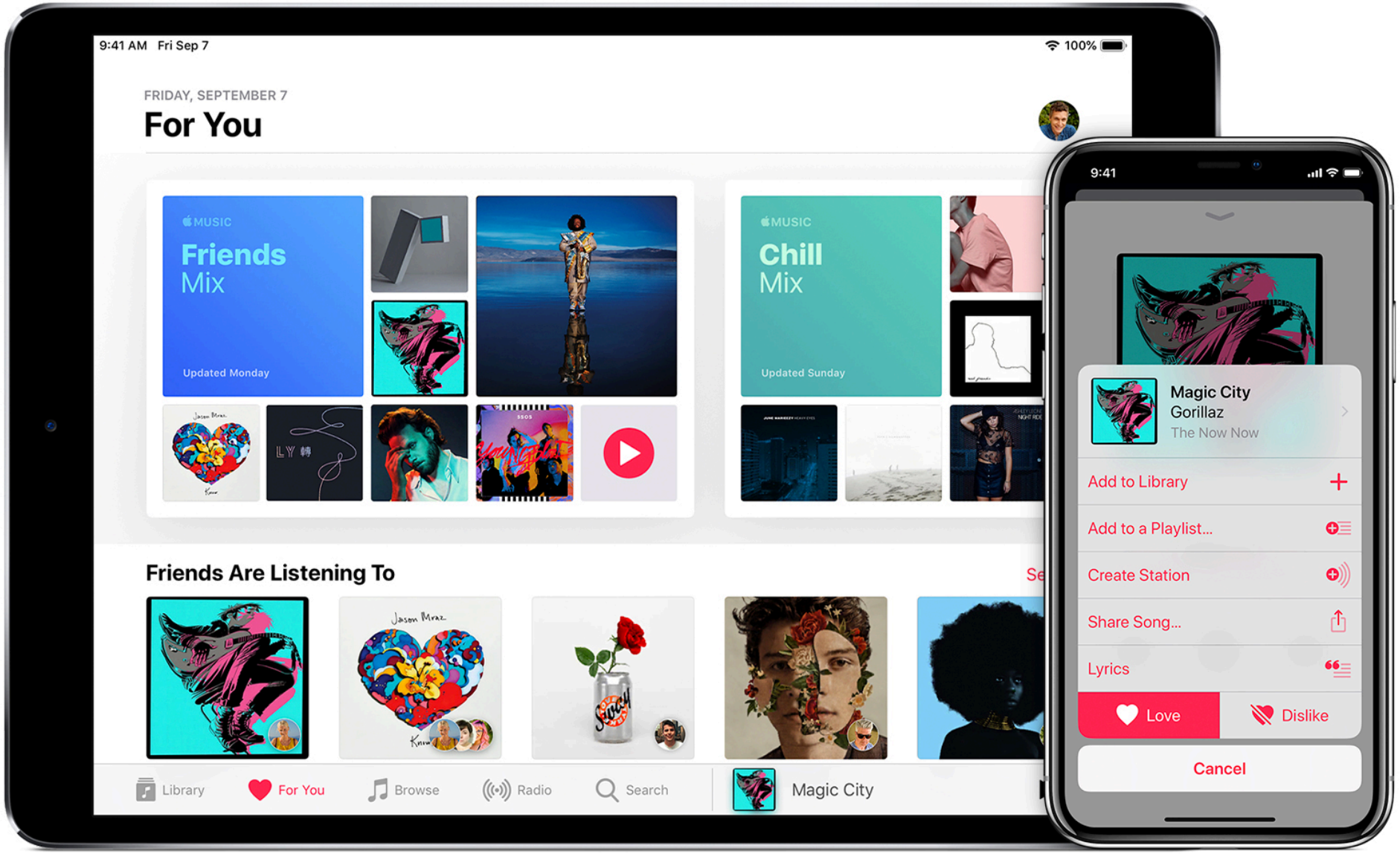
0 (1.0) Green Day	Rock Band
1 (0.8728) Unwritten Law	
2 (0.8692) P.O.D.	
3 (0.8237) Tokyo Rose	
4 (0.7903) Linkin Park	
5 (0.7532) All Star United	
6 (0.7532) Hinder	
7 (0.7420) The All-American Rejects	
8 (0.7384) Jimmy Eat World	
9 (0.7206) Third Day	
10 (0.7000) Bon Jovi	

0 (1.0) John Lennon	Pop rock
1 (0.8355) Cliff Richard	
2 (0.8199) The Who	
3 (0.7959) Status Quo	
4 (0.7836) Nick Cave and the Bad Seeds	
5 (0.7772) T.Love	
6 (0.7713) Blake Morgan	
7 (0.7538) Radney Foster	
8 (0.7509) Coldplay	
9 (0.7409) The Beach Boys	
10 (0.7391) Badly Drawn Boy	

0 (1.0) EXO	Boy group
1 (0.9554) 세븐틴	
2 (0.9496) 2PM	
3 (0.9484) NCT 127	
4 (0.9422) 몬스타엑스	
5 (0.9391) GOT7 (갓세븐)	
6 (0.9373) 비투비 (BTOB)	
7 (0.9344) 초신성	
8 (0.9314) 비스트 (Beast)	
9 (0.9273) B.A.P	
10 (0.9241) 샤이니 (SHINee)	

0 (1.0) 들국화	80's
1 (0.9199) 동물원	
2 (0.9153) 조하문	
3 (0.8833) 푸른하늘	
4 (0.8816) 김현식	
5 (0.8607) 해바라기	
6 (0.8492) 유재하	
7 (0.8468) 김현철	
8 (0.8458) 김민우	
9 (0.8427) 피노키오	
10 (0.8394) 한동준	

0 (1.0) 다이나믹 듀오(Dynamic Duo)	HIP-HOP
1 (0.9600) 드렁큰타이거	
2 (0.9456) 소울 다이브(Soul Dive)	
3 (0.9421) 개코	
4 (0.9334) 언터처블	
5 (0.9310) 허클베리피(Huckleberry P)	
6 (0.9291) 스윙스(Swings)	
7 (0.9242) 개리	
8 (0.9218) 방탄소년단	
9 (0.9201) Simon Dominic	
10 (0.9190) 빈지노(Beenzino)	


9:41 AM Fri Sep 7
100%
FRIDAY, SEPTEMBER 7
For You
MUSIC
Friends
Mix
Updated Monday
MUSIC
Chill
Mix
Updated Sunday
Friends Are Listening To
Library
For You
Browse
Radio
Search
Magic City
9:41
Magic City
Gorillaz
The Now Now
Add to Library
Add to a Playlist...
Create Station
Share Song...
Lyrics
Love
Dislike
Cancel

Spotify

CONNECT THE PYTHONISTAS

FAST FACTS

Number of subscribers (as at March 31, 2019)

100 Million

Number of monthly active users (as at March 31, 2019)

217 Million

Revenue paid to rightsholders (as at August 31, 2018)

€10 Billion

Number of tracks

50 Million +

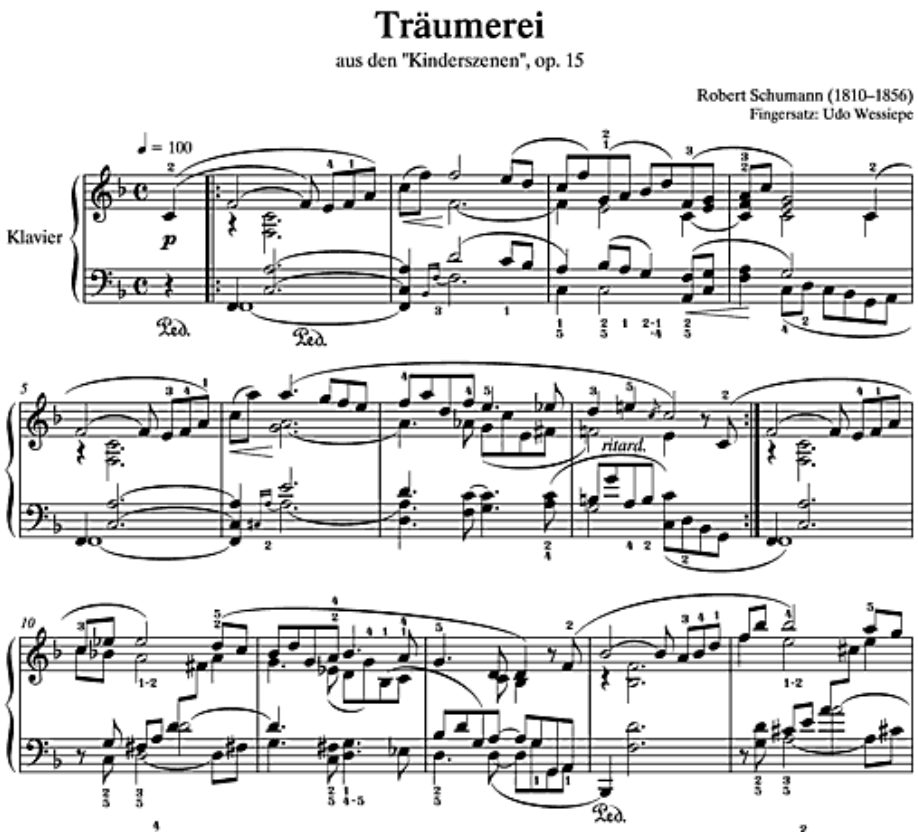
Number of playlists

3 Billion +

Spotify is available in

79 Markets

Performance Synthesis (연주 합성)



MIDI (score)

Valentina Lisitsa

Vladimir Horowitz

VirtuosoNet

Chopin Fantasie Impromptu

Generating Music

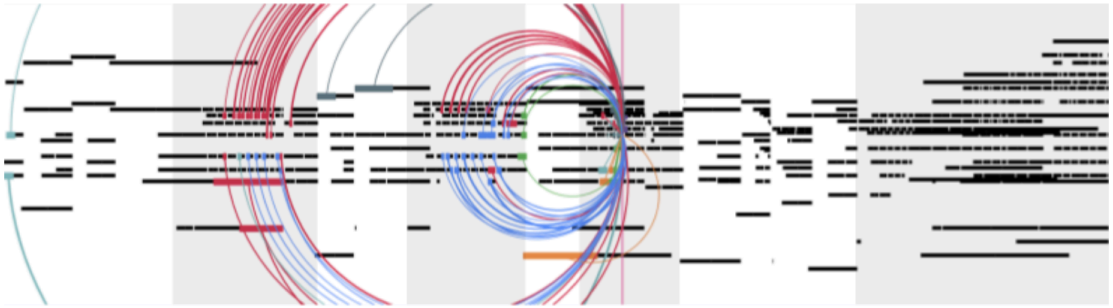
CONNECT THE PYTHONISTAS

magenta Get Started Studio Demos Blog Research Talks Community

Music Transformer: Generating Music with Long-Term Structure

Dec 13, 2018

Cheng-Zhi Anna Huang czhuang huangcza
Ian Simon iansimon iansimon
Monica Dinculescu notwaldorf notwaldorf

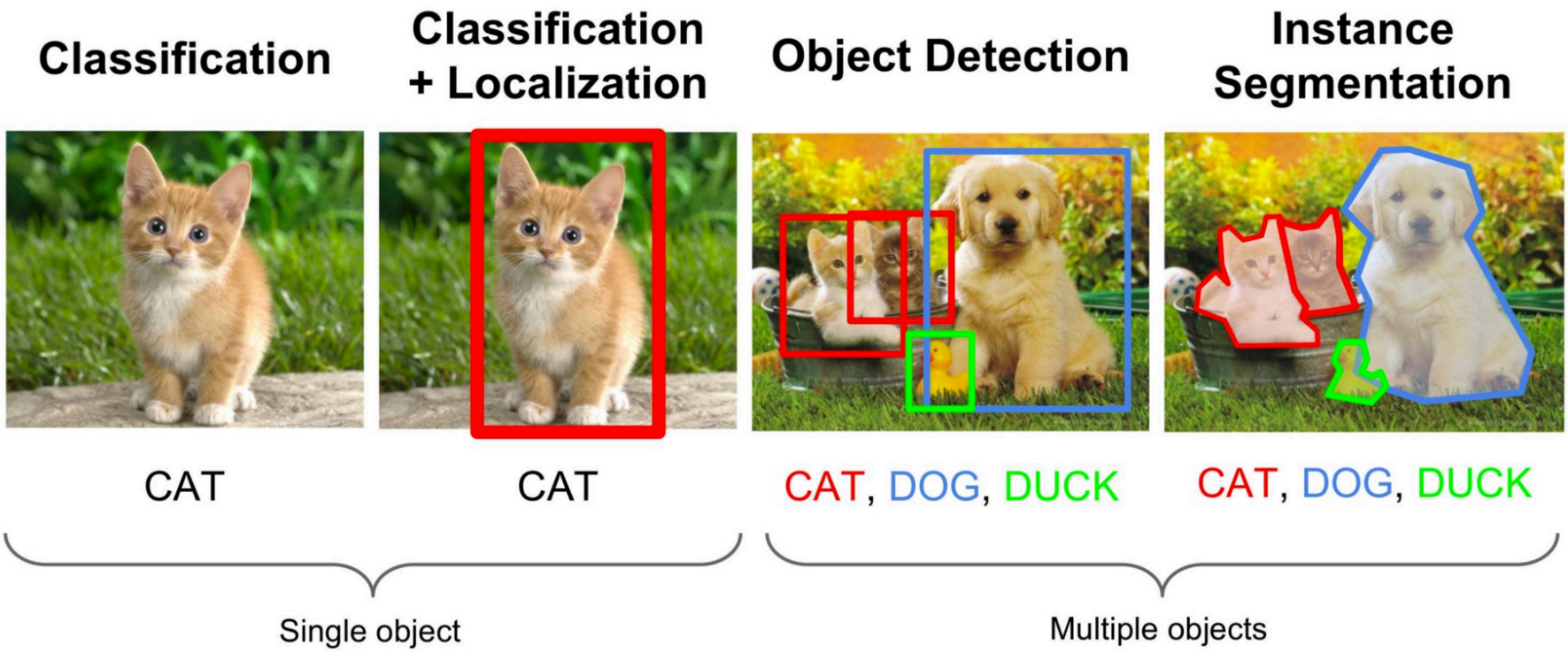


Generating long pieces of music is a challenging problem, as music contains structure at multiple timescales, from milisecond timings to motifs to phrases to repetition of entire sections. We present Music Transformer, an attention-based neural network that can generate music with improved long-term coherence. Here are three piano performances generated by the model:

오늘은
Music Genre Classification!

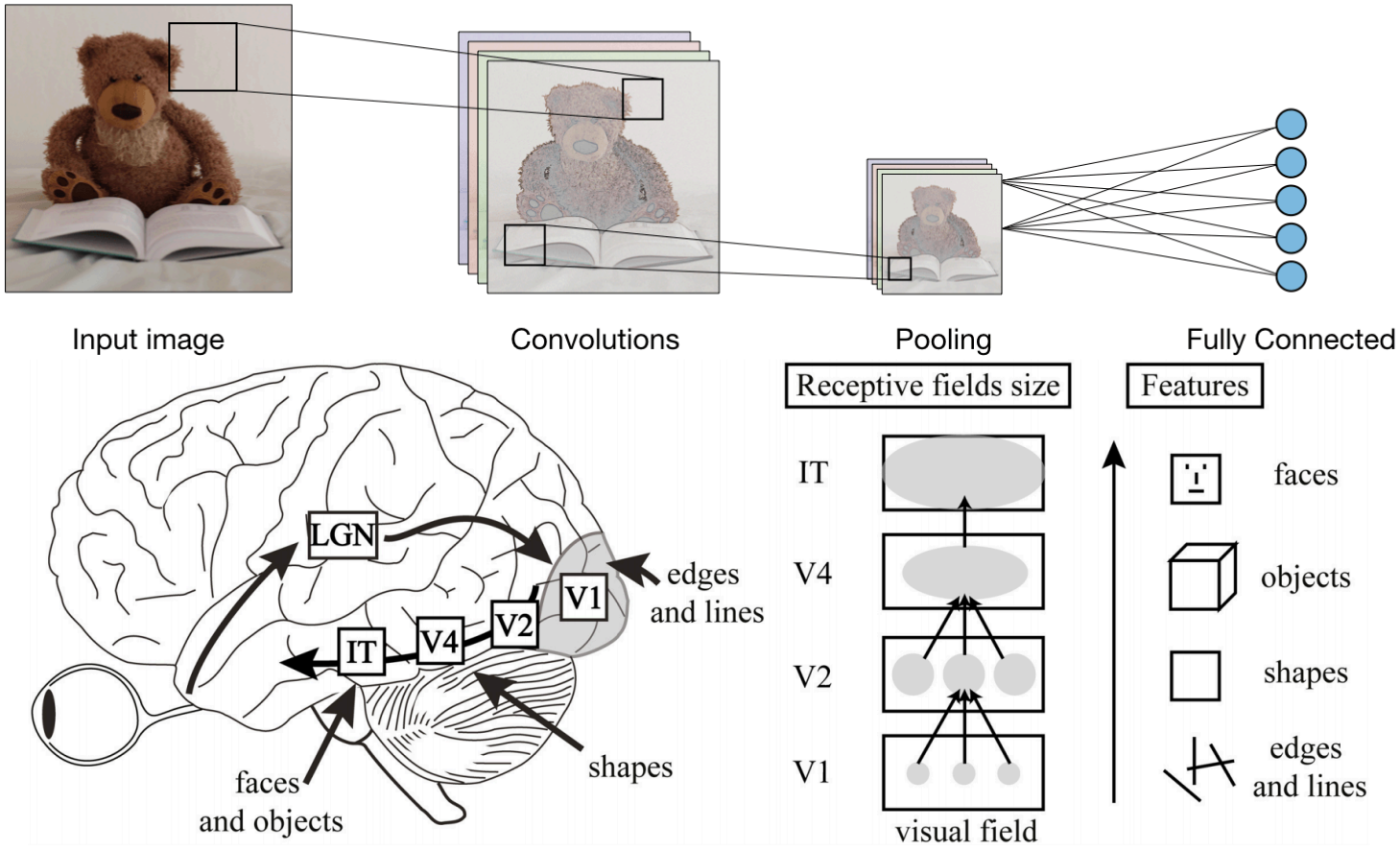
Music Classification

이미지 픽셀(x)이 들어왔을 때 고양이(y)를 찾는다!
음악(x)이 들어왔을 때 장르(y)를 찾는다!



Convolution Neural Network

Computer Vision에서 사용하는 CNN은 여러 Local한 Receptive fields를 고려하여 Feature를 학습합니다.



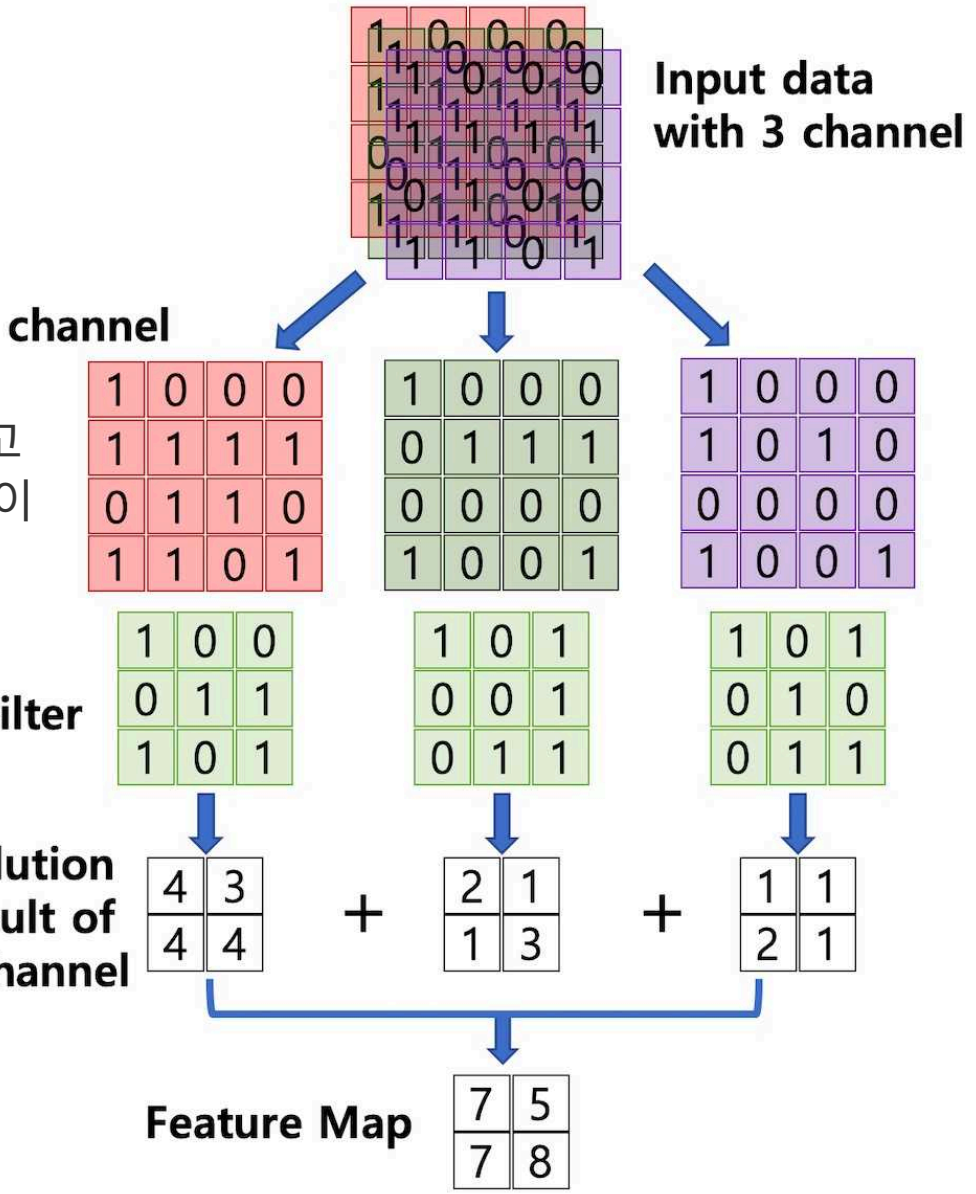
(Manassi, 2013)

Convolution Neural Network

Filter

필터는 그 특징이 데이터에 있는지 없는지를 검출해주는 함수입니다.

필터는 입력받은 데이터에서 그 특성을 가지고 있으면 큰값이 나오고, 특성이 없다면 결과값이 0에 가까운 값이 나오게 되서, 해당 특성이 있는지를 파악하게 해주게 됩니다.



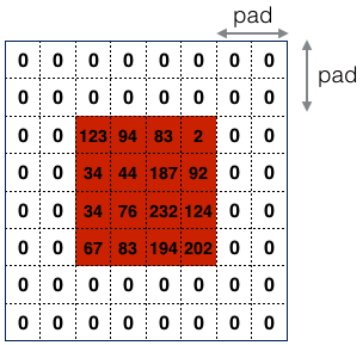
Convolution Neural Network

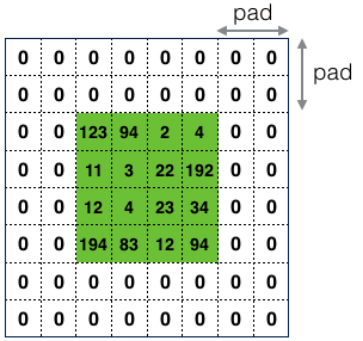
Padding

패딩은 Feature Map의 크기가 입력데이터보다 작아지는 현상을 방지하는 방법입니다.

출력데이터와 입력데이터의 크기를 갖게 만들기 위해, 입력 데이터의 외각에 지정된 픽셀만큼 특정 값으로 채워 넣는 것을 의미합니다.

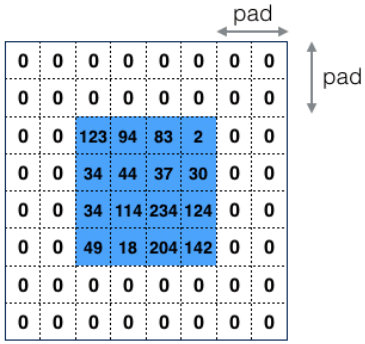
=





Stride

필터는 입력데이터를 지정한 간격으로 순회하면서 합성곱을 계산합니다. 이때 필터를 순회하는 간격을 Stride라고 합니다.

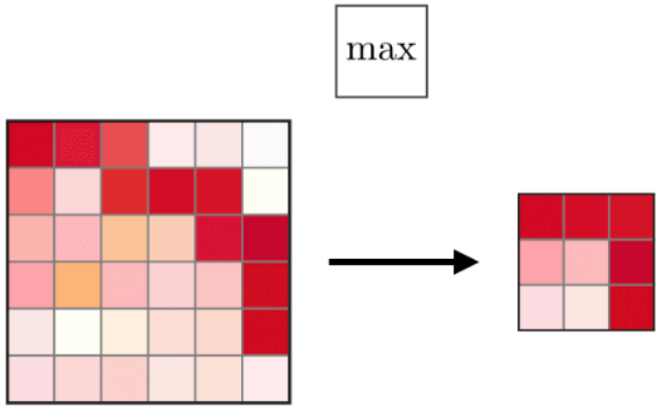
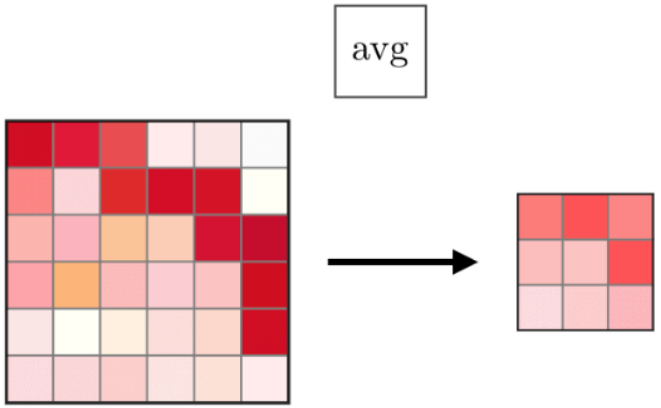


Convolution Neural Network

CONNECT THE PYTHONISTAS

Pooling Layer

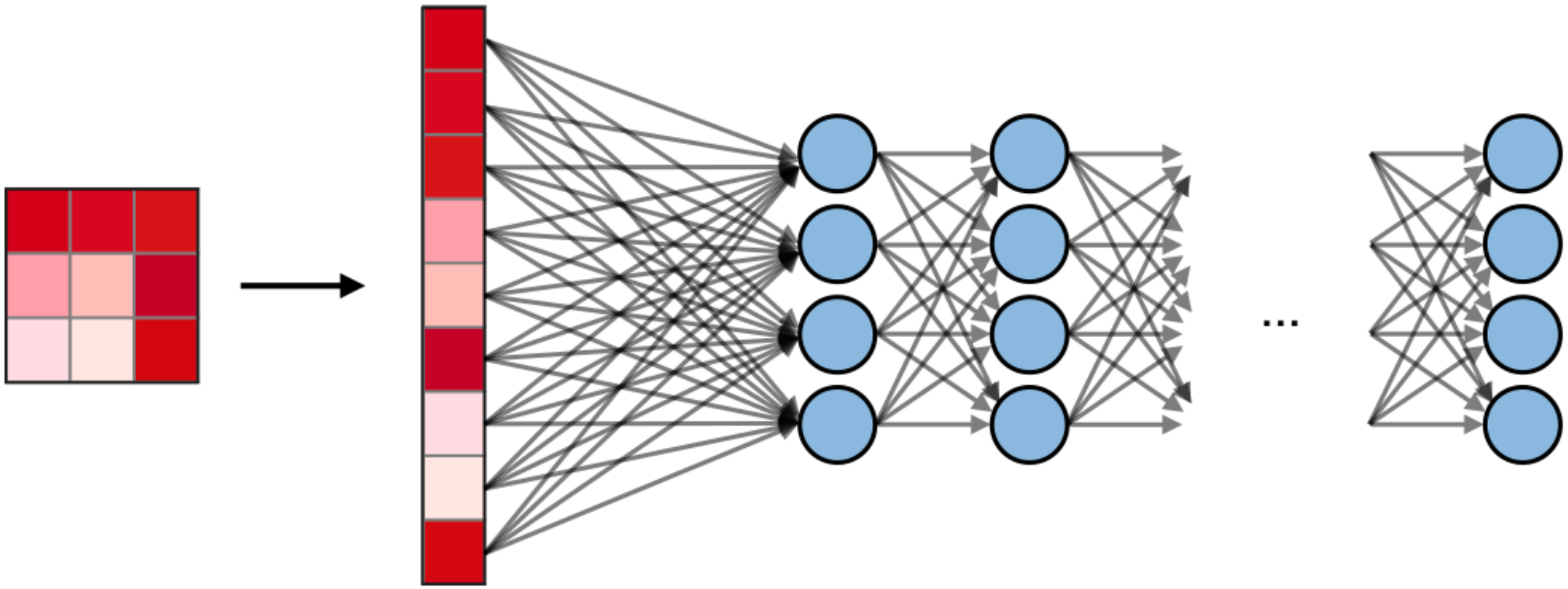
풀링레이어는 Convolution 레이어의 출력데이터를 입력으로 받아서 크기를 줄이거나 특정 데이터를 강조하는 용도로 사용합니다. 일반적으로 더 적은 차원의 매트릭스가 구성됩니다.

Type	Max pooling	Average pooling
Purpose	Each pooling operation selects the maximum value of the current view	Each pooling operation averages the values of the current view
Illustration		
Comments	• Preserves detected features • Most commonly used	• Downsamples feature map • Used in LeNet

Convolution Neural Network

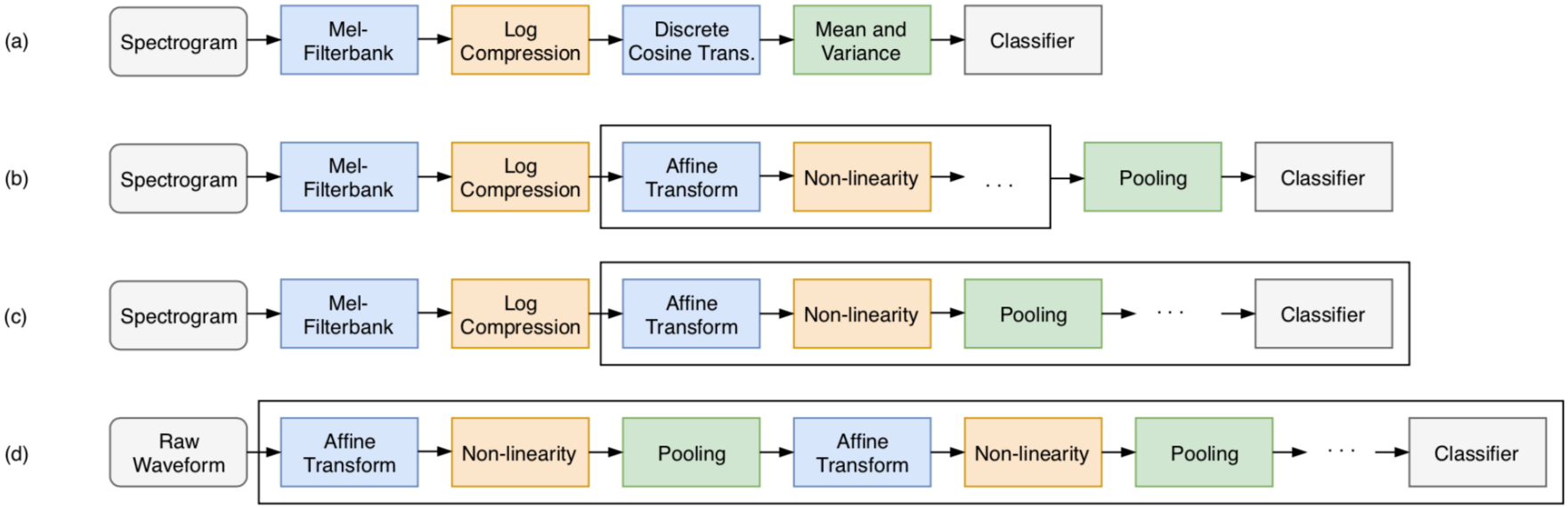
Fully Connected Layer

FC layer는 컨볼루션의 연산들의 ouput을 input으로 받아 flattened 하게 만든 후
일반적인 뉴럴 넷 연산을 적용해서 해당 Label을 분류하고 score를 매기게 됩니다



Music Classification

From Feature engineering to end-to-end learning



(a) MFCC + summarization + standard classifiers

(b) Mel-Spectrogram + RBM, AE, sparse coding (low-level feature learning)

(c) Mel-Spectrogram + Deep neural network

(d) Raw waveform + Deep neural network (end-to-end learning)

Music Classification

CNN Performance is better!

음악에서 사용하는 CNN 계열은 1-D CNN, 2-D CNN 그리고 마지막은 sample-level CNN 입니다. D,2D CNN의 경우에는, network를 더욱 flexible하게 만들어주는 효과가 있습니다. Flexibility를 향한 추구는 더 성공적인 Sample-level CNN 까지 이어지게 되었습니다.

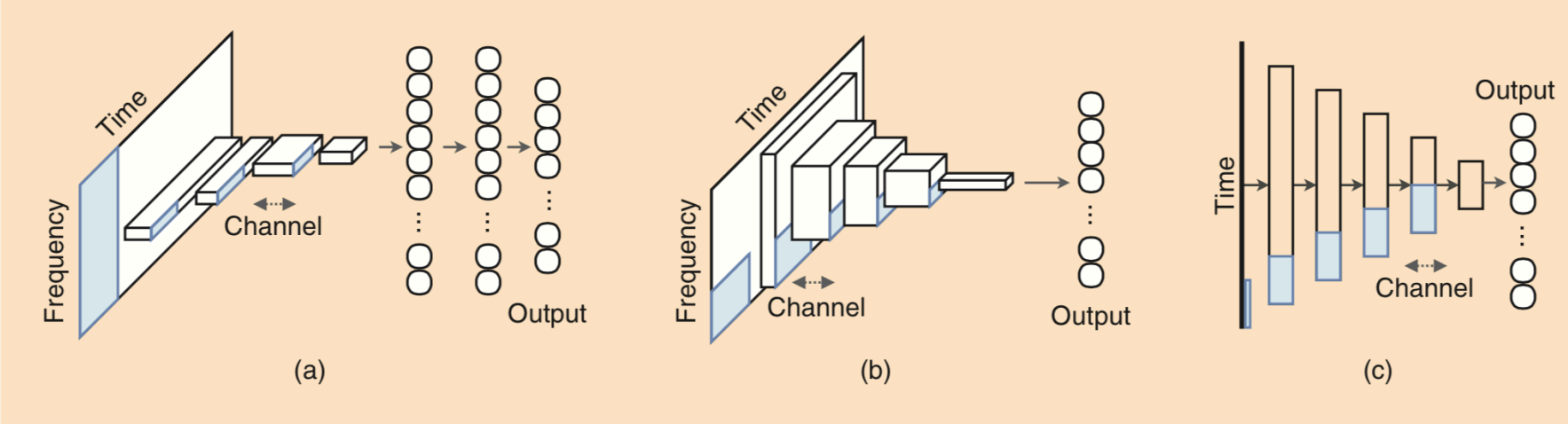


FIGURE 2. Block diagrams of (a) 1-D, (b) 2-D, and (c) sample-level CNNs. (a) and (b) are based on 2-D time–frequency representation inputs (e.g., mel-spectrograms or short-time Fourier transforms), and (c) is based on a time-series input.

- Frequency : F
- Time : T
- Channel : N
- strides : $(s1, s2)$

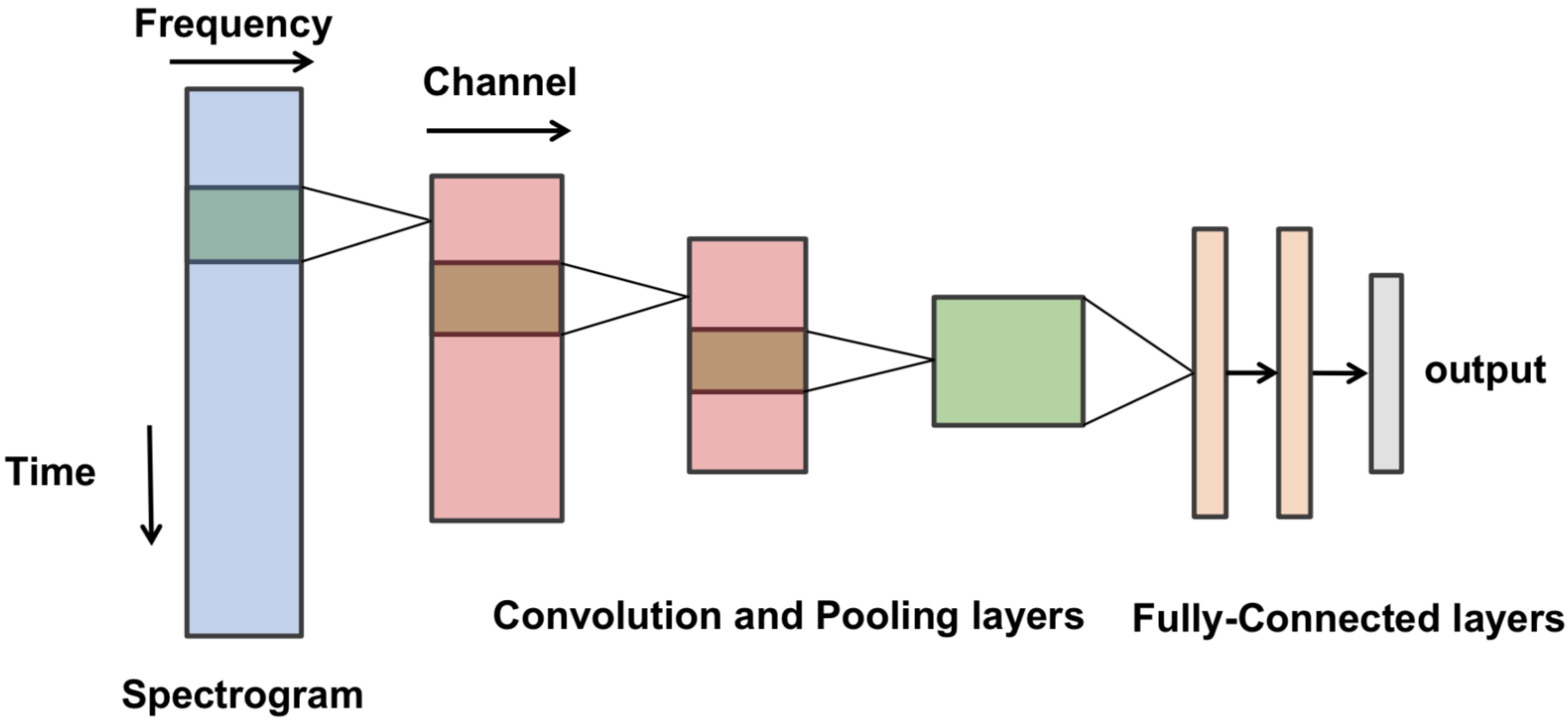
Input
- Spectrogram
- Log-Spectrogram: Mel or Constant-Q
- Raw Waveforms

Music Classification

1-D CNN

Convolution filter의 크기가 frequency 영역대는 고정되어 있으며, Time에 따라서 진행됩니다

- The 1D features map significantly reduces the number of parameters
- Fast to train
- Work well for small datasets

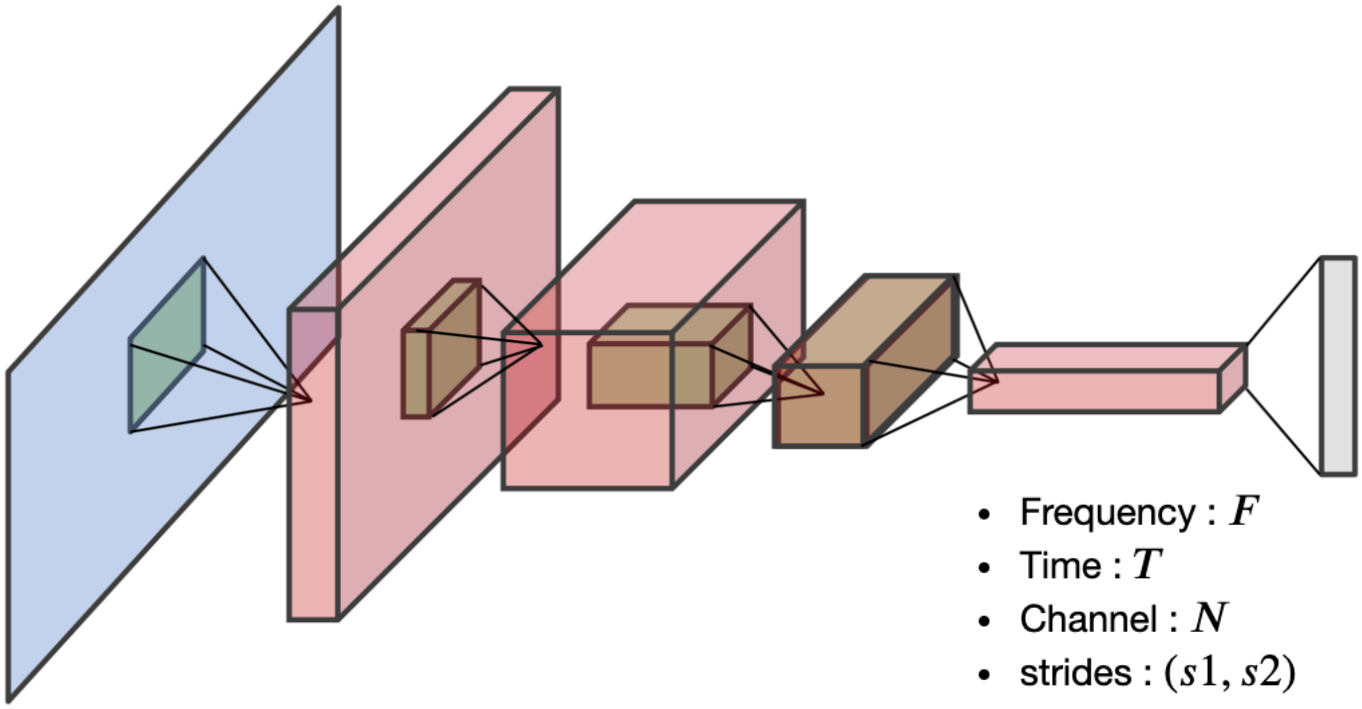


Music Classification

2-D CNN

Convolution filter의 크기가 frequency, Time 모두 가변적으로 진행됩니다.

- Advantage
  - Time-frequency 두가지 영역에서 pattern을 찾게 됩니다.
  - 2D CNN이 1D CNN에 비해 큰 데이터 셋에 대해서 높은 성능을 보입니다
  - Relatively invariant to pitch shifting

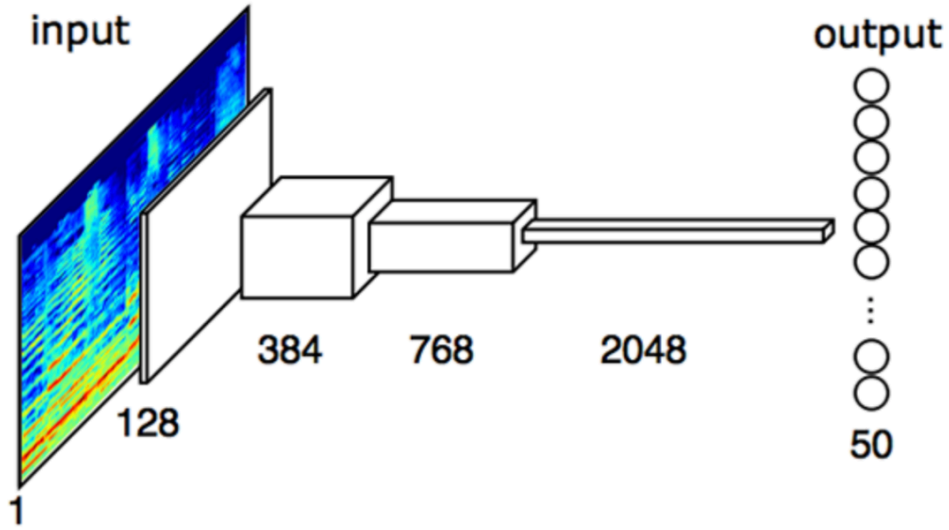


- Frequency : F
- Time : T
- Channel : N
- strides : $(s1, s2)$

Music Classification

2-D CNN

- Choi (2016)
  - VGGNet style



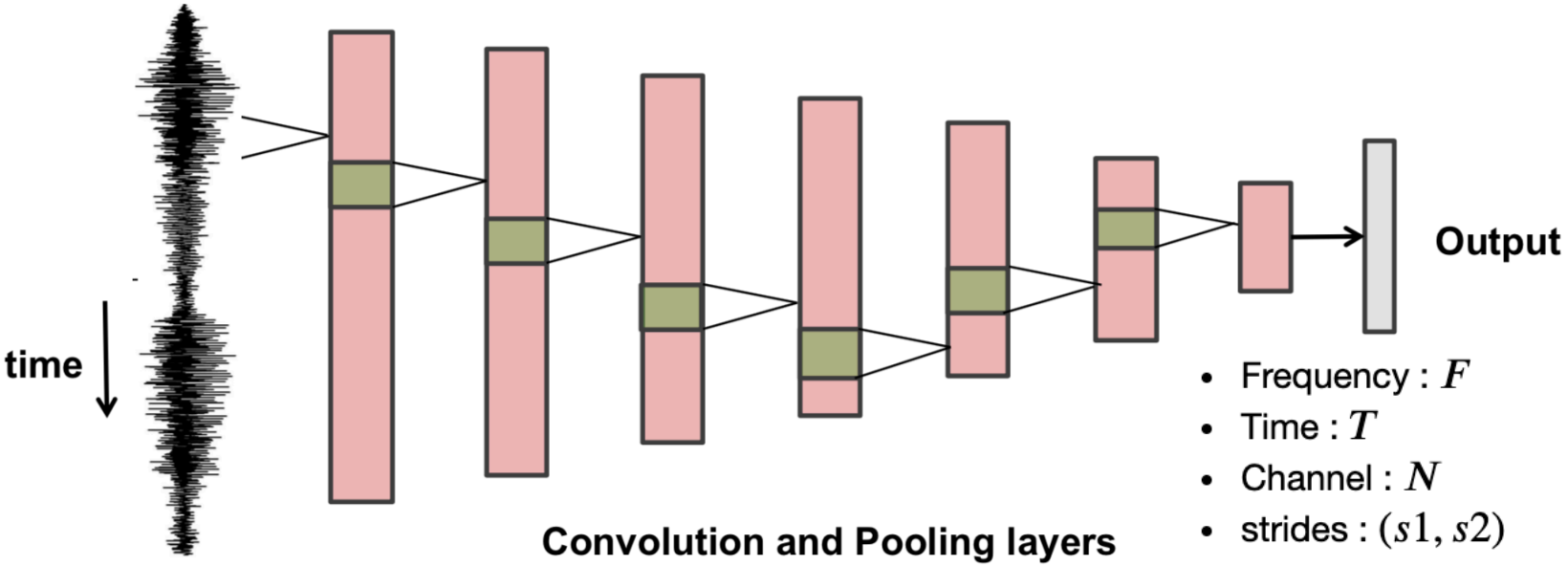
FCN-4
Mel-spectrogram *(input: 96×1366×1)*
Conv 3×3×128
MP (2, 4) *(output: 48×341×128)*
Conv 3×3×384
MP (4, 5) *(output: 24×85×384)*
Conv 3×3×768
MP (3, 8) *(output: 12×21×768)*
Conv 3×3×2048
MP (4, 8) *(output: 1×1×2048)*
Output 50×1 (sigmoid)

Music Classification

Sample CNN

Sample level CNN 의 가장 큰 특징은 바로 input데이터를 waveform
그 자체로 사용할 수 있다는 점입니다.

- Advantage
  - CNN이 "phase-invariant" representation을 반영한다는 점입니다.
  - 커널이 input signal에 대한 spectral bandwidth를 계산해 준다는 점입니다.
  - Convolution의 커널들이 harmonic component 을 반영해줍니다.

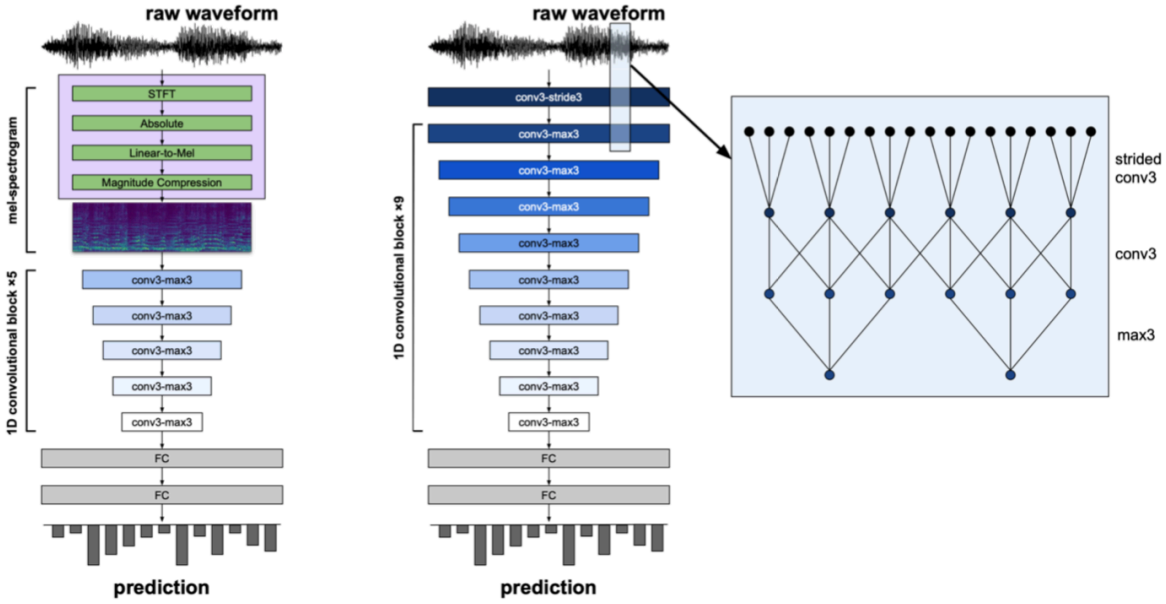


Music Classification

Sample CNN

Sample level CNN 의 가장 큰 특징은 바로 input데이터를 waveform
그 자체로 사용할 수 있다는 점입니다.

- Lee (2017)
  - 1D VGGNet



3^9 model, 19683 frames 59049 samples (2678 ms) as input			
layer	stride	output	# of params
conv 3-128	3	19683 × 128	512
conv 3-128 maxpool 3	1 3	19683 × 128 6561 × 128	49280
conv 3-128 maxpool 3	1 3	6561 × 128 2187 × 128	49280
conv 3-256 maxpool 3	1 3	2187 × 256 729 × 256	98560
conv 3-256 maxpool 3	1 3	729 × 256 243 × 256	196864
conv 3-256 maxpool 3	1 3	243 × 256 81 × 256	196864
conv 3-256 maxpool 3	1 3	81 × 256 27 × 256	196864
conv 3-256 maxpool 3	1 3	27 × 256 9 × 256	196864
conv 3-256 maxpool 3	1 3	9 × 256 3 × 256	196864
conv 3-512 maxpool 3	1 3	3 × 512 1 × 512	393728
conv 1-512 dropout 0.5	1 –	1 × 512 1 × 512	262656
sigmoid	–	50	25650
Total params			1.9×10^6

Tutorial!

Dataset

CONNECT THE PYTHONISTAS

GTZAN dataset

8개의 서로 다른 genres 가 있는 음악 30초짜리 데이터 셋입니다.
reggae, classical, country, jazz, metal, pop, disco, hiphop의 label을 가지고 있습니다.

Total Songs	730		
Average Length	~ 30 seconds		
Data Split	Train	Valid	Test
Amount	353	150	227
Total Genres	8 (classical, country, disco, hiphop, jazz, metal, pop, reggae)		

Workflow

CONNECT THE PYTHONISTAS

data augment.py

학습 데이터를 약간의 변화를 주며 늘려줍니다

data manager.py

데이터셋을 배치단위로 묶어줍니다.
Index기반으로 데이터를 호출

feature extraction.py

Frequency도메인을 학습 데이터로 사용할때 필요합니다.

model.py

nn.Module을 활용하여 뉴럴넷 아키텍쳐 설계

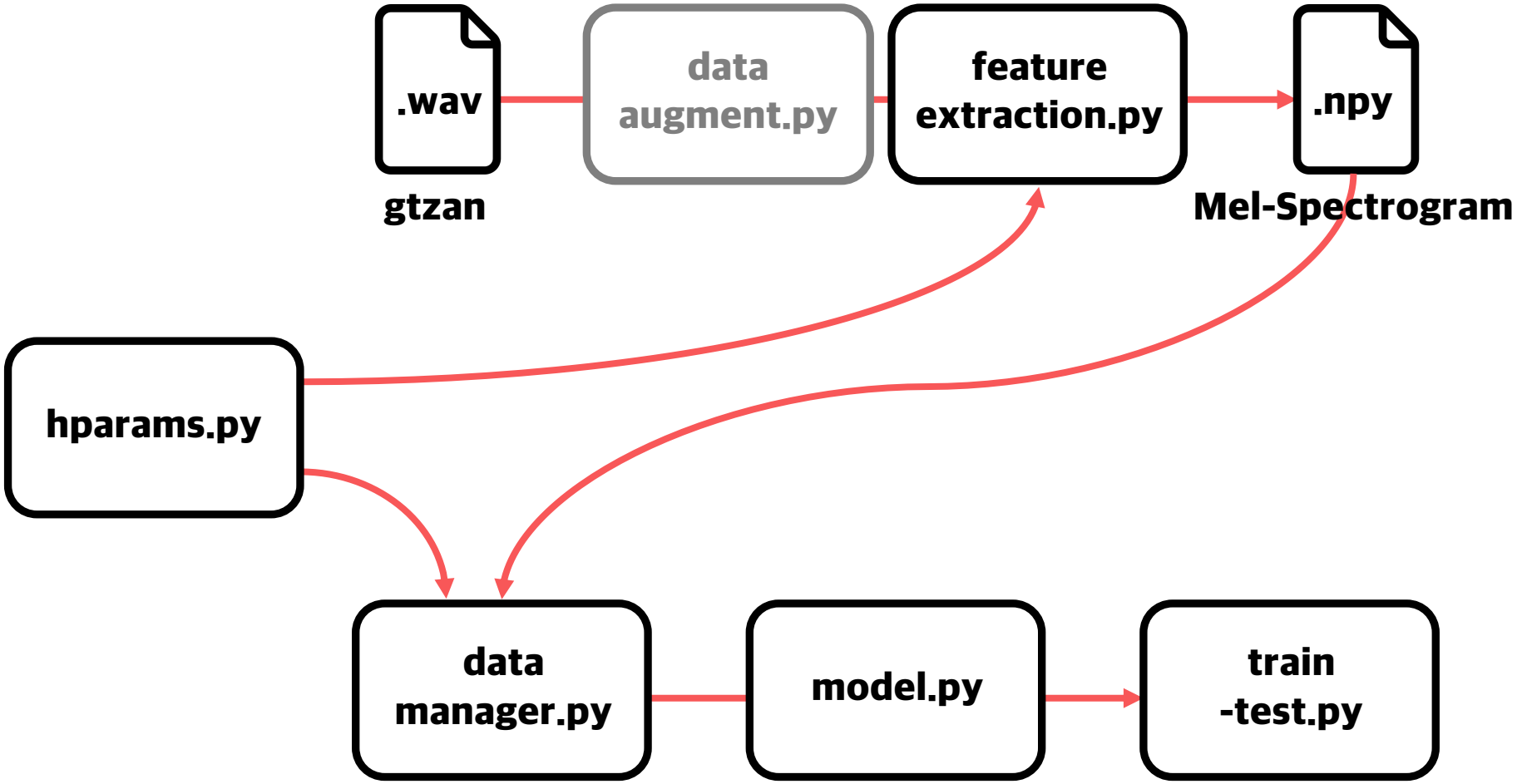
hparams.py

반복되는 실험을 도와주는 파라미터를 변경하는 문서입니다.

train -test.py

간단한 데이터셋 학습을 시켜보려고합니다.

Workflow


.wav
gtzan
data
augment.py
feature
extraction.py
.npy
Mel-Spectrogram
hparams.py
data
manager.py
model.py
train
-test.py

Music Classification

CONNECT THE PYTHONISTAS

Hparams

실험에 사용할 기본적인 하이퍼 파라미터를 저장,

1. 데이터 경로
2. STFT
3. Training 설정

```python
import argparse

class HParams(object):
    def __init__(self):
        self.dataset_path = './dataset/gtzan'
        self.feature_path= './dataset/feature_augment'
        self.genres =  ['classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop', 'reggae']

        # Feature Parameters
        self.sample_rate=22050
        self.fft_size = 1024
        self.win_size = 1024
        self.hop_size = 512
        self.num_mels = 128
        self.feature_length = 1024

        # Training Parameters
        self.device = 1  # 0: CPU, 1: GPU0, 2: GPU1, ...
        self.batch_size = 128
        self.num_epochs = 100
        self.learning_rate = 1e-2

    # Function for pasing argument and set hParams
    def parse_argument(self):
        parser = argparse.ArgumentParser()
        for var in vars(self):
            value = getattr(hparams, var)
            argument = '--' + var
            parser.add_argument(argument, type=type(value), default=value)
        args = parser.parse_args()
        for var in vars(self):
            setattr(hparams, var, getattr(args,var))

hparams = HParams()
hparams.parse_argument()
```

Music Classification

Feature_extraction.py

1. Data Load
2. STFT
3. mel-fliter
4. log-scale
5. Resizing (Music Length)

```python
import os
import numpy as np
import librosa
from hparams import hparams

def load_list(list_name, hparams):
    with open(os.path.join(hparams.dataset_path, list_name)) as f:
        file_names = f.read().splitlines()
        return file_names

def melspectrogram(file_name, hparams):
    y, sr = librosa.load(os.path.join(hparams.dataset_path, file_name), hparams.sample_rate)
    S = librosa.stft(y, n_fft=hparams.fft_size, hop_length=hparams.hop_size,
win_length=hparams.win_size)
    mel_basis = librosa.filters.mel(hparams.sample_rate, n_fft=hparams.fft_size,
n_mels=hparams.num_mels)
    mel_S = np.dot(mel_basis, np.abs(S))
    mel_S = np.log10(1+10*mel_S)
    mel_S = mel_S.T

    return mel_S

def resize_array(array, length):
    resize_array = np.zeros((length, array.shape[1]))
    if array.shape[0] >= length:
        resize_array = array[:length]
    else:
        resize_array[:array.shape[0]] = array
    return resize_array
```

Music Classification

CONNECT THE PYTHONISTAS

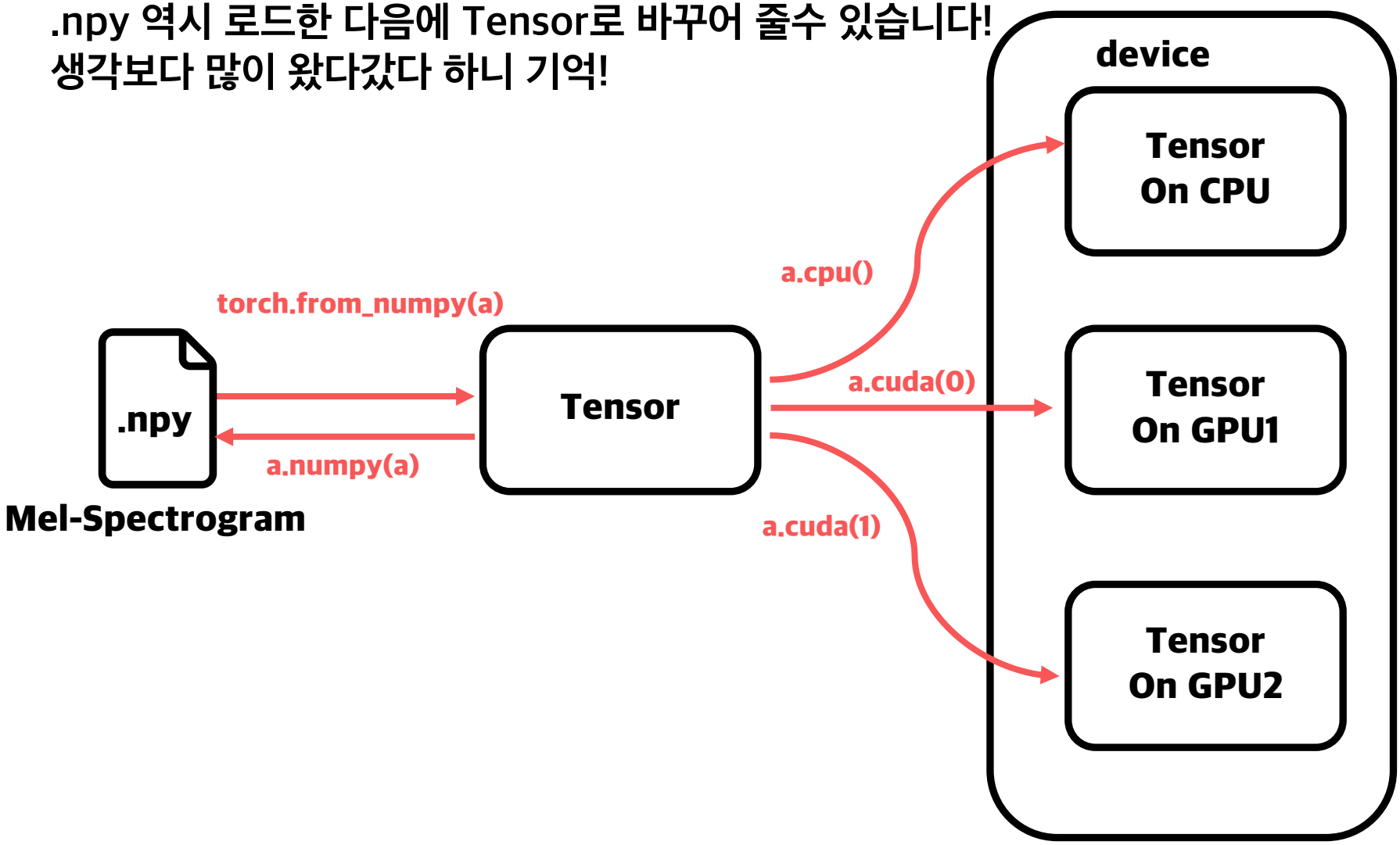
Feature_extraction.py

1. List 호출
2. 30초 음악 자르기
3. 데이터 .npy 저장

```python
def main():
    print("Extracting Feature")
    list_names = ['train_list.txt', 'valid_list.txt', 'test_list.txt']

    for list_name in list_names:
        set_name = list_name.replace('_list.txt', '')
        file_names = load_list(list_name, hparams)

        for file_name in file_names:
            feature = melspectrogram(file_name, hparams)
            feature = resize_array(feature, hparams.feature_length)

            # Data Arguments
            num_chunks = feature.shape[0]/hparams.num_mels
            data_chuncks = np.split(feature, num_chunks)

            for idx, i in enumerate(data_chuncks):
                save_path = os.path.join(hparams.feature_path, set_name, file_name.split('/')[0])
                save_name = file_name.split('/')[1].split('.wav')[0]+str(idx)+".npy"
                if not os.path.exists(save_path):
                    os.makedirs(save_path)

                np.save(os.path.join(save_path, save_name), i.astype(np.float32))
                print(os.path.join(save_path, save_name))

    print('finished')

if __name__ == '__main__':
    main()
```

.npy? Not Tensor?

.npy 역시 로드한 다음에 Tensor로 바꾸어 줄수 있습니다!
생각보다 많이 왔다갔다 하니 기억!



Music Classification

data_manager.py

데이터로드를 위한
Class를 만들고,

__getitem__
인덱스를 기반으로
데이터를 리턴

__len__
데이터 사이즈

get_label
Hparams에서
파일별 장르를
레이블로 달아줌

```python
import os
import numpy as np
from torch.utils.data import Dataset, DataLoader

class GTZANDataset(Dataset):
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __getitem__(self, index):
        return self.x[index], self.y[index]

    def __len__(self):
        return self.x.shape[0]

# Function to get genre index for the give file
def get_label(file_name, hparams):
    genre = file_name.split('.')[0]
    label = hparams.genres.index(genre)
    return label
```

Music Classification

CONNECT THE PYTHONISTAS

data_manager.py

Train, Test, Validation Set의 파일경로를 타고 각 .npy 데이터를 로드해서 list에 append, 그 후 np.stack

```
def load_dataset(set_name, hparams):
    x = []
    y = []
    dataset_path = os.path.join(hparams.feature_path, set_name)
    for root,dirs,files in os.walk(dataset_path):
        for file in files:
            data = np.load(os.path.join(root,file))
            label = get_label(file, hparams)
            x.append(data)
            y.append(label)
    x = np.stack(x)
    y = np.stack(y)

    return x,y
```

Music Classification

CONNECT THE PYTHONISTAS

data_manager.py

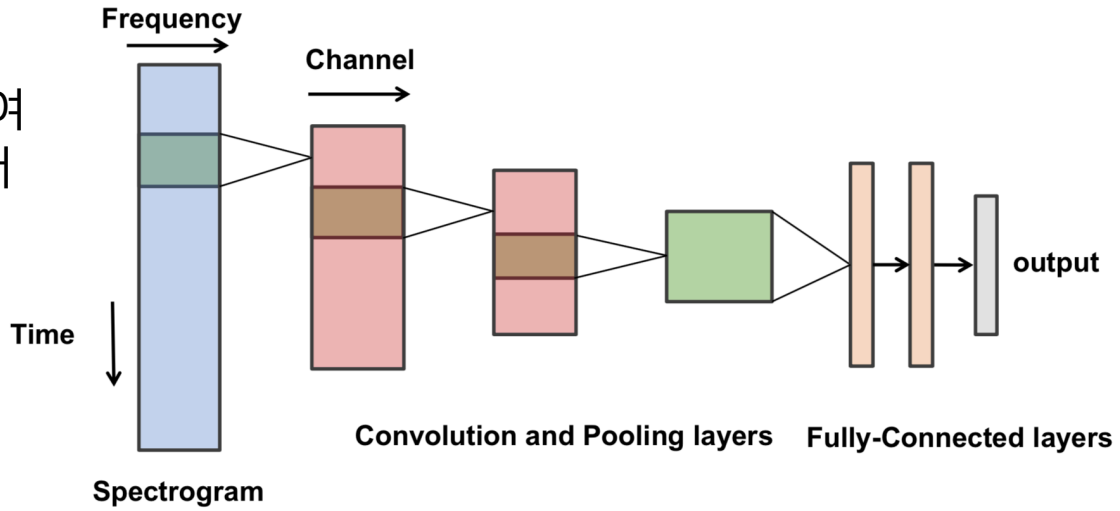
위에서 짠 load_dataset 함수를 통해서 각 데이터셋을 호출, Normalize과정을 거친후
GTZANDataset Class를 통해서 생성,
DataLoader를 통해서 Batch로 묶어서 데이터를 학습시킬 준비를 완료

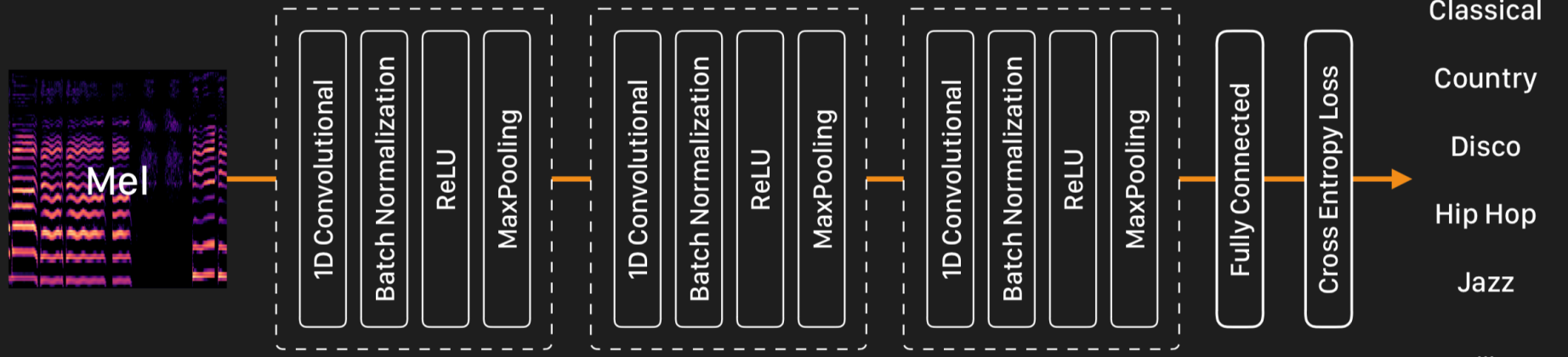
```
def get_dataloader(hparams):
    x_train, y_train = load_dataset('train', hparams)
    x_valid, y_valid = load_dataset('valid', hparams)
    x_test, y_test = load_dataset('test', hparams)

    mean = np.mean(x_train)
    std = np.std(x_train)
    x_train = (x_train - mean)/std
    x_valid = (x_valid - mean)/std
    x_test = (x_test - mean)/std

    train_set = GTZANDataset(x_train, y_train)
    vaild_set = GTZANDataset(x_valid, y_valid)
    test_set = GTZANDataset(x_test, y_test)

    train_loader = DataLoader(train_set, batch_size=hparams.batch_size, shuffle=True, drop_last=False)
    valid_loader = DataLoader(vaild_set, batch_size=hparams.batch_size, shuffle=False, drop_last=False)
    test_loader = DataLoader(test_set, batch_size=hparams.batch_size, shuffle=False, drop_last=False)

    return train_loader, valid_loader, test_loader
```

Music Classification

net.py

1D Convolution을 3번 반복하여
한번 음악의 시간 흐름에 따라서
학습을 진행해 봅시다.





Music Classification

net.py

Train_Test.py 에서 호출할 Model Class를 만들어 보자!
기왕이면 객체를 생성할 때, Deep Learning 모형이 바로 생성되면 좋다. **(init 메서드)**

Model을 생성할 때, 모든 layer에 대해서 공통으로 적용되는 것은 **init 메서드**의 input으로 받는다.
(eg. activation, weight initialization 방법)

```python
import torch
torch.manual_seed(123)
import torch.nn as nn

class Baseline(nn.Module):
    def __init__(self, hparams):
        super(Baseline, self).__init__()
        self.conv0 = nn.Sequential(
            nn.Conv1d(hparams.num_mels, 32, kernel_size=8, stride=1, padding=0),
            nn.BatchNorm1d(32),
            nn.ReLU(),
            nn.MaxPool1d(8, stride=8)
        )
        self.conv1 = nn.Sequential(
            nn.Conv1d(32, 32, kernel_size=8, stride=1, padding=0),
            nn.BatchNorm1d(32),
            nn.ReLU(),
            nn.MaxPool1d(8, stride=8)
        )
        self.conv2 = nn.Sequential(
            nn.Conv1d(32, 64, kernel_size=4, stride=1, padding=0),
            nn.BatchNorm1d(64),
            nn.ReLU(),
            nn.MaxPool1d(4, stride=4))
        self.linear = nn.Sequential(
            nn.Linear(192, 64),
            nn.ReLU(),
            nn.Dropout(),
            nn.Linear(64, len(hparams.genres)))
        self.apply(self._init_weights)


    def forward(self, x):
        x = x.transpose(1, 2)
        x = self.conv0(x)
        x = self.conv1(x)
        x = self.conv2(x)
        x = x.view(x.size(0), x.size(1)*x.size(2))
        x = self.linear(x)
        return x

    def _init_weights(self, layer) -> None:
        if isinstance(layer, nn.Conv1d):
            nn.init.kaiming_uniform_(layer.weight)
        elif isinstance(layer, nn.Linear):
            nn.init.xavier_uniform_(layer.weight)
```

Music Classification

train_test.py

data_manager, model, hparmas를 따로 모듈화를 해둔 이유는 모델의 아키텍쳐나 Feautre extraction을 수정하면서 실험하기 때문입니다.
동일한 조건에서도 init값과 관련없이 동일한 결과를 위해서는 torch.manual_seed와 torch.backends.cudnn.deterministic을 활용합니다.

```
import torch
torch.manual_seed(1234)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
import numpy as np
np.random.seed(0)
import data_manager
from model import *
from hparams import hparams
```

train_test.py

__init__
- model
- Criterion
- Optimizer
- Learning rate

accuracy
- metric

Run (Train + Eval)
train, test 상황을 나눠줍니다

```python
class Runner(object):
    def __init__(self, hparams):
        self.model = net(hparams)
        self.criterion = torch.nn.CrossEntropyLoss()
        self.optimizer = torch.optim.SGD(self.model.parameters(), lr=hparams.learning_rate)
        self.learning_rate = hparams.learning_rate
        self.device = torch.device("cpu")

        if hparams.device > 0:
            torch.cuda.set_device(hparams.device - 1)
            self.model.cuda(hparams.device - 1)
            self.criterion.cuda(hparams.device - 1)
            self.device = torch.device("cuda:" + str(hparams.device - 1))

    # Accuracy function works like loss function in PyTorch
    def accuracy(self, source, target):
        source = source.max(1)[1].long().cpu()
        target = target.long().cpu()
        correct = (source == target).sum().item()

        return correct/float(source.size(0))

    # Running model for train, test and validation. mode: 'train' for training, 'eval' for validation
and test
    def run(self, dataloader, mode='train'):
        self.model.train() if mode is 'train' else self.model.eval()

        epoch_loss = 0
        epoch_acc = 0
        for batch, (x, y) in enumerate(dataloader):
            x = x.to(self.device)
            y = y.to(self.device)

            prediction = self.model(x)
            loss = self.criterion(prediction, y.long())
            acc = self.accuracy(prediction, y.long())

            if mode is 'train':
                loss.backward()
                self.optimizer.step()
                self.optimizer.zero_grad()

            epoch_loss += prediction.size(0)*loss.item()
            epoch_acc += prediction.size(0)*acc

        epoch_loss = epoch_loss/len(dataloader.dataset)
        epoch_acc = epoch_acc/len(dataloader.dataset)

        return epoch_loss, epoch_acc
```

Music Classification

CONNECT THE PYTHONISTAS

train_test.py

내가 GPU가 세팅된 환경에서 작업이 이루어질 경우를 대비해서 device_name 함수를 지정해 둡시다. 나중에 Hparmas에서 관련 정보를 받아 올 수 있습니다.

```
def device_name(device):
    if device == 0:
        device_name = 'CPU'
    else:
        device_name = 'GPU:' + str(device - 1)

    return device_name
```

Music Classification

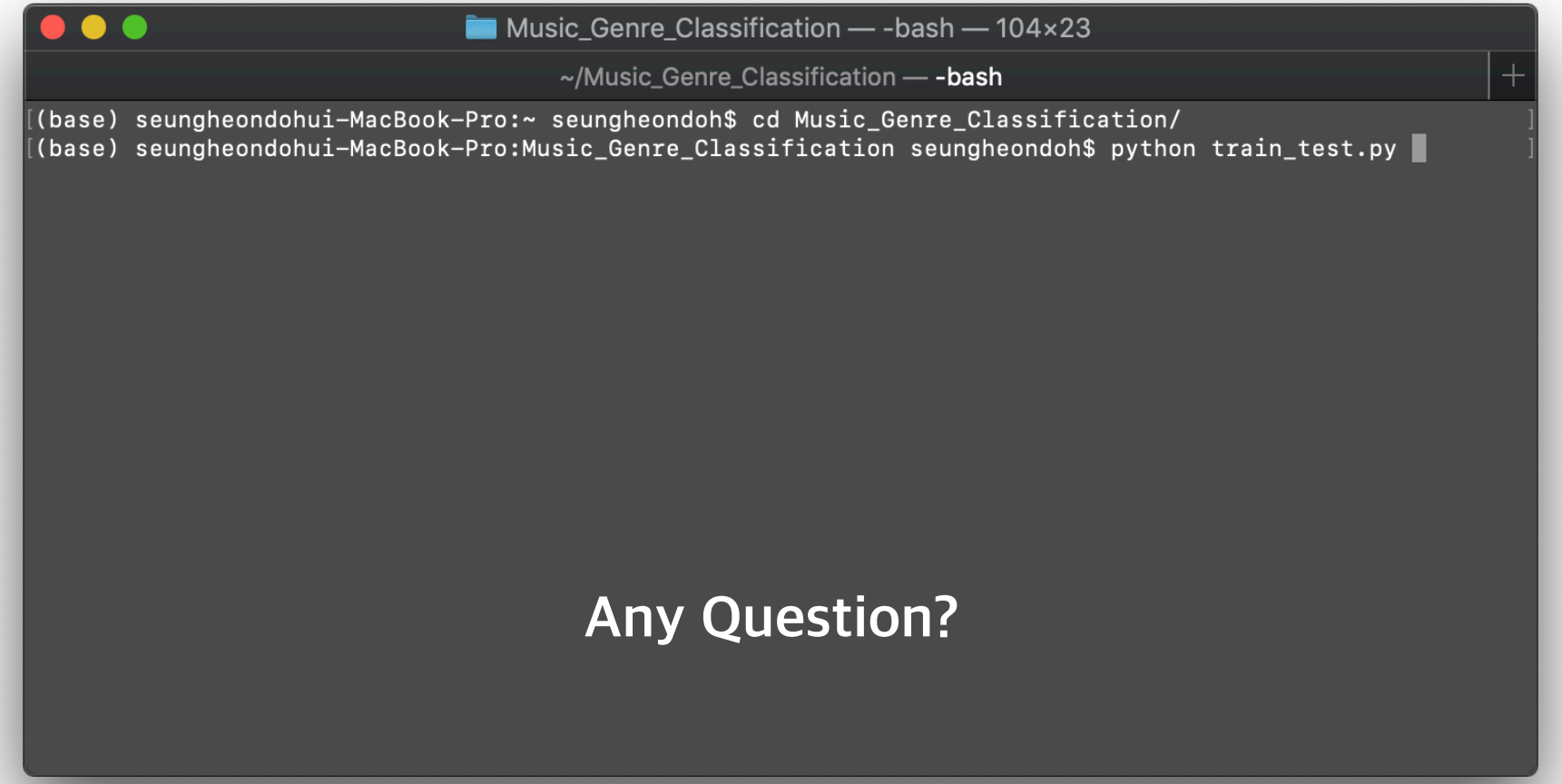
CONNECT THE PYTHONISTAS

train_test.py

Epoch를 반복하면서 학습이 overfitting 여부를 체크하거나 early stopping을 추가로 넣어줄 수도 있습니다.

```
def main():
    train_loader, valid_loader, test_loader = data_manager.get_dataloader(hparams)
    runner = Runner(hparams)

    print('Training on ' + device_name(hparams.device))
    for epoch in range(hparams.num_epochs):
        train_loss, train_acc = runner.run(train_loader, 'train')
        valid_loss, valid_acc = runner.run(valid_loader, 'eval')

        print("[Epoch %d/%d] [Train Loss: %.4f] [Train Acc: %.4f] [Valid Loss: %.4f] [Valid Acc: %.4f]"
%
              (epoch + 1, hparams.num_epochs, train_loss, train_acc, valid_loss, valid_acc))
    test_loss, test_acc = runner.run(test_loader, 'eval')
    print("Training Finished")
    print("Test Accuracy: %.2f%%" % (100*test_acc))

if __name__ == '__main__':
    main()
```

Music Classification



오늘 발표를 마무리하며

CONNECT THE PYTHONISTAS

- Music and Deep Learning에 대한 배경지식을 복습
- 집에가서 오늘 짠 아키텍쳐를 템플릿으로 다른 작업도 해보자 (다른 데이터셋 or 다른 뉴럴넷 아키텍쳐)
- Pytorch의 기본적인 기능들을 복습해보자
- 뒤에나오는 레퍼런스를 공부해보자

Reference

KAIST, NYU, CMU | Music, Speech Signal Processing 자료

http://mac.kaist.ac.kr/~juhan/gct634/

http://www.nyu.edu/classes/bello/Teaching.html

http://www.speech.cs.cmu.edu/15-492/

Music Classification 관련 자료

https://www.slideshare.net/JunKim22/endtoend-music-classification-96586946

CNN 관련 자료

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

남기현님 Audio for Deep Learning (너무 좋습니다!!)

https://www.facebook.com/groups/soundly/permalink/1147149912117007/

Thank you

TORCHAUDIO TUTORIAL

PyTorch is an open source deep learning platform that provides a seamless path from research prototyping to production deployment with GPU support.

Significant effort in solving machine learning problems goes into data preparation. torchaudio leverages PyTorch's GPU support, and provides many tools to make data loading easy and more readable. In this tutorial, we will see how to load and preprocess data from a simple dataset.

For this tutorial, please make sure the `matplotlib` package is installed for easier visualization.

```
import torch
import torchaudio
import matplotlib.pyplot as plt
```

Opening a dataset

torchaudio supports loading sound files in the wav and mp3 format. We call waveform the resulting raw audio signal