


Speech Recognition


이다경

kaggle

Q


CompetitionsDatasetsKernelsDiscussionJobs...



Featured Prediction Competition

TensorFlow Speech Recognition Challenge

Can you build an algorithm that understands simple speech commands?

Google Brain · 1,315 teams · 18 days ago

\$25,000

Prize Money

OverviewDataKernelsDiscussionLeaderboardRulesTeamMy SubmissionsLate Submission

Overview

Description

Evaluation


Prizes

Timeline

Tutorials & More Info

We might be on the verge of too many screens. It seems like everyday, new versions of common objects are “re-invented” with built-in wifi and bright touchscreens. A promising antidote to our screen addiction are voice interfaces.

But, for independent makers and entrepreneurs, it’s hard to build a simple speech detector using free, open data and code. Many voice recognition datasets require preprocessing before a neural network model can be built on them. To help with this, [TensorFlow](#) recently released the Speech Commands Datasets. It



<https://www.kaggle.com/c/tensorflow-speech-recognition-challenge>



Big Voice

우리 팀 서기는 내 컴퓨터! 화자인식과
음성인식을 이용한 회의 Dialogue

김 강 열 / 박 이 삭 / 이 다 경 / 이 태 권 / 조 혜 진

Open Research

Ideas for 9th Kaggle TensorFlow Speech Recognition Challenge

■ Open Sources & Books & Conferences

kaggle

competition

speech

Open
Research

OpenResearch.ai

1 18일 전

Tensorflow Speech Recognition Challenge

짧은 명령어를 이해하는 단순하고 효과적인 모델을 두고 경쟁하는 캐글 컴피션입니다. yes, no, up, down 등과 같은 10개의 명령어를 구분(classification)해야 합니다. 거기에 추가적인 2개의 Class가 있다는 점이 특이한데,

- silence : 아무 소리도 안나는 경우의 class
- unknown : 10개의 클래스가 아닌 소리. out of vocabulary로 생각하면 됨.

의 2가지가 있습니다.

트레이닝 데이터는 65000개 쯤의 wav파일과 label로 이루어져 있습니다. 파일명에 화자 정보가 Hash 값으로 들어있습니다. 즉 파일명을 보면 같은 사람인지 다른 사람의 목소리인지는 알 수 있습니다.

테스트 데이터는 16만개가 넘는 wav파일로 이루어져 있으며 이를 브라우저 제출하는 방식으로 대회가 구성되어 있습니다.

음성인식 챌린지 9위

<http://openresearch.ai/t/ideas-for-9th-kaggle-tensorflow-speech-recognition-challenge/105>

음성 관련 코드

```
import librosa
```

```
import librosa.display
```

```
import scipy
```

```
from scipy import signal
```

```
from IPython.display import Audio
```

```
from pydub import AudioSegment
```

```
from scipy.io import wavfile
```

음성 관련 코드

오디오 파일 불러오기

```
y, sr = librosa.load(path, sr = 16000)
```

```
type(y)
```

```
numpy.ndarray
```

```
ipd.Audio(y, rate=sr)
```



```
import IPython.display as ipd
```

```
y3 = AudioSegment.from_file(path)
```

```
type(y3)
```

```
pydub.audio_segment.AudioSegment
```

```
y3
```



```
sr4, y4 = wavfile.read(path)
```

```
print(sr4)
```

```
print(type(y4))
```

```
16000
```

```
<class 'numpy.ndarray'>
```

```
ipd.Audio(y4, rate=sr4)
```



음성 관련 코드

오디오 파일 불러오기

```
y, sr = librosa.load(path, sr = 16000)
```

y

```
array([ 6.10351562e-05,  3.66210938e-04,  6.71386719e-04, ...,  
       -1.09863281e-03, -8.23974609e-04, -6.71386719e-04], dtype=float32)
```

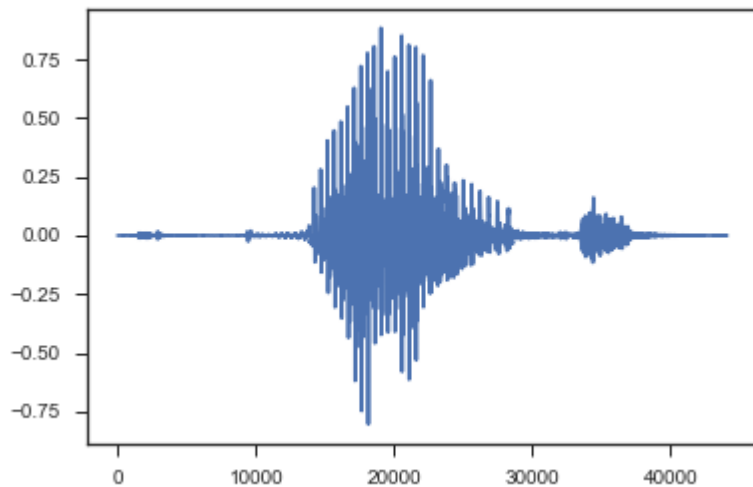
numpy 형식, 시간별 진폭 값

진폭이란? 주기적으로 진동하는 파의 진동

```
import matplotlib.pyplot as plt
```

```
plt.plot(y)
```

sample rate : 이산적인 신호를 만들기 위해 연속적인 신호에서 얻어진
단위시간 당 샘플링 횟수.
sr = 44100 : 1초에 44100 번 샘플링
default = 22050



음성 관련 코드

오디오 파일 불러오기

librosa → AudioSegment

```
def librosa2AudioSegment(y, sr):  
    samples = AudioSegment(y.tobytes(), frame_rate=sr, sample_width=y.dtype.itemsize, channels=1)  
    return samples
```

AudioSegment → librosa(numpy)

```
def AudioSegment2librosa(y):  
    samples = y.get_array_of_samples()  
    samples = np.array(samples)  
    samples = librosa.util.normalize(samples)  
    return samples
```

음성 관련 코드 - feature

SPECTROGRAM

```
frequencies, times, spectrogram = signal.spectrogram(y, sr)
plt.imshow(spectrogram.T, aspect='auto', origin='lower')
```

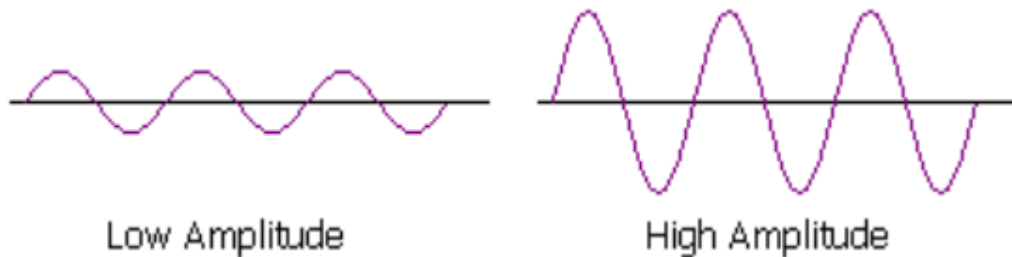
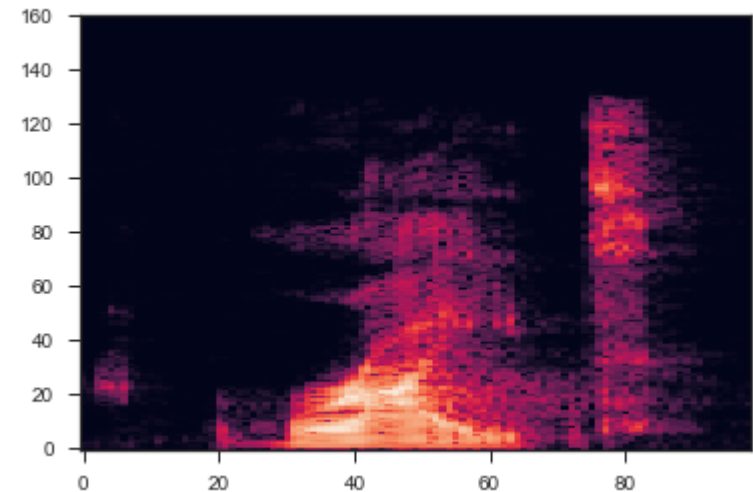
스펙트로그램(Spectrogram)이란?

Time, Frequency, Amplitude를 각 축으로 하는 3-dimensional
음성 표현

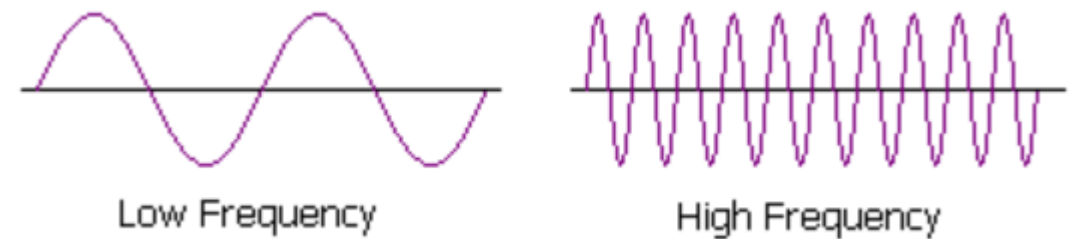
Time: 시간

Amplitude: 진폭

Frequency: 주파수(단위: Hz)



[Amplitude]



[Frequency]

음성 관련 코드 - feature

SPECTROGRAM

frequencies, times, spectrogram = signal.spectrogram(y, sr)

time domain : 시간에 따른 진폭의 변화

frequency domain(spectrum) : 주파수에 따른 진폭의 변화

spectrogram : 시간과 주파수에 따른 진폭의 변화

Low Amplitude

High Amplitude

[Amplitude]

Low Frequency

High Frequency

[Frequency]

음성 관련 코드 - feature

MFCC(Mel Frequency Cepstral Coefficient)

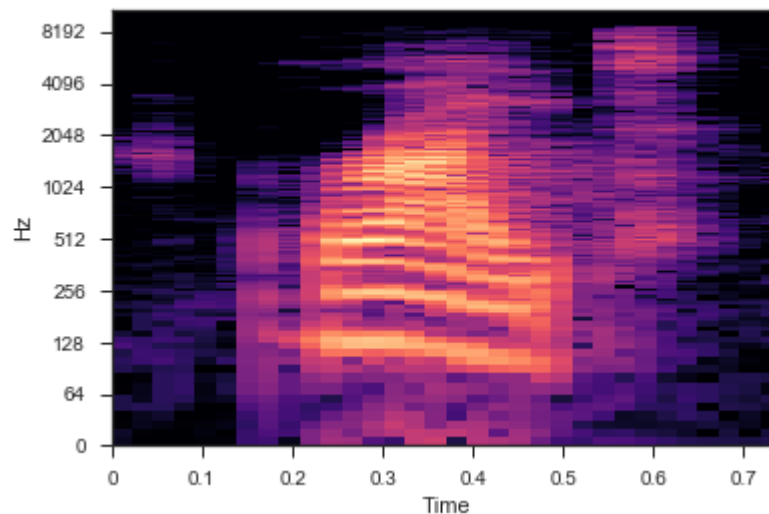
```
D = librosa.stft(y)
```

```
log_power = librosa.logamplitude(D**2, ref_power= np.max)
```

```
librosa.display.specshow(log_power, x_axis='time', y_axis="log")
```

MFCC 란?

인간이 소리를 듣는 방식을 반영해서 추출한 음성 특징 벡터



음성 관련 코드 - feature

```
D = librosa.stft(y)
log_power = librosa.logamplitude(D**2, ref_power= np.max)

librosa.display.specshow(log_power, x_axis='time', y_axis="log")
```

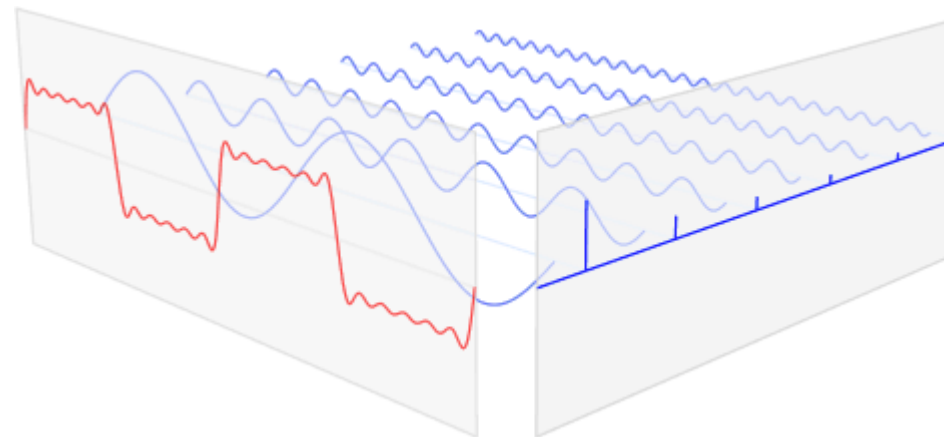
Short-time Fourier Transform (STFT)

임의의 입력 신호를 다양한 주파수를 갖는 주기함수들의 합으로 분해하여 표현하는 것

sin, cos 등의 삼각함수

→ 고주파부터 저주파까지 다양한 주파수 대역의 sin, cos 함수들로 원본 신호를 분해하는 것

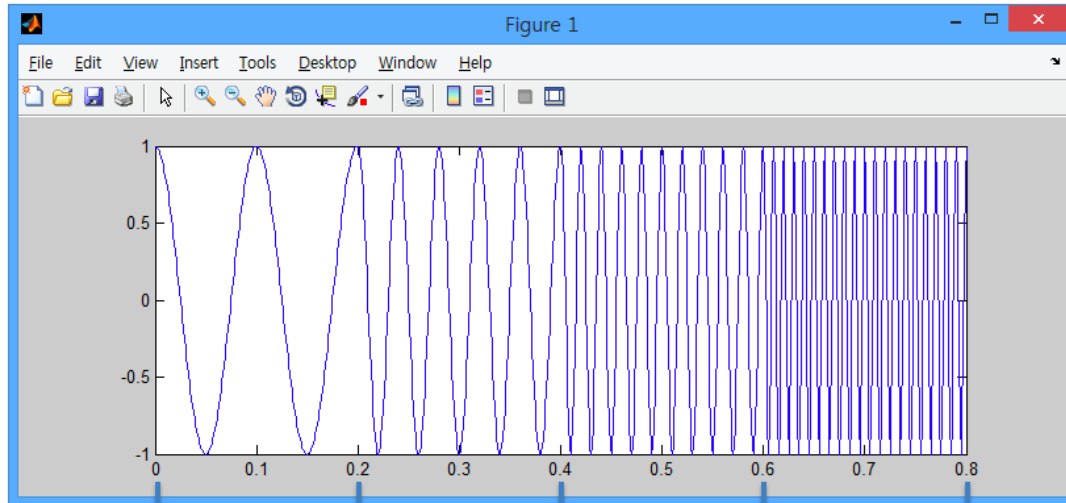
→ Time Domain에서 Frequency Domain으로 Domain을 변경하는 것



[Fourier transform]

음성 관련 코드 - feature

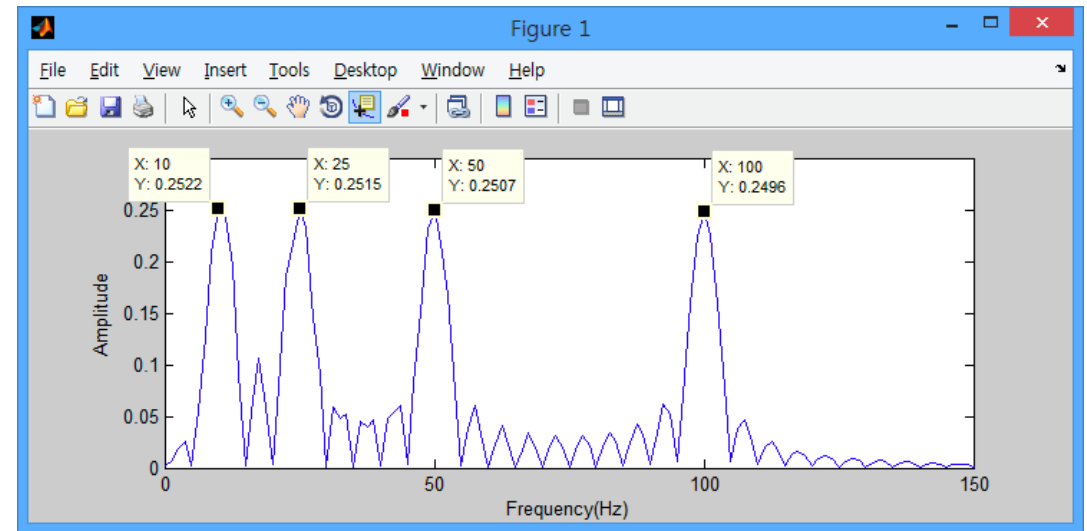
Short-time Fourier Transform (STFT)



[Time Domain]



[Frequency Domain]

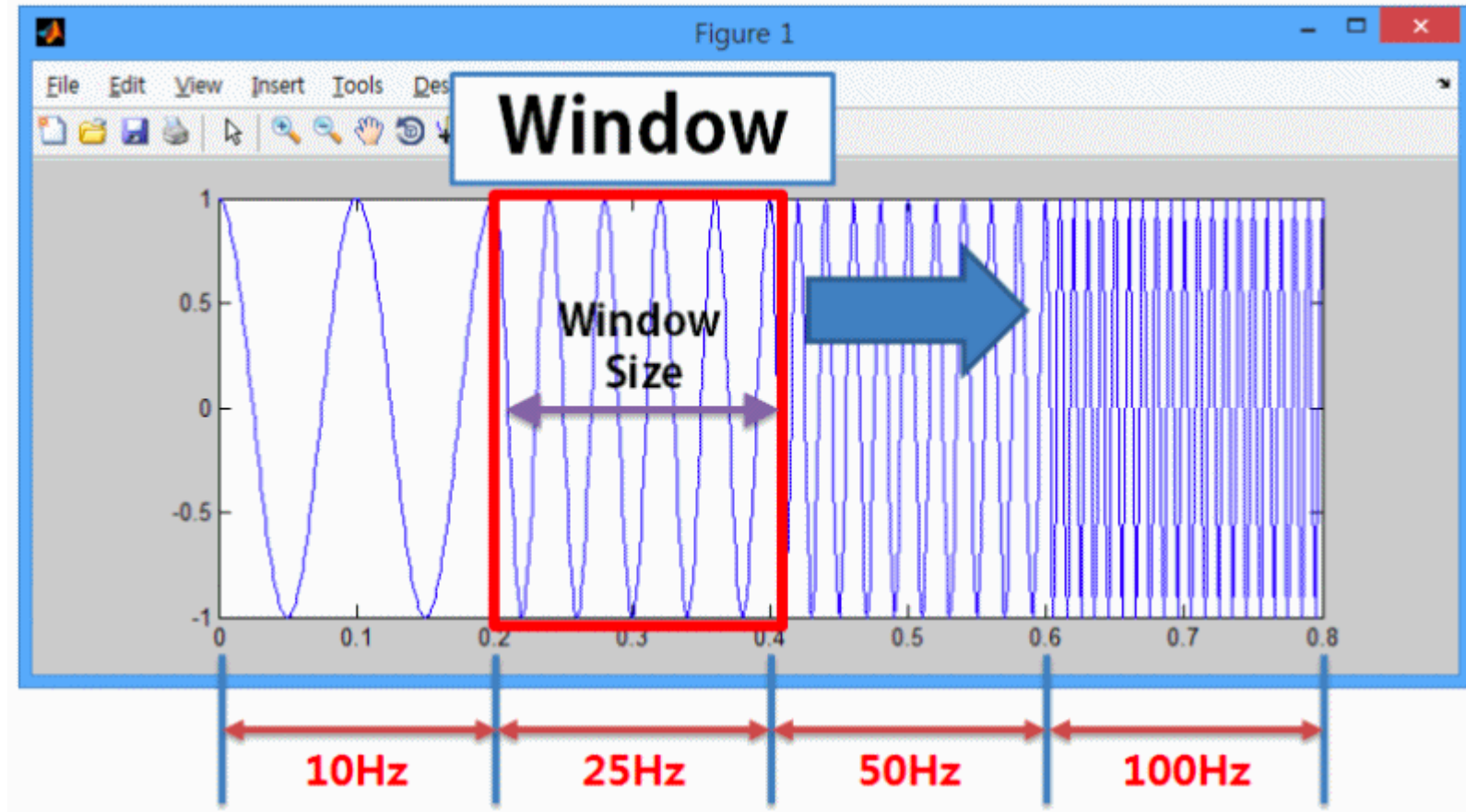


But Frequency Domain에는 시간 정보가 사라짐!

→ Short-time Fourier Transform (STFT)

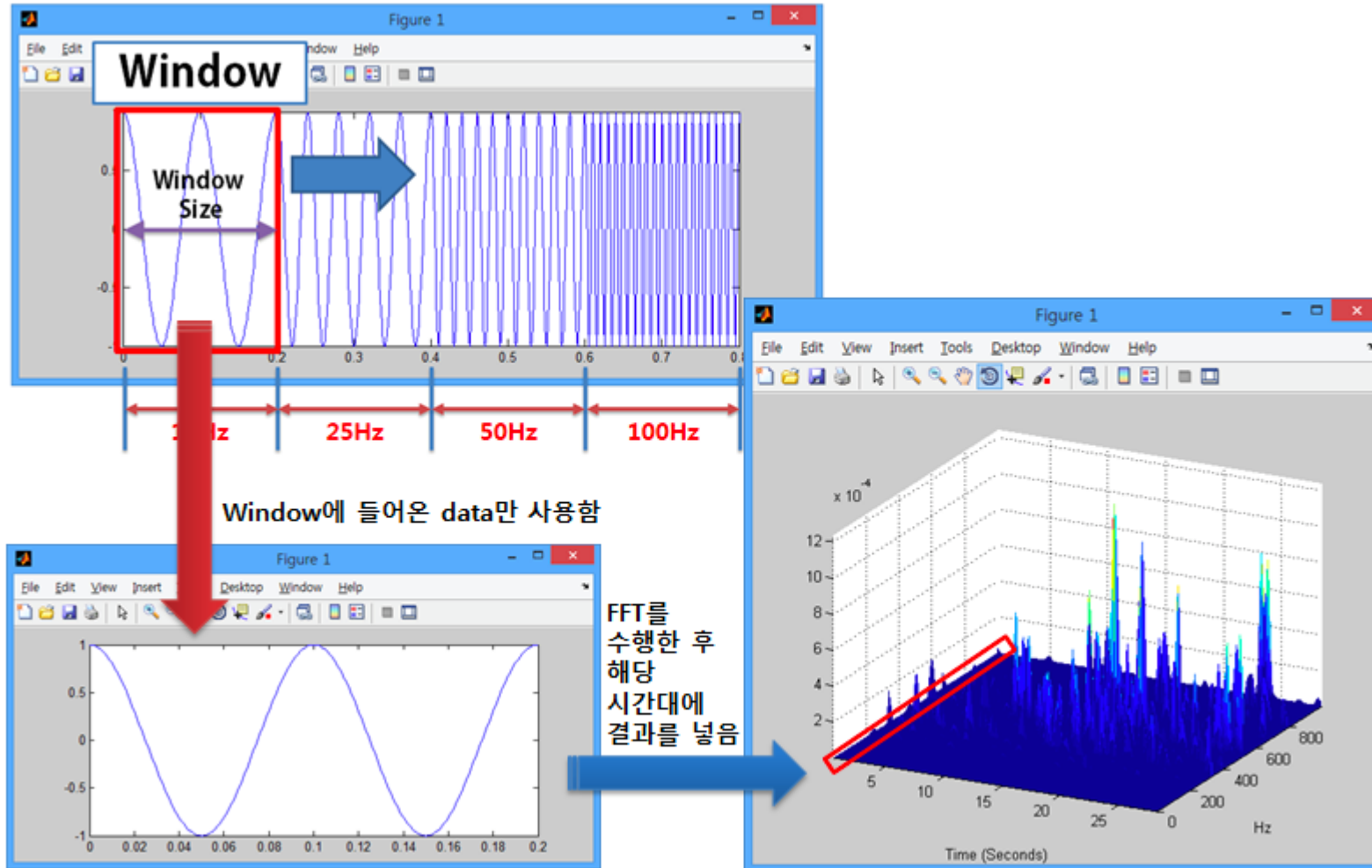
음성 관련 코드 - feature

Short-time Fourier Transform (STFT)



음성 관련 코드 - feature

Short-time Fourier Transform (STFT)



음성 관련 코드 - feature

```
D = librosa.stft(y)
log_power = librosa.logamplitude(D**2, ref_power= np.max)

librosa.display.specshow(log_power, x_axis='time', y_axis="log")
```

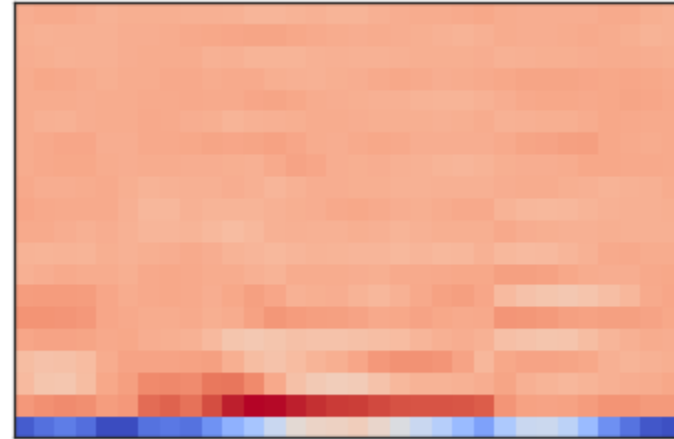
인간은 실제로 고주파보다 저주파의 소리를 더 잘 들음
따라서 log를 씌워 인간이 실제로 듣는 소리와 비슷하게 하기 위함



음성 관련 코드 - feature

MFCC(Mel Frequency Cepstral Coefficient)

```
MFCC = librosa.feature.mfcc(y=y, sr=sr)  
librosa.display.specshow(MFCC)
```



음성 관련 코드

Speed up

```
def speedUp(y, n_step = 2):  
    y_D = librosa.stft(y)  
    y_D_fast = librosa.phase_vocoder(y_D, n_step)  
    y_faster = librosa.istft(y_D_fast)  
    return y_faster
```

Speed down

```
def speedDown(y, n_step=0.5):  
    y_D = librosa.stft(y)  
    y_D_slow = librosa.phase_vocoder(y_D, n_step)  
    y_slower = librosa.istft(y_D_slow)  
    return y_slower
```

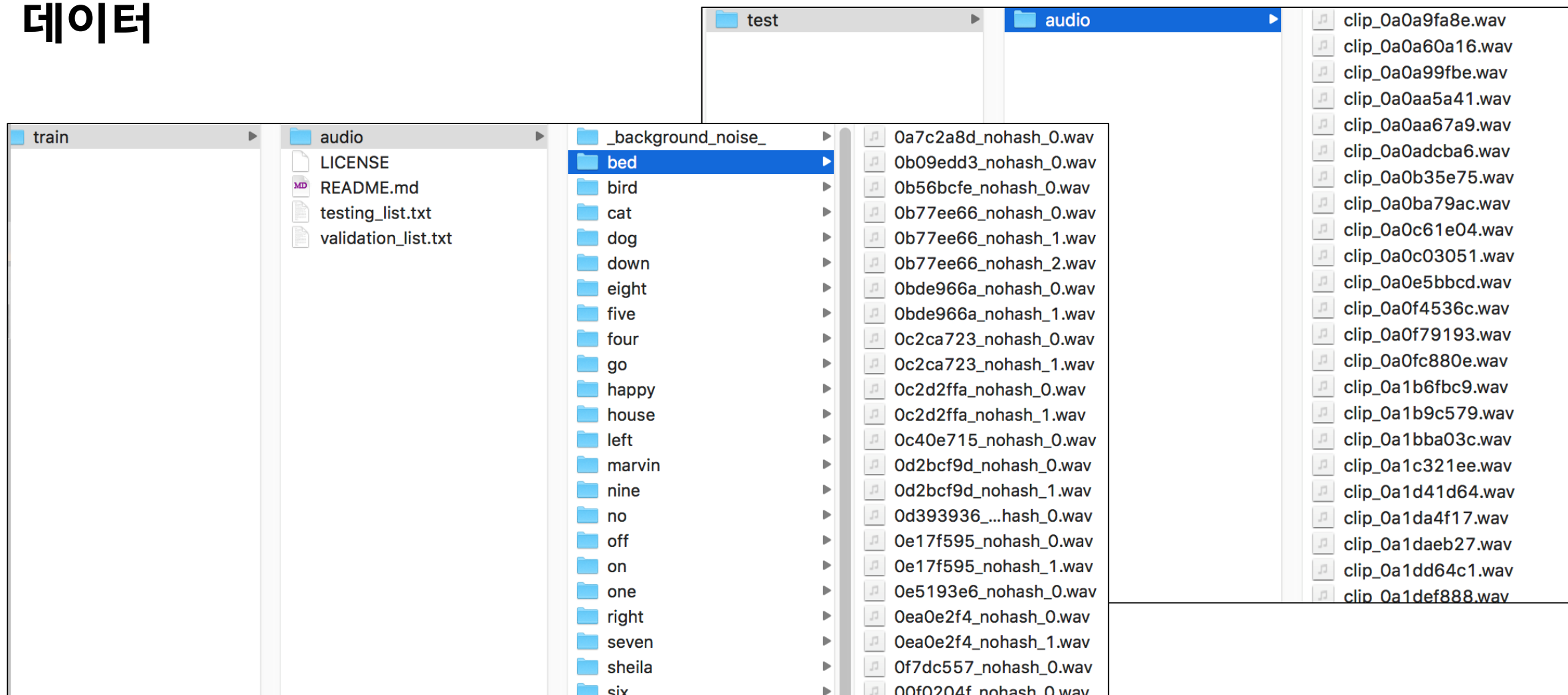
Pitch up

```
def pitchUp(y, sr = 44100, n_step = 10):  
    y_pitch_higher = librosa.effects.pitch_shift(y, sr, n_steps=n_step)  
    return y_pitch_higher
```

Pitch down

```
def pitchDown(y, sr = 44100, n_step = -10):  
    y_pitch_lower = librosa.effects.pitch_shift(y, sr, n_steps=n_step)  
    return y_pitch_lower
```

데이터



train

단어 별로 폴더, wav파일,

파일 이름에 화자정보가 Hash로

test

train에서 validation set 나누기

파일명을 통해서 화자 ID 확인 가능

→ train과 validation 나눌 때 같은 화자는 한 쪽 set에만 존재하도록

→ overfitting을 막기 위함일 듯

데이터 처리

1. 볼륨조절

70%~130% 수준까지 볼륨 조절. 일정 확률로 작은 소리(30%~50%)로도 조절

```
song = AudioSegment.from_mp3(path)
_song = song + 20
```

2. 위치조절 ??

소리의 peak 볼륨으로부터 10% 수준이면 배경음이라고 생각하고, 배경음인 부분 만큼만 shift해서 위치 조절.

즉 같은 소리지만, 소리가 빨리 나거나, 늦게 나는 등의 조절.

3. stretch 조절

소리를 늘리거나 빠르게 만듦.

```
y_D = librosa.stft(y)
y_D = librosa.phase_vocoder(y_D, n_step)
y_D = librosa.istft(y_D)
```

4. Pitch 조절

pitch를 높이거나 줄여서 음의 높낮이를 변경.

```
y_pitch = librosa.effects.pitch_shift(y, sr, n_steps=n_step)
```



5. 노이즈 추가

- 화이트노이즈, 브라운노이즈, 핑크노이즈, 블루노이즈, ...
- 트레이닝셋에 있는 배경음을 임의로 삽입

```
def noise_synthsis_path(audio_path, noise_path):
    audio = AudioSegment.from_file(audio_path)
    noise = AudioSegment.from_file(noise_path)
    combine = audio.overlay(noise)
    return combine
```

CNN Architecture

VGG16

80%

*Input은
spectrogram

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as “conv<receptive field size>-<number of channels>”. The ReLU activation function is not shown for brevity.

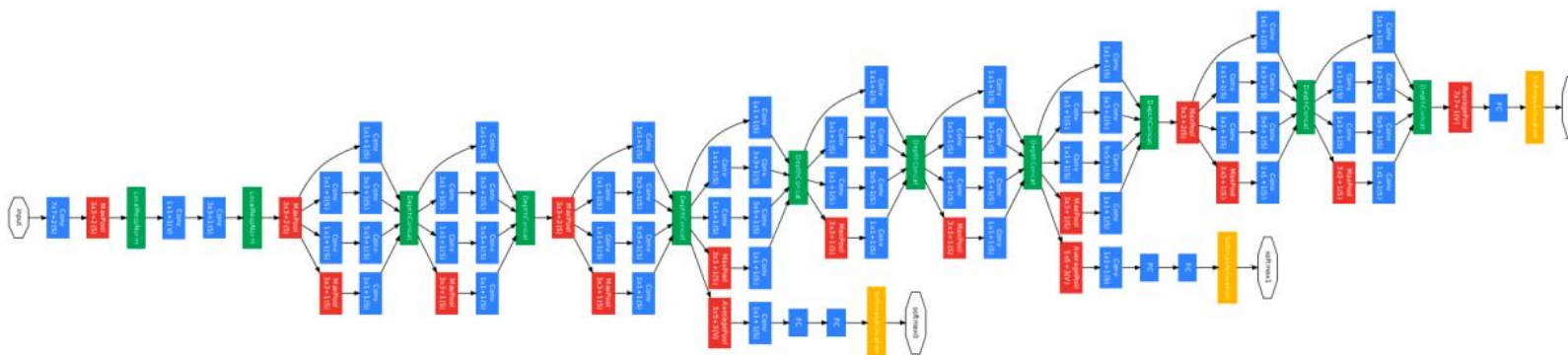
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

<http://openresearch.ai/t/vggnet-very-deep-convolutional-networks-for-large-scale-visual-recognition/29>

CNN Architecture

Inception v4

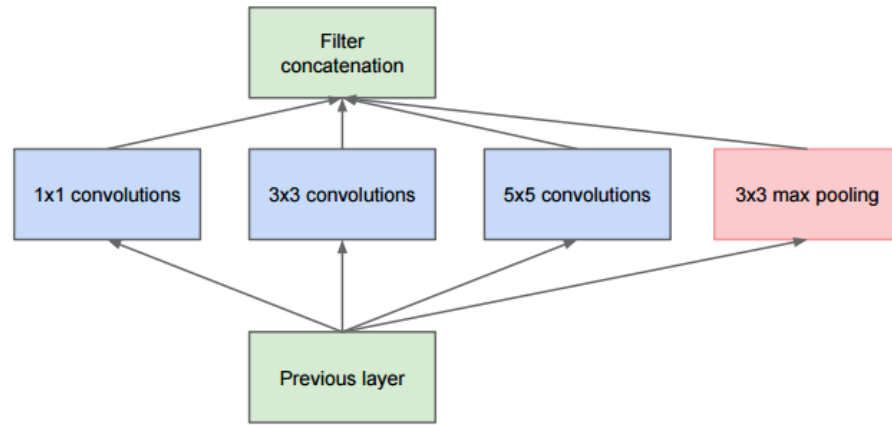
84%



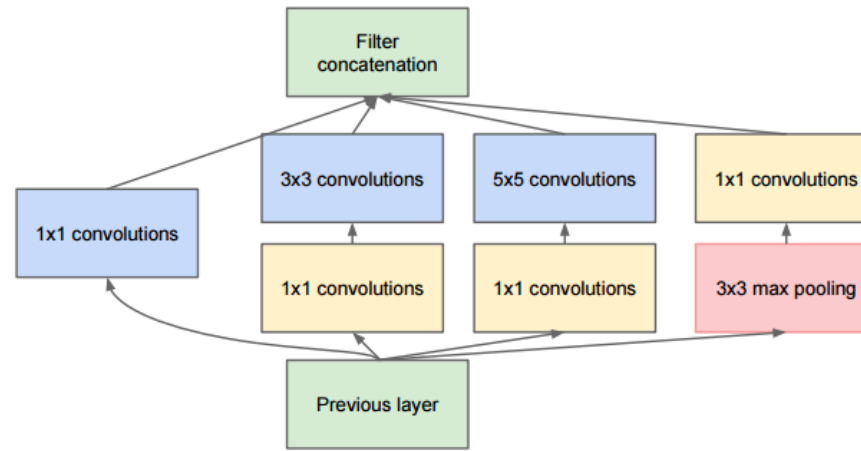
https://norman3.github.io/papers/docs/google_inception.html

CNN Architecture - Inception v1

- 같은 layer에서 서로 다른 크기를 갖는 conv filter를 적용하여 다른 scale의 feature를 얻을 수 있도록 함
- 1 x 1 conv를 적절히 사용하여 차원을 줄이고 망이 깊어졌을 때 연산량이 늘어나는 문제를 해결



(a) Inception module, naïve version



(b) Inception module with dimension reductions

더 성능 높이기

1. High Resolution Mel Spectrogram **87% ~ 88%**

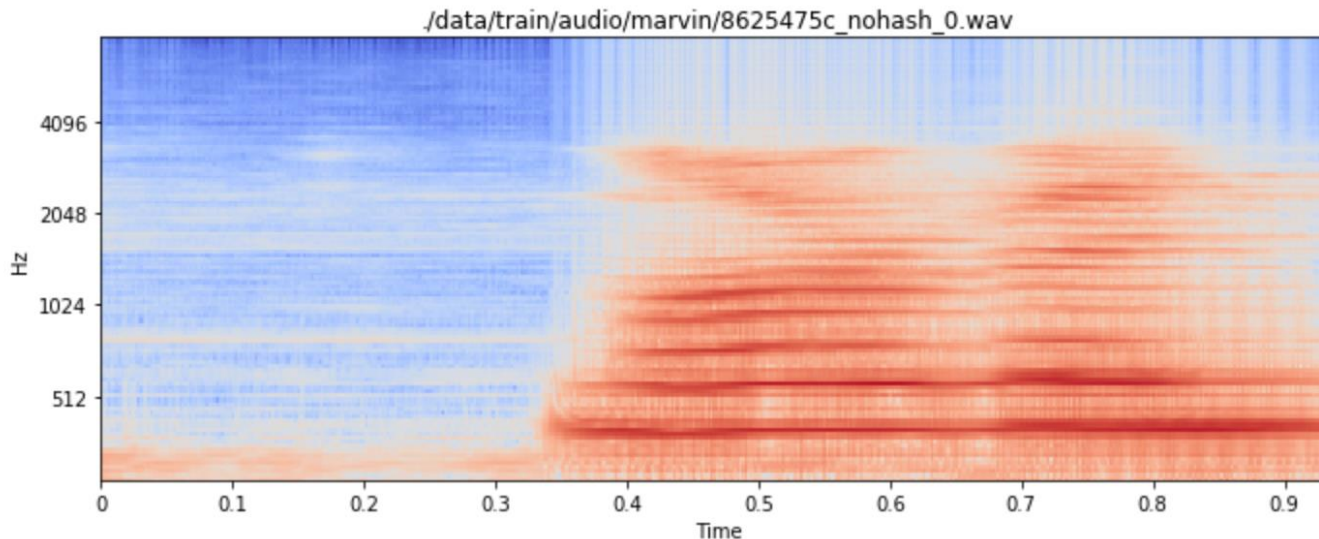
CNN에 Input으로 사용하는 spectrogram을 더 정교하게 뽑자!

spectrogram을 추출할 때, parameter값들을 다양하게 하여 뽑은 피쳐들을 적절히 합쳐

고해상도 spectrogram 추출

→ 고해상도 이기 때문에 담고있는 정보가 많음

→ on, off, up과 같은 클래스들을 더 잘 분류하게 됨

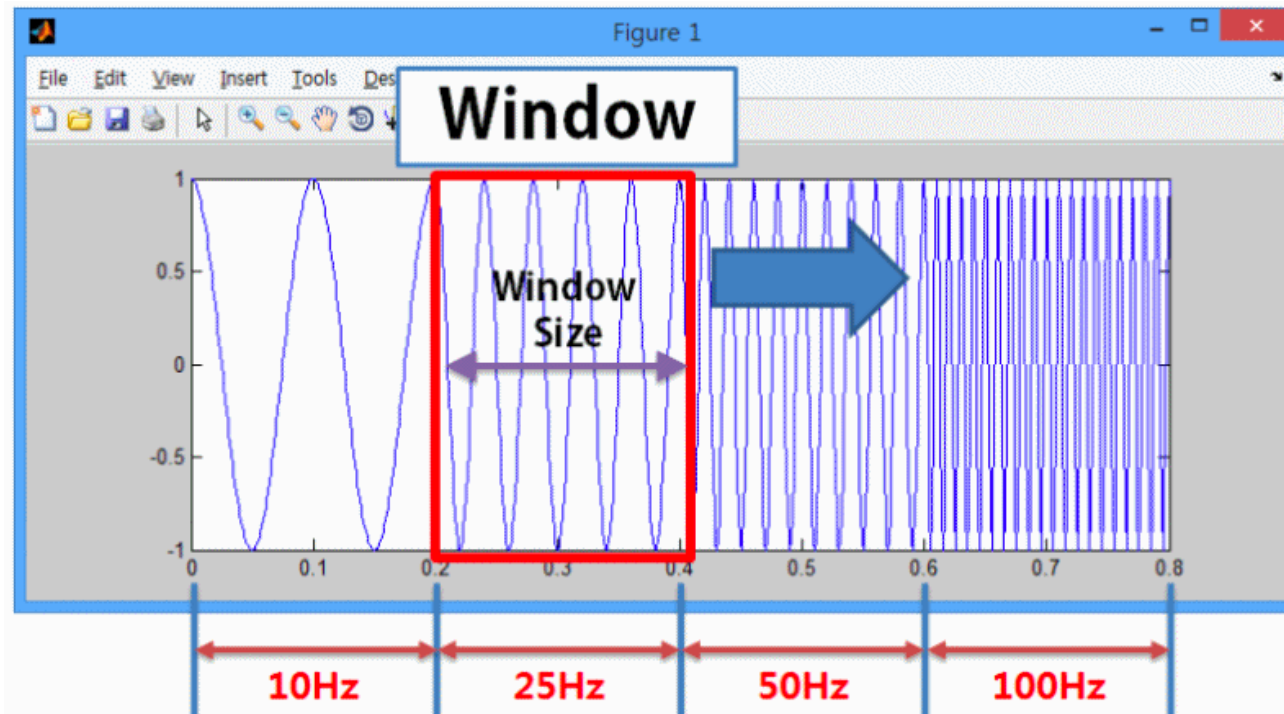


더 성능 높이기

1. High Resolution Mel Spectrogram

Fourier transform 을 할 때 window사이즈가 작으면 시간에 대한 정보는 크지만, 주파수의 폭은 부족, window사이즈가 크면 주파수의 폭은 적절하지만 시간에 대한 정밀도는 부족할 것.

→ 다양한 window사이즈의 푸리에 변환을 실행하고 Mel가중치를 적용하여 결합



더 성능 높이기

1. High Resolution Mel Spectrogram

Fourier transform 을 할 때 window사이즈가 작으면 시간에 대한 정보는 크지만, 주파수의 폭은 부족, window사이즈가 크면 주파수의 폭은 적절하지만 시간에 대한 정밀도는 부족할 것.

→ 다양한 window사이즈의 푸리에 변환을 실행하고 Mel가중치를 적용하여 결합

Mel scale(mel : melody에서 유래)

Convert Frequency to Mel Scale : $M(f) = 1125 \ln(1 + \frac{f}{700})$

Hz	20	160	394	670	1000	1420	1900	2450	3120	4000	5100	6600	9000	14000
mel	0	250	500	750	1000	1250	1500	1750	2000	2250	2500	2750	3000	3250

Frequency → Pitch(melody)

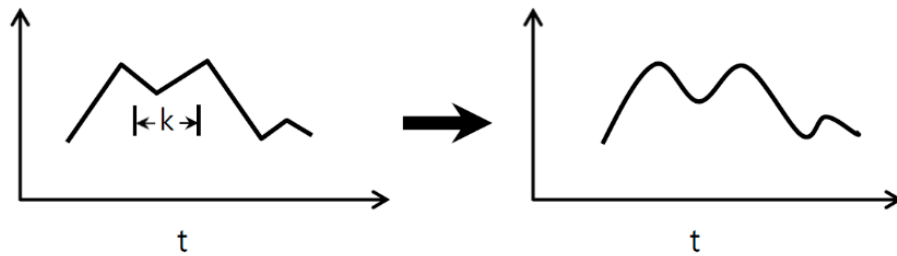
더 성능 높이기

2. 1d conv 설계 및 앙상블

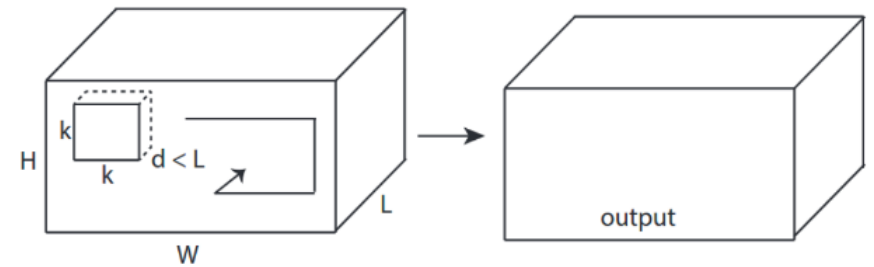
90.6%



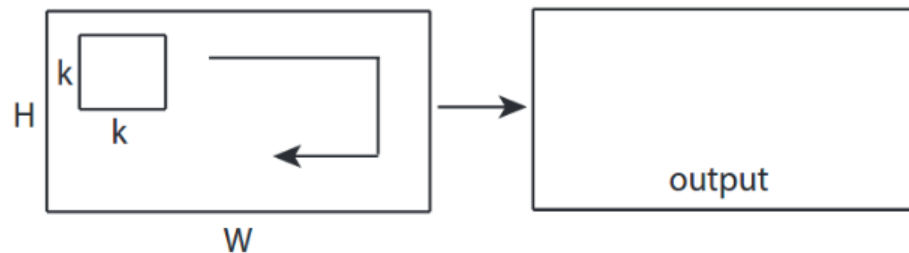
1D Conv



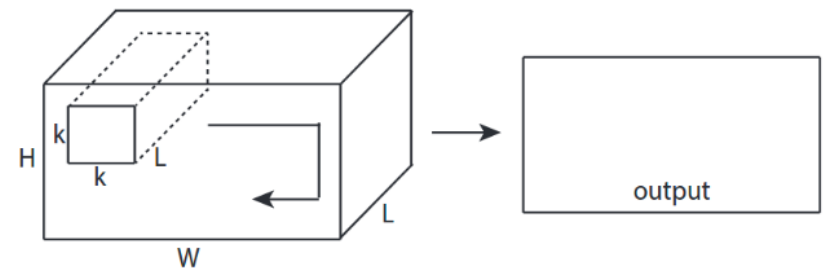
3D Conv
































2D Conv



2D with 3D Conv



제언

#	Δpub	Team Name	Kernel	Team Members	Score	Entries	Last
1	▲3	Heng-Ryan-See * good bug? *		  	0.91060	276	20d
2	▼1	Thomas O'Malley			0.91048	119	20d
3	—	Little Boat			0.91013	126	21d
4	▼2	high five		    	0.90931	195	20d
5	▲5	은주니(ttagu99) & sjv		 	0.90896	206	20d
6	▲1	S4		   	0.90825	170	20d
7	▼2	GREAT@SHU		   	0.90790	183	20d
8	▲3	VAZ		    	0.90649	270	20d
9	▼1	Gold Gazua		 	0.90637	174	20d
10	▼4	but		 	0.90532	123	20d

최대가 91%. kaggle의 다른 competition에 비해 score가 낮은 이유?

제언

train.7z - Contains a few informational files and a folder of audio files. The audio folder contains subfolders with 1 second clips of voice commands, with the folder name being the label of the audio clip. There are more labels that should be predicted. The labels you will need to predict in Test are **yes, no, up, down, left, right, on, off, stop, go**. Everything else should be considered either **unknown or silence**. The folder **_background_noise_** contains longer clips of "silence" that you can break up and use as training input.

train에 주어진 class는 총 30개
but test는 그 중 10개와 두 가지 class 추가

1. silence : 아무 소리도 안나는 경우
2. unknown : 주어진 클래스가 아닌 소리
 - 30개 중 나머지 20개
 - **아예 주어지지 않은 소리**

→ 네트워크에게 '알 수 없음' 을 가르쳐야 함

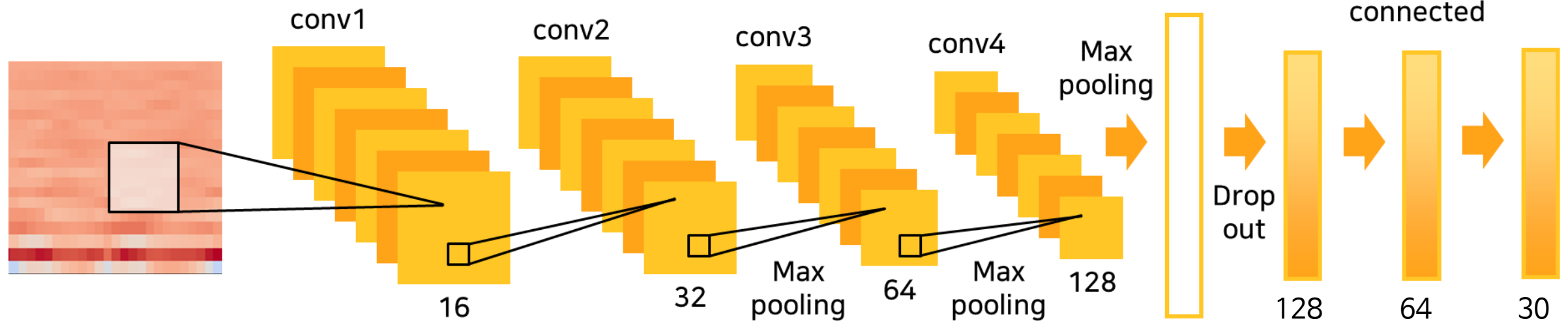
아마 이 문제 때문에 최대 91%가 아닐까

이삭 & 다경

sr = 44100

feature : MFCC

train, validation random하게
데이터 Augmentation없이



위와 비슷한 4가지 모델 앙상블

이삭 & 다경

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 284, 276, 8)	224
conv2d_2 (Conv2D)	(None, 282, 274, 16)	1168
max_pooling2d_1 (MaxPooling2)	(None, 94, 91, 16)	0
conv2d_3 (Conv2D)	(None, 92, 89, 32)	4640
max_pooling2d_2 (MaxPooling2)	(None, 30, 29, 32)	0
conv2d_4 (Conv2D)	(None, 28, 27, 64)	18496
max_pooling2d_3 (MaxPooling2)	(None, 9, 9, 64)	0
flatten_1 (Flatten)	(None, 5184)	0
dense_1 (Dense)	(None, 128)	663680
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 30)	1950
Total params: 698,414		
Trainable params: 698,414		

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 282, 274, 8)	608
conv2d_6 (Conv2D)	(None, 278, 270, 16)	3216
max_pooling2d_4 (MaxPooling2)	(None, 92, 90, 16)	0
conv2d_7 (Conv2D)	(None, 88, 86, 32)	12832
max_pooling2d_5 (MaxPooling2)	(None, 29, 28, 32)	0
conv2d_8 (Conv2D)	(None, 25, 24, 64)	51264
max_pooling2d_6 (MaxPooling2)	(None, 8, 8, 64)	0
flatten_2 (Flatten)	(None, 4096)	0
dense_4 (Dense)	(None, 128)	524416
dropout_2 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 64)	8256
dense_6 (Dense)	(None, 30)	1950
Total params: 602,542		
Trainable params: 602,542		
Non-trainable params: 0		

이삭 & 다경

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 282, 274, 8)	608
max_pooling2d_7 (MaxPooling2D)	(None, 94, 91, 8)	0
conv2d_10 (Conv2D)	(None, 90, 87, 16)	3216
max_pooling2d_8 (MaxPooling2D)	(None, 30, 29, 16)	0
conv2d_11 (Conv2D)	(None, 26, 25, 32)	12832
max_pooling2d_9 (MaxPooling2D)	(None, 8, 8, 32)	0
conv2d_12 (Conv2D)	(None, 4, 4, 64)	51264
flatten_3 (Flatten)	(None, 1024)	0
dense_7 (Dense)	(None, 128)	131200
dropout_3 (Dropout)	(None, 128)	0
dense_8 (Dense)	(None, 64)	8256
dense_9 (Dense)	(None, 30)	1950
Total params: 209,326		
Trainable params: 209,326		
Non-trainable params: 0		

Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 282, 274, 8)	608
conv2d_14 (Conv2D)	(None, 278, 270, 16)	3216
max_pooling2d_10 (MaxPooling2D)	(None, 92, 90, 16)	0
conv2d_15 (Conv2D)	(None, 88, 86, 32)	12832
max_pooling2d_11 (MaxPooling2D)	(None, 29, 28, 32)	0
conv2d_16 (Conv2D)	(None, 25, 24, 64)	51264
max_pooling2d_12 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_17 (Conv2D)	(None, 4, 4, 128)	204928
flatten_4 (Flatten)	(None, 2048)	0
dense_10 (Dense)	(None, 128)	262272
dropout_4 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 64)	8256
dense_12 (Dense)	(None, 30)	1950
Total params: 545,326		
Trainable params: 545,326		
Non-trainable params: 0		

이삭 & 다경

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
output2.csv	just now	0 seconds	2 seconds	0.77275

Complete

[Jump to your position on the leaderboard](#) ▼

이삭 & 다경

sr = 16000

feature : MFCC

train, validation 화자별로
데이터 Augmentation없이

Name	Submitted	Wait time	Execution time	Score
spectrogram_output2.csv	a few seconds ago	4 seconds	2 seconds	0.66282

Complete

[Jump to your position on the leaderboard ▼](#)

이삭 & 다경

sr = 16000

sr = 44100 → sr = 16000

0.77275 → 0.66282

원래 wav파일의 sr이 16000 이더라도 44100으로 추출하면

44100개 sampling

→ 더 정교한 feature라서?

이삭 & 다경

train, validation에 화자 겹치지 않도록,
랜덤하게 noise추가,
High resolution spectrogram,
sr = 16000
모델은 우리 모델로 앙상블

Name	Submitted	Wait time	Execution time	Score
spectrogram_ensemble_output.csv	just now	0 seconds	2 seconds	0.77686

Complete

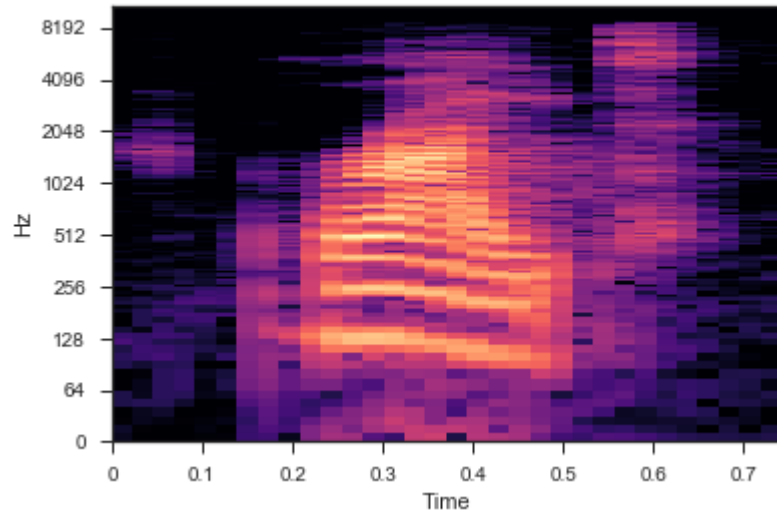
[Jump to your position on the leaderboard ▼](#)

sr = 16000으로 했을 때, 0.66282
→ 약 11% 증가

데이터 Augumentation,
sr = 44100으로도 해 볼 예정!

*번외 - 화자인식

투빅스 8기 Big Voice팀 5명의 목소리



일정한 시간동안 MFCC추출

→ n by m 행렬을 1 by $n*m$ 행렬로 핀 후

→ SVM으로 분류

→ Accuracy 99%

→ K-means 클러스터링은 Accuracy 약 80%정도

감사합니다