폐 RD?
(Respiratory disease)?

bAIkery

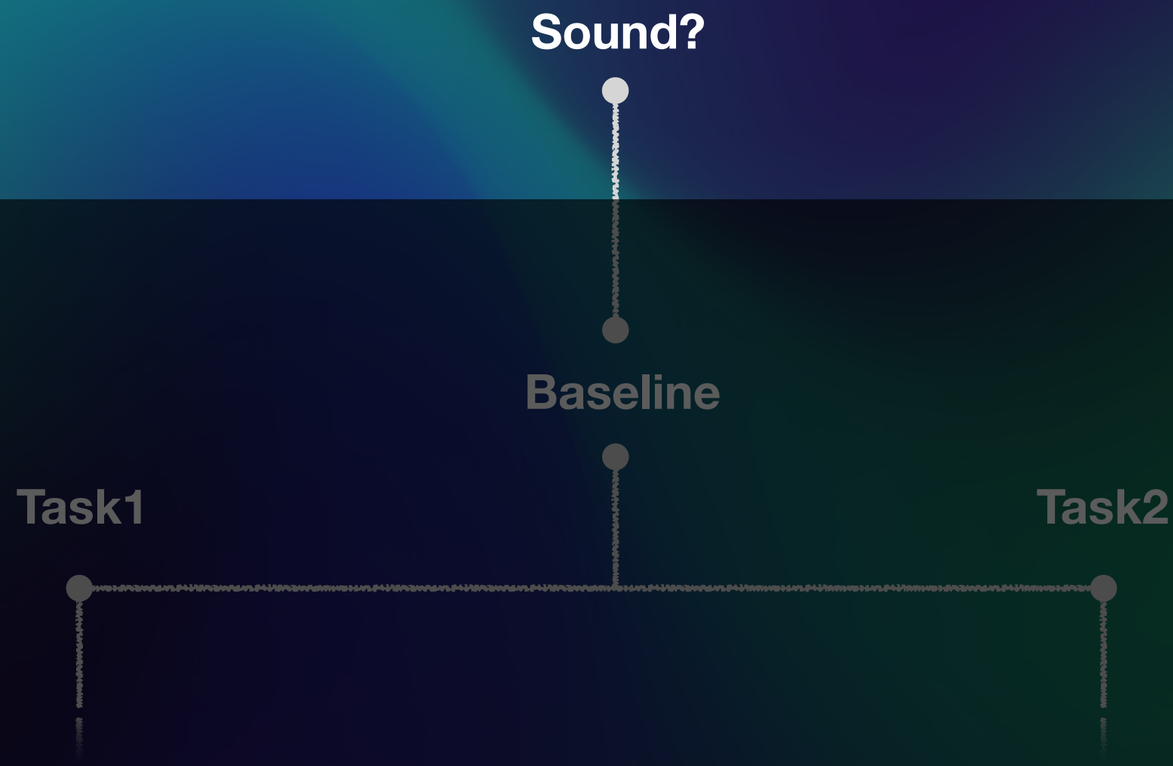
조 송 희
오 원 진
장 혜 선
최 정 인

AIFFEL X 스마트사운드

Contents

사운드 데이터셋 처리 과정, 과업

구현 기법



과업

주어진 호흡음 데이터셋을 사용하여 질병 분류하기 + 호흡 주기를 정상, 비정상으로 분류하기

-

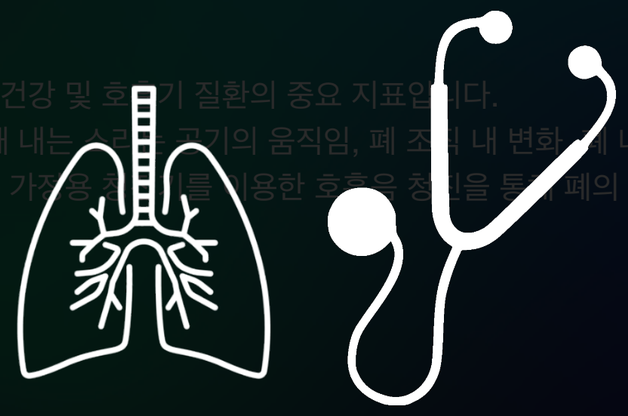
호흡음은 호흡기 건강 및 호흡기 질환의 중요 지표입니다.
사람이 숨을 쉴 때 내는 소리는 공기의 움직임, 폐 조직 내 변화, 폐 내 분비물의 위치와 직접적인 관련이 있습니다.
의료 전문용 혹은 가정용 청진기를 이용한 호흡음 청진을 통해 폐의 이상유무 및 1차 질병 예측을 해봄으로써 질병 확산을 미연에 방지할 수 있습니다.

과업

호흡음으로 질병 분류하기

-

호흡음은 호흡기 건강 및 호흡기 질환의 중요 지표입니다.
사람이 숨을 쉴 때 내는 소리를 공기의 움직임, 폐 조직 내 변화, 폐 내 분비물의 위치와 직장적 [illegible] 있습니다.
의료 전문용 혹은 가정용 청진기를 이용한 호흡음 청진을 통해 폐의 이상유무 및 1차 질병 예측을 [illegible] 질병확산을 [illegible] 있습니다.



<ICBHI dataset>

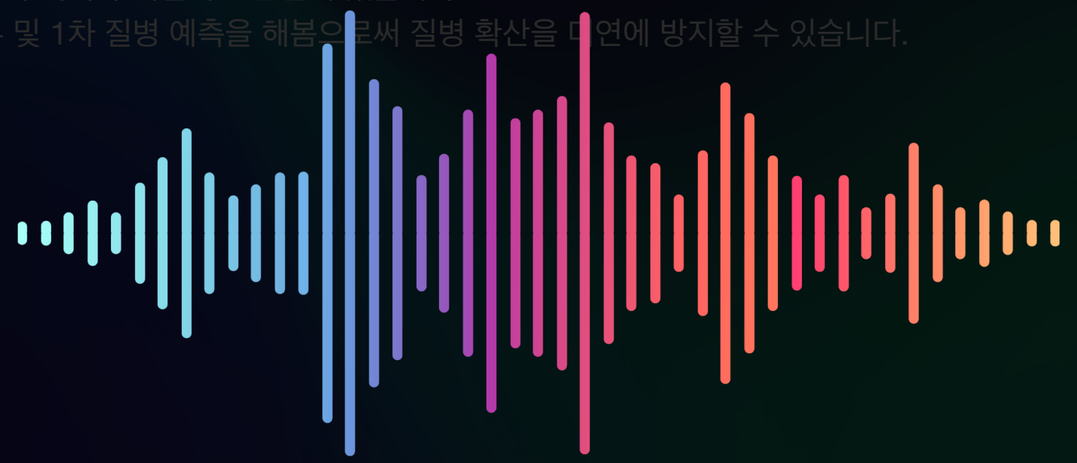
• 수포음(crackle) - High-pitched, 100 -2500Hz의 주파수 대역과 80msec 이상의 지속시간
• 천명음(wheeze) - High-pitched, crackle의 지속시간은 20 ms보다 더 낮고 주파수 대역은 100와 200 Hz 사이

과업

호흡음으로 질병 분류하기

-

호흡음은 호흡기 건강 및 호흡기 질환의 중요 지표입니다.
사람이 숨을 쉴 때 내는 소리는 공기의 움직임, 폐 조직 내 변화, 폐 내 분비물의 위치와 직접적인 관련이 있습니다.
의료 전문용 혹은 가정용 청진기를 이용한 호흡음 청진을 통해 폐의 이상유무 및 1차 질병 예측을 해봄으로써 질병 확산을 미연에 방지할 수 있습니다.



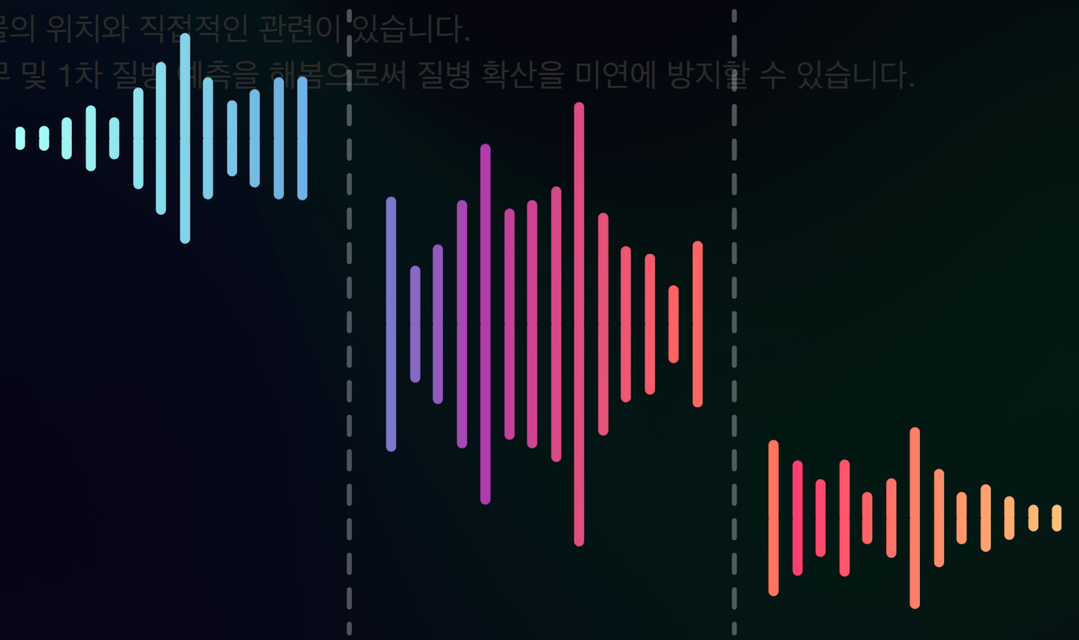
Time

과업

호흡음으로 질병 분류하기

-

호흡음은 호흡기 건강 및 호흡기 질환의 중요 지표입니다.
사람이 숨을 쉴 때 내는 소리는 공기의 움직임, 폐 조직 내 변화, 폐 내 분비물의 위치와 직접적인 관련이 있습니다.
의료 전문용 혹은 가정용 청진기를 이용한 호흡음 청진을 통해 폐의 이상유무 및 1차 질병 예측을 해봄으로써 질병 확산을 미연에 방지할 수 있습니다.



과업

호흡음으로 질병 분류하기

-

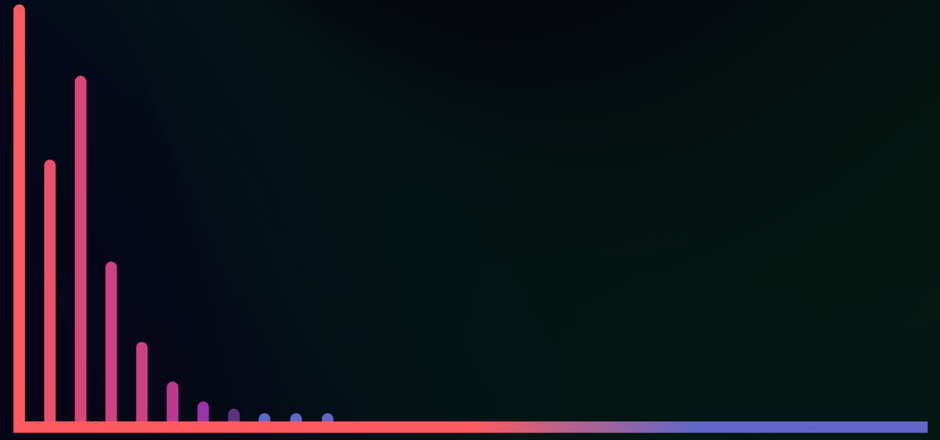
호흡음은 호흡기 건강 및 호흡기 질환의 중요 지표입니다.
사람이 숨을 쉴 때 내는 소리는 공기의 움직임, 폐 조직 내 변화, 폐 내 분비물의 위치와 직접적인 관련이 있습니다.
의료 전문용 혹은 가정용 청진기를 이용한 호흡음 청진을 통해 폐의 이상유무 및 1차 질병 예측을 해봄으로써 질병 확산을 미연에 방지할 수 있습니다.

과업

호흡음으로 질병 분류하기

-

호흡음은 호흡기 건강 및 호흡기 질환의 중요 지표입니다.
사람이 숨을 쉴 때 내는 소리는 공기의 움직임, 폐 조직 내 변화, 폐 내 분비물의 위치와 직접적인 관련이 있습니다.
의료 전문용 혹은 가정용 청진기를 이용한 호흡음 청진을 통해 폐의 이상유무 및 1차 질병 예측을 해봄으로써 질병 확산을 미연에 방지할 수 있습니다.

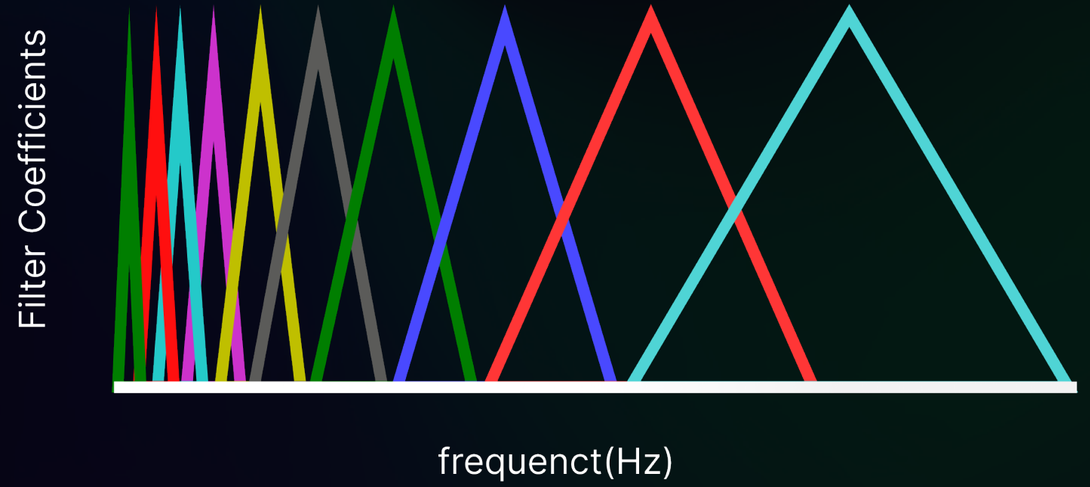


과업

호흡음으로 질병 분류하기

-

호흡음은 호흡기 건강 및 호흡기 질환의 중요 지표입니다.
사람이 숨을 쉴 때 내는 소리는 공기의 움직임, 폐 조직 내 변화, 폐 내 분비물의 위치와 직접적인 관련이 있습니다.
의료 전문용 혹은 가정용 청진기를 이용한 호흡음 청진을 통해 폐의 이상유무 및 1차 질병 예측을 해봄으로써 질병 확산을 미연에 방지할 수 있습니다.



특성 추출

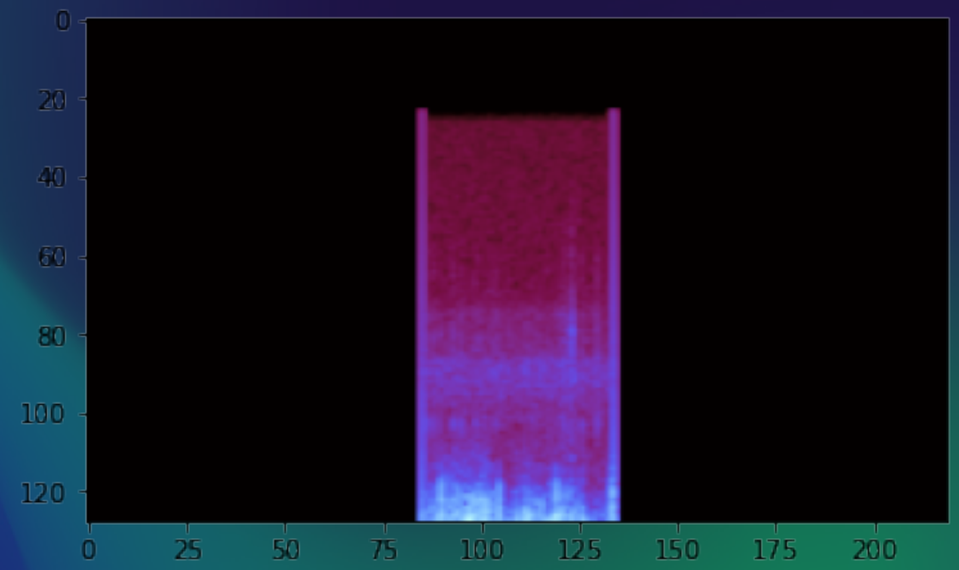
mel feature extraction

-

소리의 고유한 특징을 추출

라이브러리

librosa




```python
#mel feature extraction
def create_mel_raw(current_window, sr, n_mels=128, f_min=100, f_max=16000, nfft=2048, hop=512, resz=1):
    S = lb.feature.melspectrogram(y=current_window, sr=sr, n_mels=n_mels, fmin=f_min, fmax=f_max, n_fft=nfft, hop_length=hop)
    S = lb.power_to_db(S, ref=np.max)
    S = (S-S.min()) / (S.max() - S.min())
    S *= 255
    img = cv2.applyColorMap(S.astype(np.uint8), cmapy.cmap('magma'))
    height, width, _ = img.shape
    if resz > 0:
        img = cv2.resize(img, (width*resz, height*resz), interpolation=cv2.INTER_LINEAR) #bicubic interpolation
    img = cv2.flip(img, 0)
    return img
```

과업

호흡음으로 질병 분류하기

-

호흡음은 호흡기 건강 및 호흡기 질환의 중요 지표입니다.
사람이 숨을 쉴 때 내는 소리는 공기의 움직임, 폐 조직 내 변화, 폐 내 분비물의 위치와 직접적인 관련이 있습니다.
의료 전문용 혹은 가정용 청진기를 이용한 호흡음 청진을 통해 폐의 이상유무 및 1차 질병 예측을 해봄으로써 질병 확산을 미연에 방지할 수 있습니다.

log

Data Load

데이터 불러오기 - df

-

1. wav파일에 대한 정보가 담긴 라벨링 데이터 pid로 concat
2. 데이터 프레임으로 불러오기

Data Load

데이터 불러오기 - df

-

1. wav파일에 대한 정보가 담긴 라벨링 데이터 pid로 concat
2. 데이터 프레임으로 불러오기

```
0.036   0.579   0       0
0.579   2.45    0       0
2.45    3.893   0       0
3.893   5.793   0       0
5.793   7.521   0       0
7.521   9.279   0       0
9.279   11.15   0       0
11.15   13.036  0       0
13.036  14.721  0       0
14.721  16.707  0       0
16.707  18.507  0       0
18.507  19.964  0       0
```

101_1b1_Al_sc_Meditron.**txt**

```
[ ]  raw_df = pd.read_csv(root + 'csv_data/data.csv')
     raw_df.head()
```

	start	end	crackles	weezels	pid	mode	filename	disease
0	1.862	5.718	0	1	160	mc	160_1b3_Al_mc_AKGC417L	COPD
1	5.718	9.725	1	1	160	mc	160_1b3_Al_mc_AKGC417L	COPD
2	9.725	13.614	0	1	160	mc	160_1b3_Al_mc_AKGC417L	COPD
3	13.614	17.671	0	1	160	mc	160_1b3_Al_mc_AKGC417L	COPD
4	17.671	19.541	0	0	160	mc	160_1b3_Al_mc_AKGC417L	COPD

Data Load

데이터 불러오기 - .wav

-

1. sampling rate 지정
   respireNet에서는 4kHz로 down sampling
2. librosa.load 사용
   load 시 sampling rate 파라미터로 전체 data에 일괄 적용 가능

Data Load

데이터 불러오기 - .wav

-

1. sampling rate 지정
   respireNet에서는 4kHz로 down sampling
2. librosa.load 사용
   load 시 sampling rate 파라미터로 전체 data에 일괄 적용 가능

라이브러리

librosa

```
[ ]  root = '/content/drive/MyDrive/aiffel/aiffelthon/'
     AUTOTUNE = tf.data.experimental.AUTOTUNE
     sr = 16000 # sampling rate
```

```
[ ]  wav_show = []
     wav_root = root + 'archive/Respiratory_Sound_Database/Respiratory_Sound_Database/audio_and_txt_files/'
     wav_filename = ['157_1b1_Pr_sc_Meditron.wav',
                     '154_4b4_Pl_mc_AKGC417L.wav',
                     '151_3p2_Tc_mc_AKGC417L.wav']

     for idx in range(3):
       audio_2, _= lb.load(wav_root + wav_filename[idx], sr = sr)
       wav_show.append(audio_2)
```

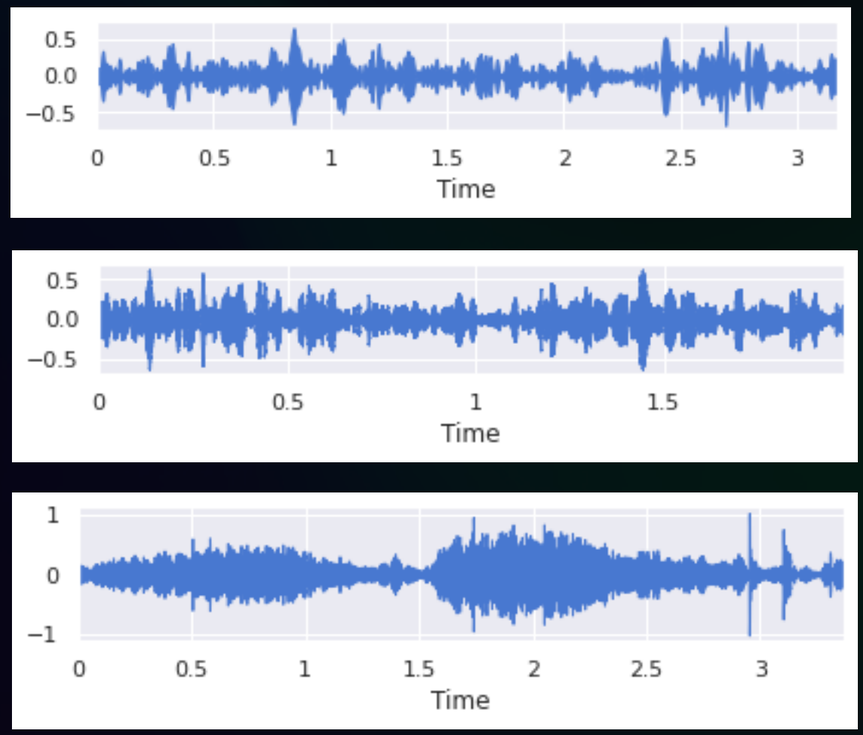
Data Load

데이터 불러오기 - .wav

-

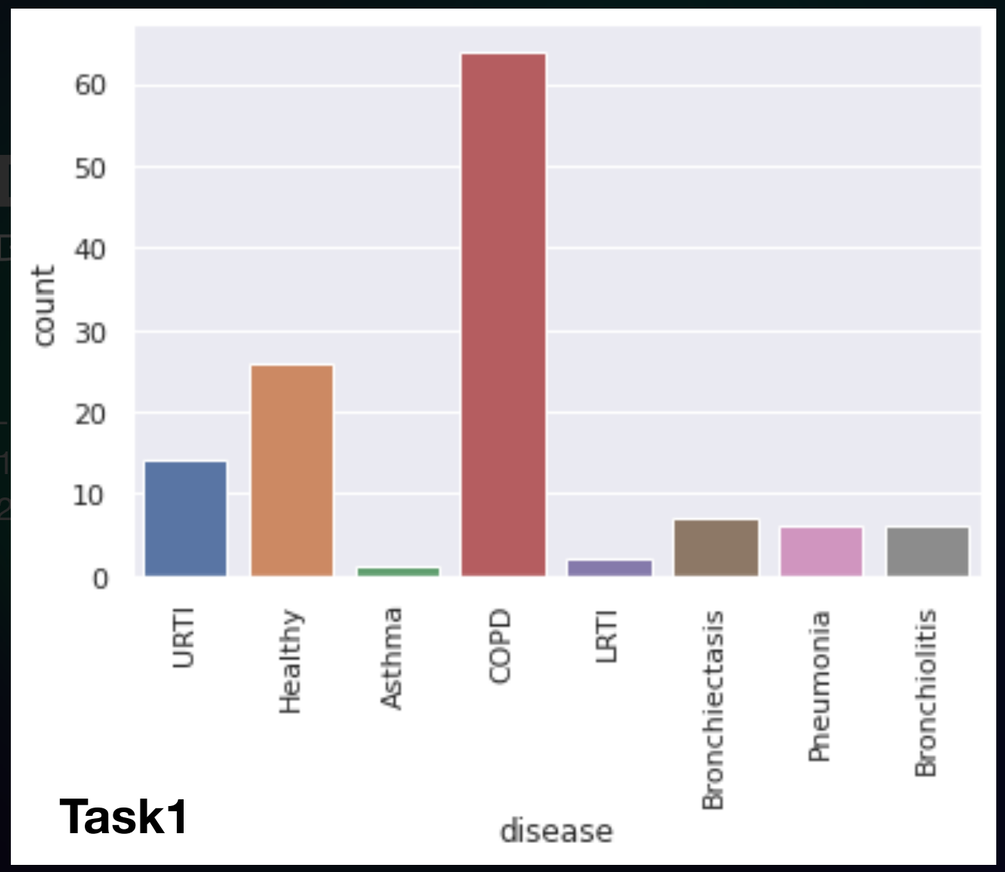
1. sampling rate 지정
   respireNet에서는 4kHz로 down sampling
2. librosa.load 사용
   load 시 sampling rate 파라미터로 전체 data에 일괄 적용 가능

```
import librosa.display as lbd

sns.set_theme(style="darkgrid")
sns.set_palette("muted")
plt.figure()

for idx in range(3):
  plt.subplot(3,1,idx+1)
  lbd.waveplot(np.array(wav_show, dtype=object)[idx], sr=sr)
  plt.tight_layout()
  plt.show()
```

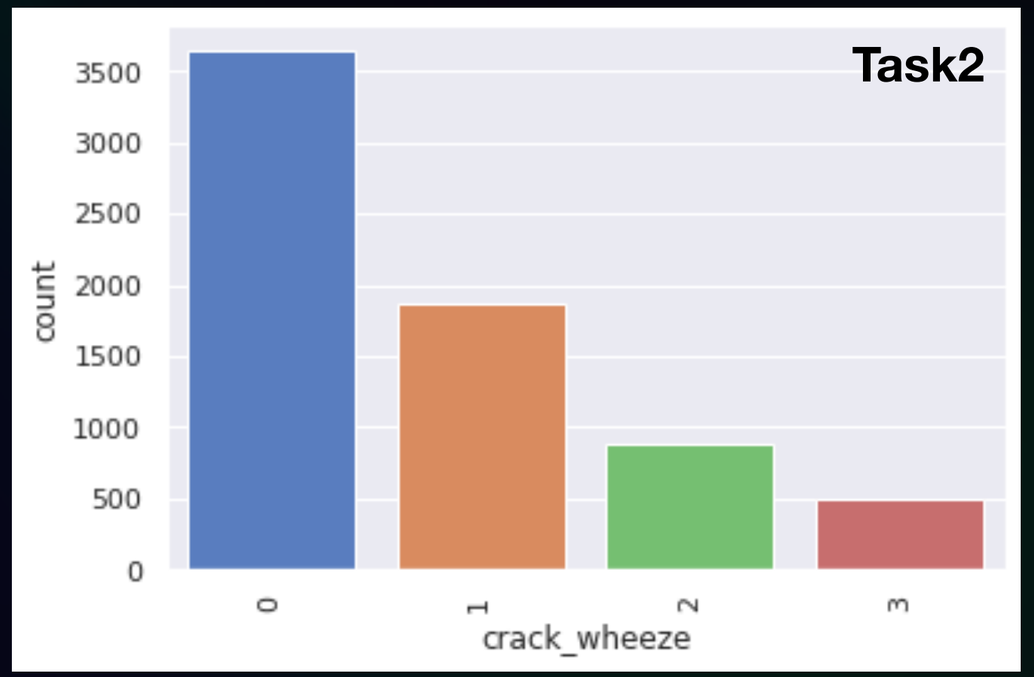


Data Load
데이터 분석

-
1. 데이터 imbalance 확인
2. 적절한 패딩 시간(초) 분석




```
COPD              0.832910
Healthy           0.046756
Pneumonia         0.041319
URTI              0.035158
Bronchiolitis     0.023197
Bronchiectasis    0.015042
LRTI              0.004712
Asthma            0.000906
Name: disease, dtype: float64
```



- 0 : Normal
- 1 : Crackle
- 2 : Wheeze
- 3 : Crackle & Wheeze

```
0    0.527909
1    0.270207
2    0.128489
3    0.073396
Name: crack_wheeze, dtype: float64
```

Class Imbalance

의료 데이터셋의 특성

-

1. Concatenation-based Augmentation
2. Train -> .wav
3. Train -> img

Data Load

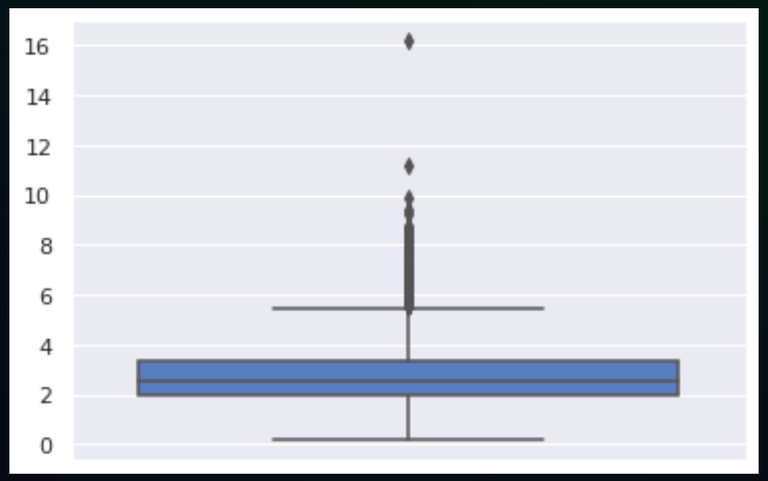
데이터 분석

-

1. 데이터 imbalance 확인
2. 적절한 패딩 시간(초) 분석

```
sns.scatterplot(x=(raw_df2.end-raw_df2.start), y=raw_df2.pid)
```


```
sns.boxplot(y=(raw_df2.end-raw_df2.start))
```



Pre-processing
data slicing

-
한 호흡 주기에 맞춰서 data 자르기

Pre-processing

data slicing

-

한 호흡 주기에 맞춰서 data 자르기

```
def getPureSample(raw_data,start,end,sr=22050):

    '''
    Takes a numpy array and spilts its using start and end args

    raw_data=numpy array of audio sample
    start=time
    end=time
    sr=sampling_rate
    mode=mono/stereo
    '''

    max_ind = len(raw_data) #원본 데이터
    start_ind = min(int(start * sr), max_ind) #시작 시간 x sample rate
    end_ind = min(int(end * sr), max_ind)
    return raw_data[start_ind: end_ind]
```

```
i,c=0,0

filename2 = []
start2 = []
end2 = []
pid2 = []
mode2 = []
disease2 = []
crack_wheeze = []
for index,row in raw_df.iterrows():
    start=row['start']
    end=row['end']
    filename=row['filename']

    # 불러올 파일 경로
    audio_file_loc=root + filename + '.wav'

    # 새로운 파일 이름 지정
    if index > 0:
        if raw_df.iloc[index-1]['filename']==filename:
            i+=1
        else:
            i=0
    filename= filename + '_' + str(i) + '.wav'

    filename2.append(filename)
    start2.append(row['start'])
    end2.append(row['end'])
    pid2.append(row['pid'])
    mode2.append(row['mode'])
```

Pre-processing

data slicing

-

한 호흡 주기에 맞춰서 data 자르기

```
    disease2.append(row['disease'])

    if row['crackles'] == 0 and row['weezels'] == 0:
      crack_wheeze.append(0)
    elif row['crackles'] == 1 and row['weezels'] == 0:
      crack_wheeze.append(1)
    elif row['crackles'] == 0 and row['weezels'] == 1:
      crack_wheeze.append(2)
    else:
      crack_wheeze.append(3)

    save_path='/content/drive/MyDrive/aiffel/aiffelthon/processed_audio_files_8/' + filename
    c+=1 #파일 개수 세기

    audioArr,sampleRate=lb.load(audio_file_loc, sr = 16000)
    pureSample=getPureSample(audioArr,start,end,sampleRate)

    sf.write(file=save_path,data=pureSample,samplerate=sampleRate)
print('Total Files Processed: ',c)
```

Baseline

과업
- Goal

Data Load
- Data Analysis
- data slicing

Pre-processing
- zero padding
- 5th butter worth filter

Train/Test Split

Augmentation
- shift, speed
- concatenation-based Augmentation

Feature Extraction
- mel feature extraction

Build Model
- MFCC model + CSTFT model + MSPEC model

Report

Baseline

과업

- Goal

Data Load

- Data Analysis
- data slicing

시도한 기법들

Pre-processing

- zero padding
- 5th butter worth filter

- concatenation-based Augmentation

Train/Test Split

Augmentation

- shift, speed

Feature Extraction

- mel feature extraction

- MFCC model + CSTFT model + MSPEC model

Build Model

Report

Pre-processing
Butterworth filter

-
low, high frequency를 완만하게 잘라줌
위를 잘라주니까 blank region clipping과 같은 효과

SciPy

```
def butter_bandpass(lowcut, highcut, fs, order=5):
        nyq = 0.5 * fs
        low = lowcut / nyq
        high = highcut / nyq
        sos = butter(order, [low, high], analog=False, btype='band', output='sos')
        return sos

def butter_bandpass_filter(data, lowcut, highcut, fs, order=5):
        sos = butter_bandpass(lowcut, highcut, fs, order=order)
        y = sosfilt(sos, data)
        return y
```

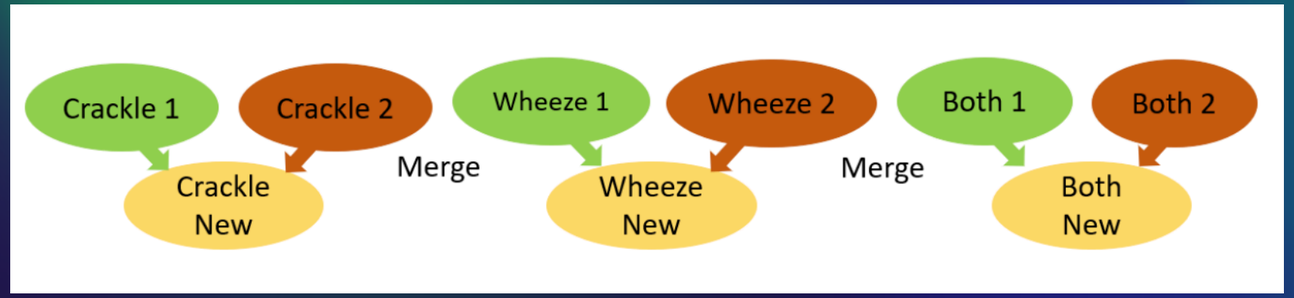
Data Augmentation

Concatenation-based Augmentation

-

Data lack problem 및 imbalance 해결을 위해 같은 class data를 concate 해서 새로운 data 생성

=> abnormal 클래스의 분류 정확도 꽤(non-trivially) 향상(RespireNet(2020))



Train Test Split

stratify

-

normal/crackle/wheeze/both 클래스 비율대로

```
from sklearn.model_selection import train_test_split

Xtrain,Xval,ytrain,yval=train_test_split( #disease 분류
    processed,processed.disease,stratify=processed.disease,random_state=42,test_size=0.2)

Xtrain_1,Xval_1,ytrain_1,yval_1=train_test_split( #crack_wheeze 분류
    processed,processed.crack_wheeze,stratify=processed.crack_wheeze,random_state=42,test_size=0.2)
```

Augmentation

shift, speed

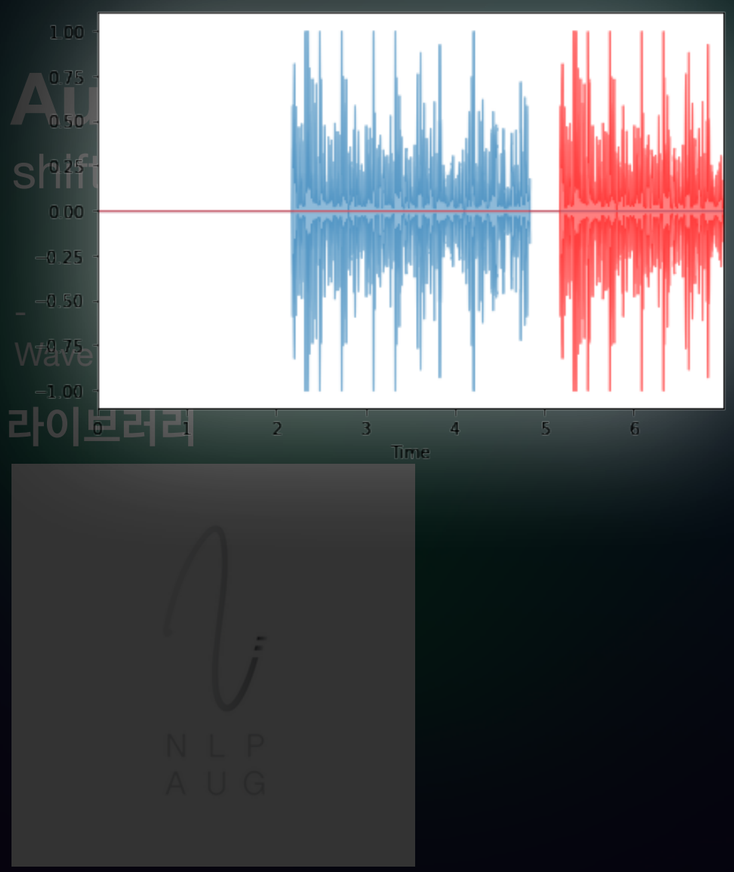
-

Wave augmentation

라이브러리

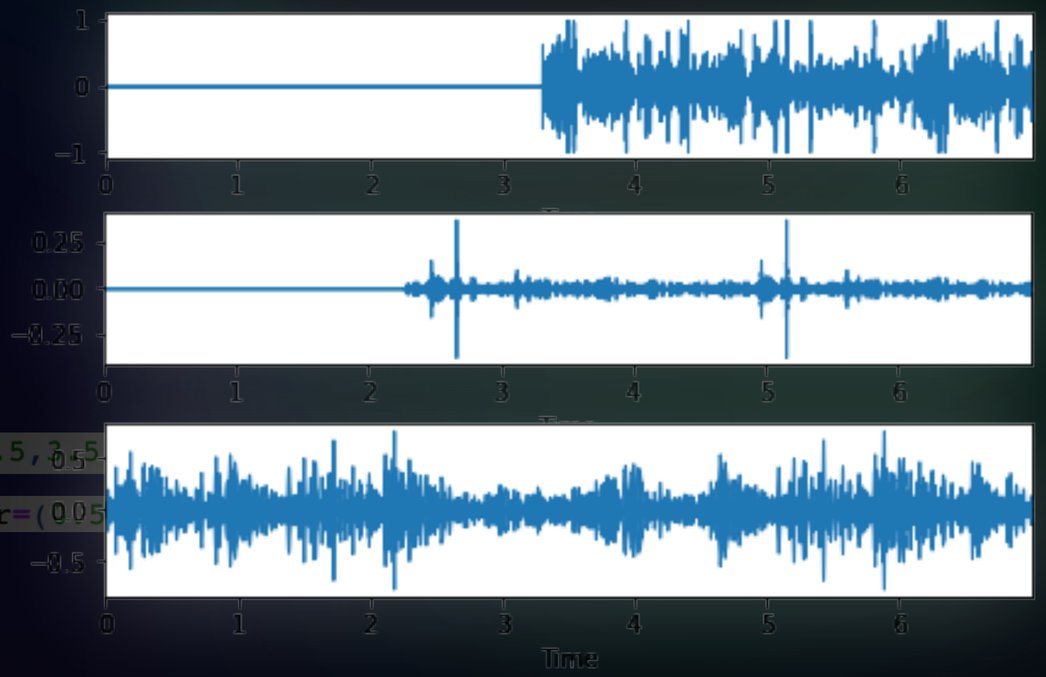

```
aug = naa.ShiftAug(sampling_rate=sr, duration=uniform(1.5,3.5)) #0과 1 사이의 실수
```

```
aug = naa.SpeedAug(zone=(0.2, 0.8), coverage=0.5, factor=(0.5, 1), stateless=True) #0과 1 사이의 실수
```



- 증강 적용

```python
def pre_aug(signal, label, sr=sr):
    prob = np.random.randint(0,2)
    if prob == 0:
        return shift_aug(signal), label
    else:
        aug = speed_aug(signal)
        return audio_slice(aug), label
```




```python
aug = naa.ShiftAug(sampling_rate=sr, duration=uniform(1.5,3.5)
```

```python
aug = naa.SpeedAug(zone=(0.2, 0.8), coverage=0.5, factor=(0.5
```

Task1

- Disease classification

과업

- Goal

Data Load

- Data Analysis
- data slicing

Pre-processing

- zero padding
- 5th butter worth filter

- concatenation-based Augmentation

Train/Test Split

Augmentation

- shift, speed

Feature Extraction

- mel feature extraction

- NET_1

Build Model

Report

Task1

- Disease classification

과업

- Goal

Data Load

- Data Analysis
- data slicing

Pre-processing

- zero padding
- 5th butter worth filter

- concatenation-based Augmentation

Train/Test Split

Augmentation

- shift, speed

Feature Extraction

- mel feature extraction

- NET_1

Build Model

Report

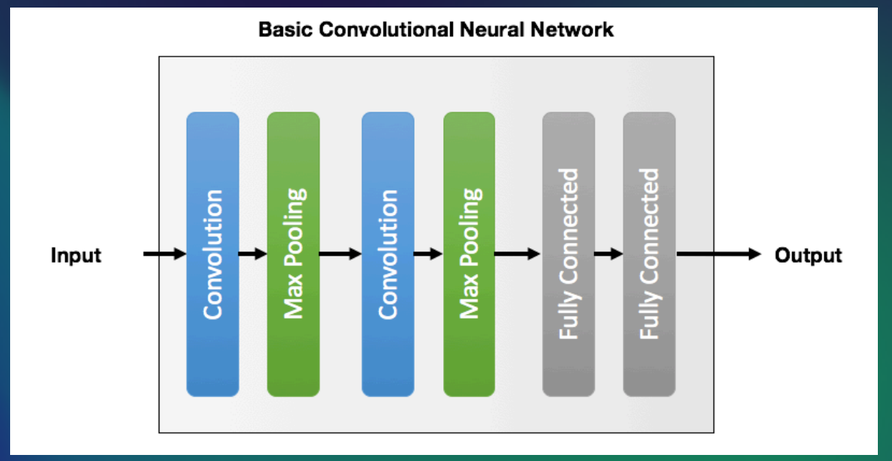
Task1
질병 분류하기

-
질병 클래스 - COPD(만성폐쇄성질환)을 포함한 8개의 질병

Build Model
질병 분류기

-

NET_1 = MFCC model + CSTFT model + mSpec Model + ADD model




```
skip_x = x # (32, 43, 32) #skip connection 추가
skip_x = keras.layers.Conv2D(96,3,strides=(4,4),padding='same')(skip_x) #(8, 11, 96)

x=keras.layers.Conv2D(64,3,strides=(2,2),padding='same')(x) #(16, 22, 64)
x=keras.layers.BatchNormalization()(x)
x=keras.layers.Activation(keras.activations.relu)(x)
x=keras.layers.MaxPooling2D(pool_size=2,padding='valid')(x) #(8, 11, 64)

x=keras.layers.Conv2D(96,2,padding='same')(x) #(8, 11, 96)
x = keras.layers.Add()([x,skip_x])
```

Task2
(additional)

- Crackle and wheeze detection

- zero padding/duplicated padding/smart padding

- concatenation-based Augmentation

과업
- Goal

Data Load
- Data Analysis
- data slicing

Pre-processing
- zero padding
- 5th butter worth filter

Train/Test Split

Augmentation
- shift, speed

Feature Extraction
- mel feature extraction

Post-processing
- blank region clipping
- Time Masking
- Frequency Masking

Build Model
- simple CNN
- ResNet-34

Report

Task2
(additional)

- Crackle and wheeze detection

- zero padding/duplicated padding/smart padding

- concatenation-based Augmentation

Post-processing
- blank region clipping
- Time Masking
- Frequency Masking

- simple CNN
- ResNet-34

과업
- Goal

Data Load
- Data Analysis
- data slicing

Pre-processing
- zero padding
- 5th butter worth filter

Train/Test Split

Augmentation
- shift, speed

Feature Extraction
- mel feature extraction

Build Model

Report

Task2

crackle/wheeze detection(additional)

-

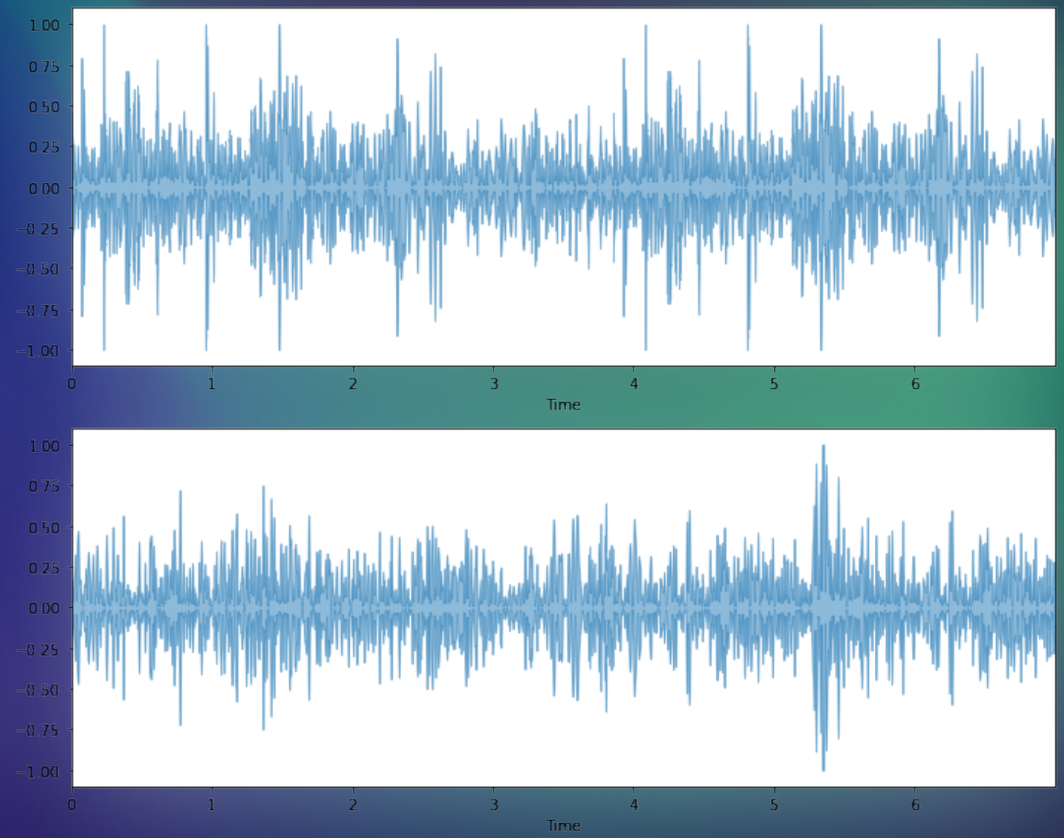
호흡음 클래스 - normal / crackle / wheeze / both(crackle and wheeze)

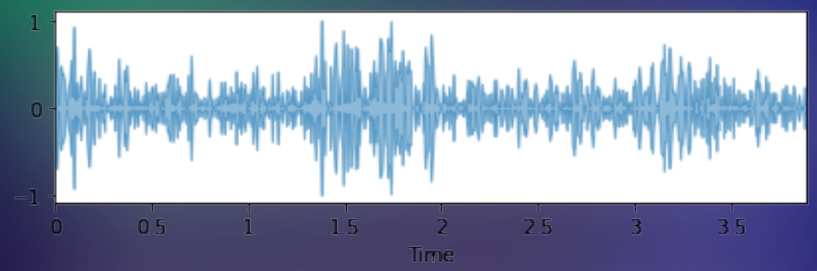
Pre-processing

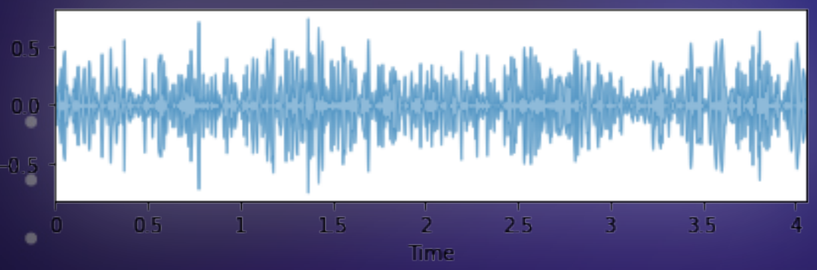
zero padding/duplicated padding/smart padding

-

다양한 패딩을 구현, 적절하게 사용하기







Pre-processing

zero padding/duplicated padding/smart padding

-

다양한 패딩을 구현, 적절하게 사용하기

```python
def duplicated_padding(idx):
    audio_data, _ = lb.load(root + df.loc[idx, 'filename'], sr=sr)
    return (duplicated_check_length(np.concatenate((audio_data, audio_data))))
```

```python
def smart_padding(idx_i, idx_j):
    audio_i, _ = lb.load(root + df.loc[idx_i, 'filename'], sr=sr)
    audio_j, _ = lb.load(root + df.loc[idx_j, 'filename'], sr=sr)
    new_audio = np.concatenate((audio_i, audio_j))
    return smart_check_length(new_audio, idx_j, audio_i, audio_j)
```

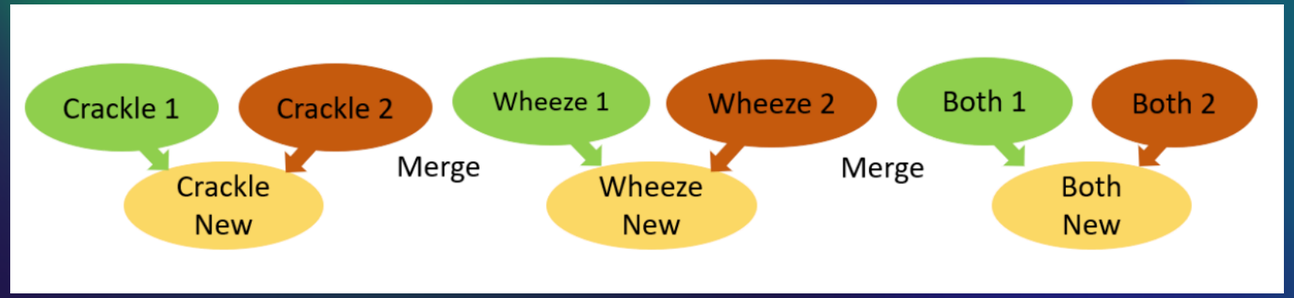
Data Augmentation

Concatenated Based Augmentation

-

Data lack problem 및 imbalance 해결을 위해 같은 class data를 concate 해서 새로운 data 생성

=> abnormal 클래스의 분류 정확도 꽤(non-trivially) 향상(RespireNet(2020))



Data Augmentation

Concatenated Based Augmentation

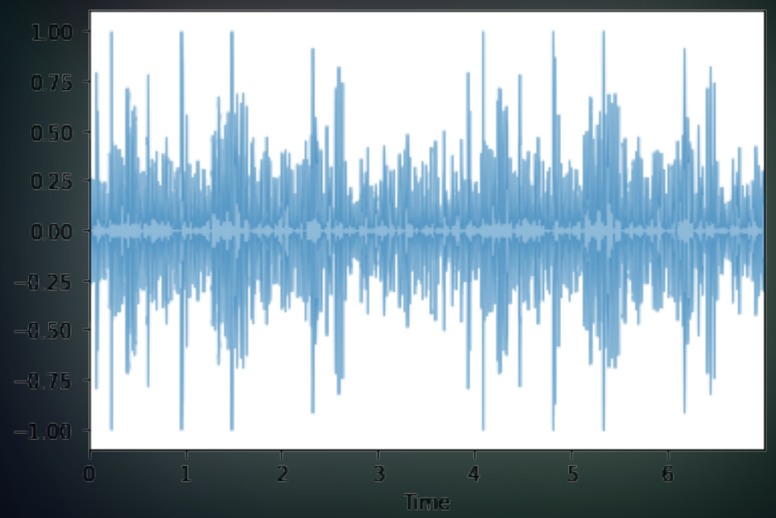
라이브러리

librosa

Duplicated padding

```python
#duplicated padding
for index, row in df.iterrows():
    filename = row['filename']
    audio_file = root + filename
    audio_data, _ = lb.load(audio_file, sr=16000)
    duplicated = np.concatenate((audio_data, audio_data))
    check_data = len(duplicated) / 16000

    if check_data > maxLen:
        test.append(duplicated)

    else:
        check_length2(duplicated, maxLen=maxLen)
```

시각화



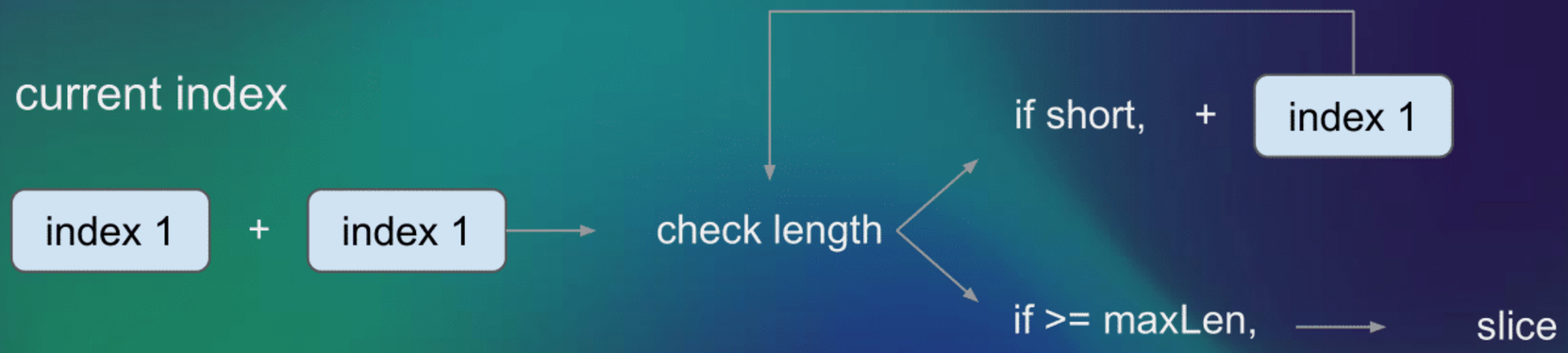
Smart padding

```
def smart_check_length(new_audio, idx_j, audio_i, audio_j):
    if len(new_audio)/sr >= maxLen:
        return audio_slicing(new_audio)
    else:
        if df['pid'][idx_j] == df['pid'][idx_j+1] and (df.loc[idx_j, 'crack_wheeze'] == df.loc[idx_j+1, 'crack_wheeze'] or df.loc[idx_j+1
            audio_k, _ = lb.load(root + df.loc[idx_j+1, 'filename'], sr=sr)
            new_audio_1 = np.concatenate((new_audio, audio_k))
            idx_j = idx_j + 1
            return smart_check_length(new_audio_1, idx_j, audio_i, audio_j)
        else:
            prob = np.random.randint(0,2)
            if prob == 0:
                audio_1 = np.concatenate((new_audio, audio_i))
                return smart_check_length(audio_1, idx_j, audio_i, audio_j)
            else:
                audio_2 = np.concatenate((new_audio, audio_j))
                return smart_check_length(audio_2, idx_j, audio_i, audio_j)
```

```
def smart_padding(idx_i, idx_j):
    audio_i, _ = lb.load(root + df.loc[idx_i, 'filename'], sr=sr)
    audio_j, _ = lb.load(root + df.loc[idx_j, 'filename'], sr=sr)
    new_audio = np.concatenate((audio_i, audio_j))
    return smart_check_length(new_audio, idx_j, audio_i, audio_j)
```

```
smart_padding_test=[]
for idx in range(len(df[:10])):
    if df['pid'][idx] == df['pid'][idx+1] and (df.loc[idx, 'crack_wheeze'] == df.loc[idx+1, 'crack_wheeze']
                                               or df.loc[idx+1, 'crack_wheeze'] ==0):
        smart_padding_test.append(smart_padding(idx, idx+1))
    else:
        smart_padding_test.append(duplicated_padding(idx))
```

duplicated padding



smart padding

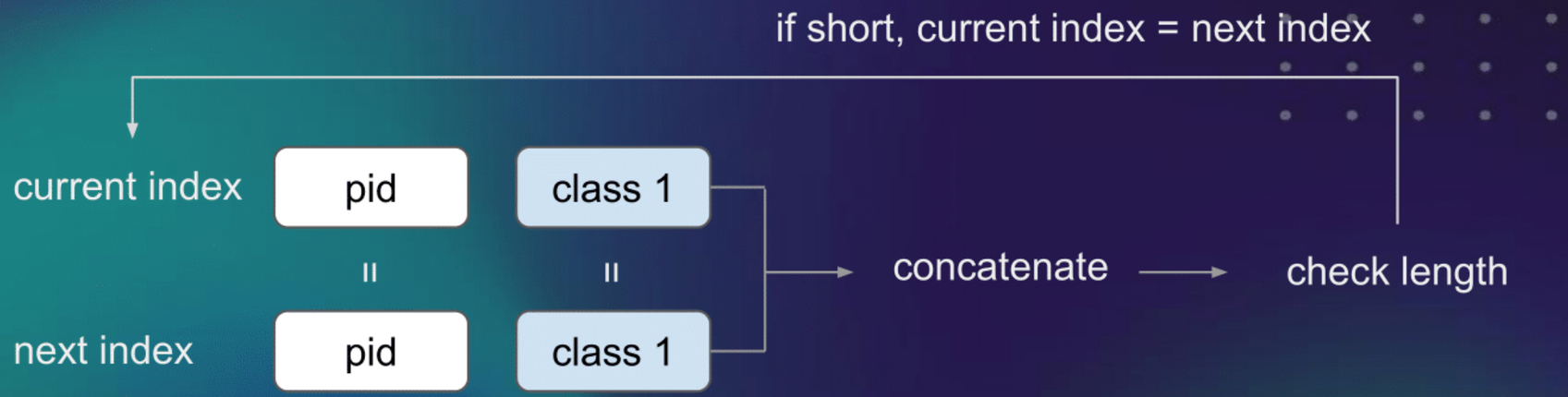
class normal	class crackle	class wheeze	class both
class 0 + class 0	class 1 + class 1	class 2 + class 2	class 3 + class 3
	class 1 + class 0	class 2 + class 0	class 3 + class 0

smart padding

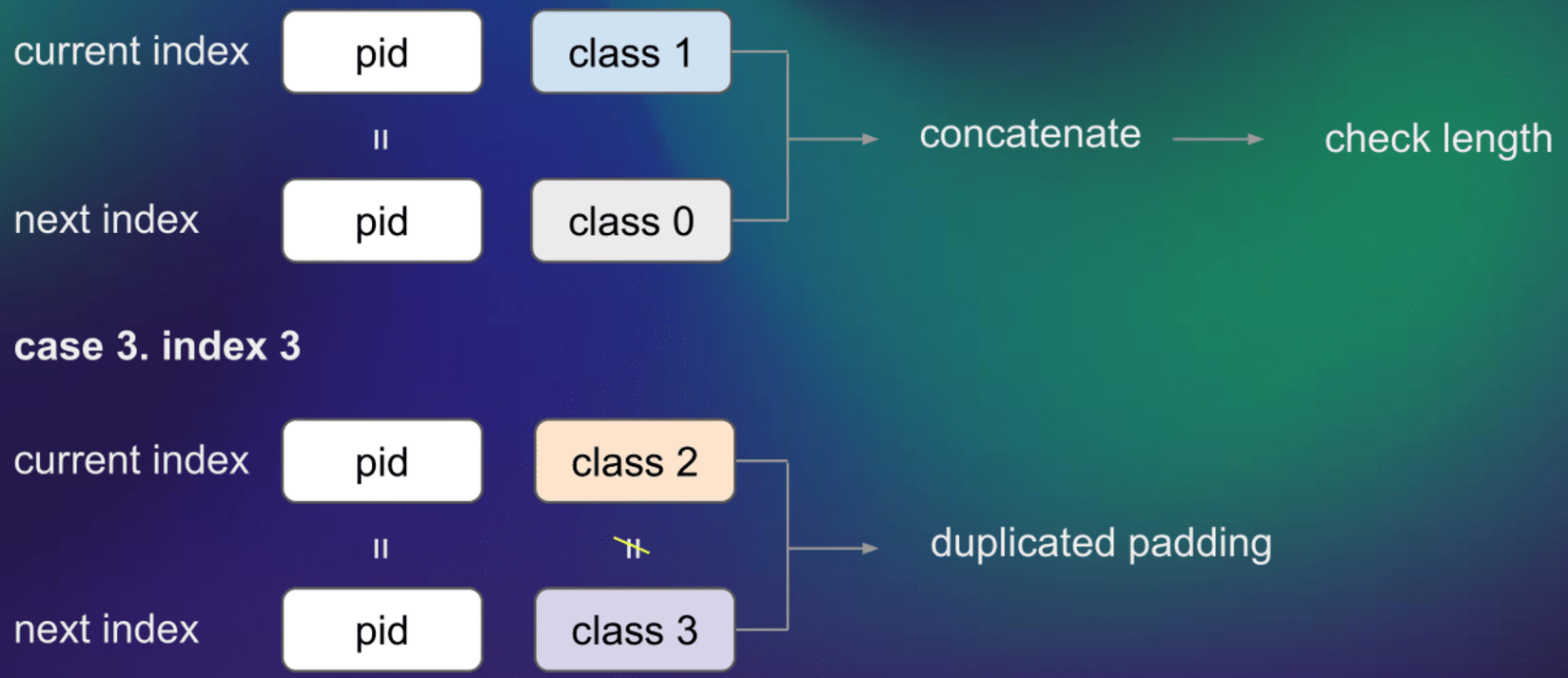
class 0 | class 1 | class 2 | class 3

index	pid	class
0	160	1
1	160	1
2	160	0
3	161	2
4	161	3
5	161	3

case 1. index 0



case 2. index 1

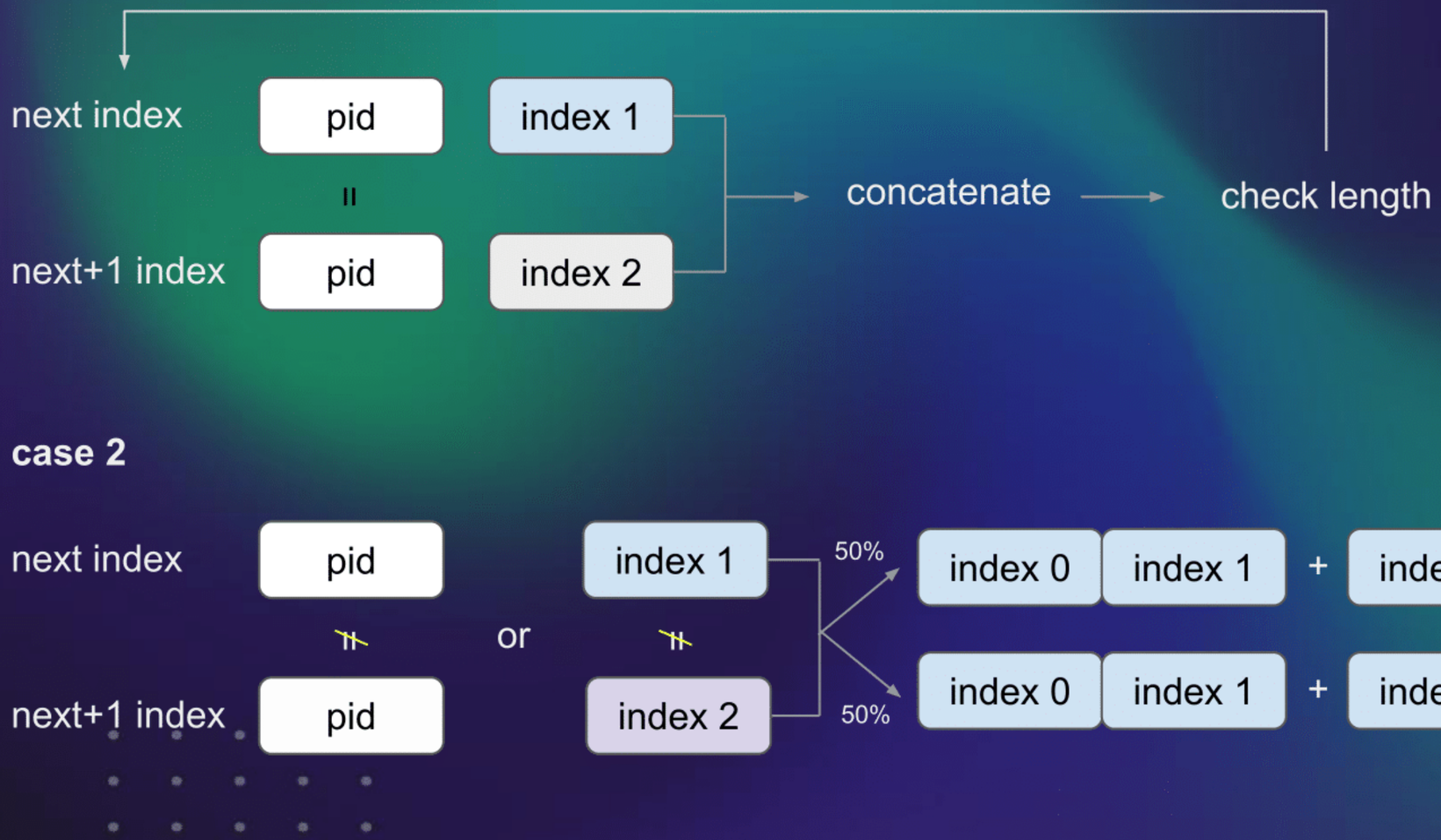


case 3. index 3

smart padding

in-depth case - index 0

case 1



case 2

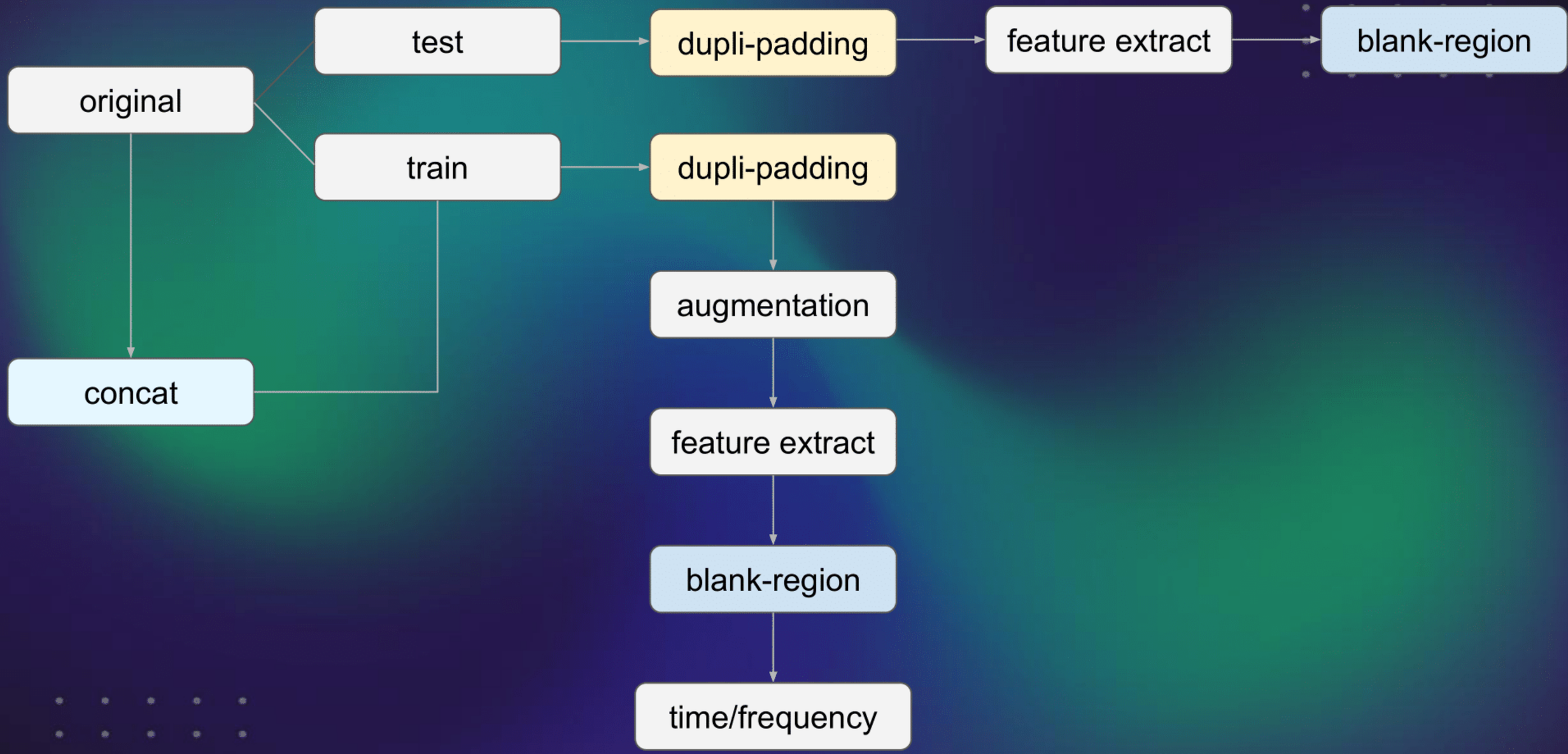
audio example

zero padding

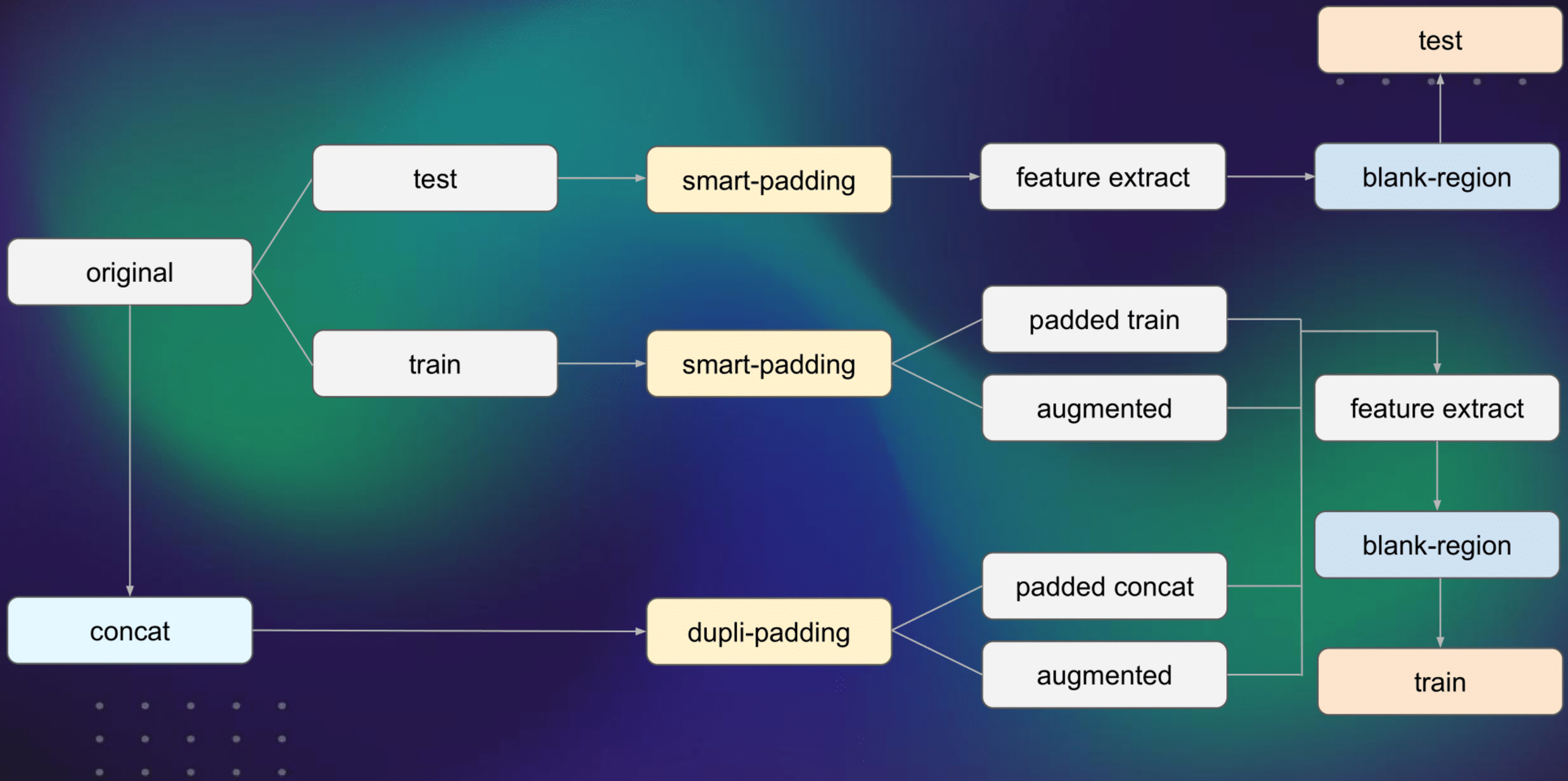
duplicated padding

smart padding

trial 1


original
test
dupli-padding
feature extract
blank-region
train
dupli-padding
augmentation
concat
feature extract
blank-region
time/frequency

trial 2


test
test
smart-padding
feature extract
blank-region
original
train
smart-padding
padded train
augmented
feature extract
blank-region
concat
dupli-padding
padded concat
augmented
train

Post-processing
blank region clipping

-
높은 주파수 대역의 빈 영역을 제거하기 위해 적용

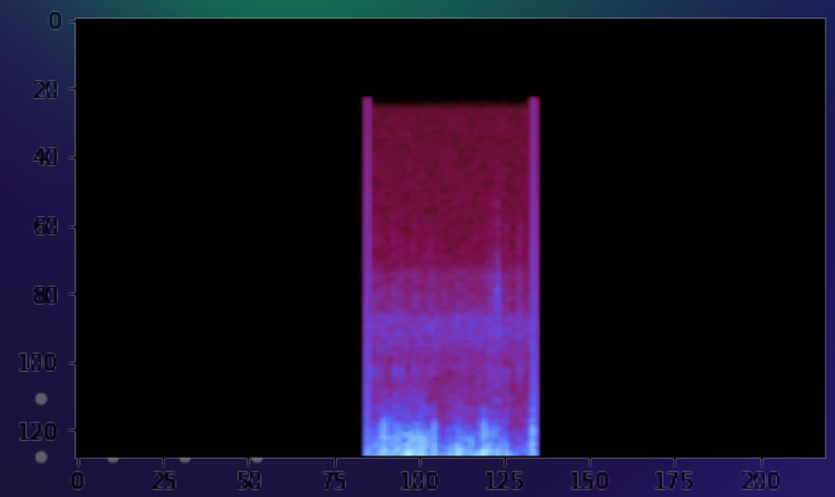
라이브러리

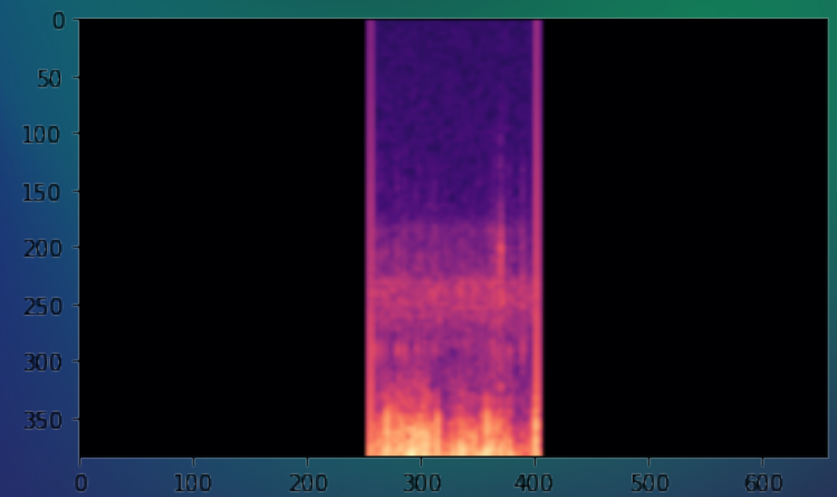
Post-processing
blank region clipping

-
높은 주파수 대역의 빈 영역을 제거하기 위해 적용

라이브러리

librosa



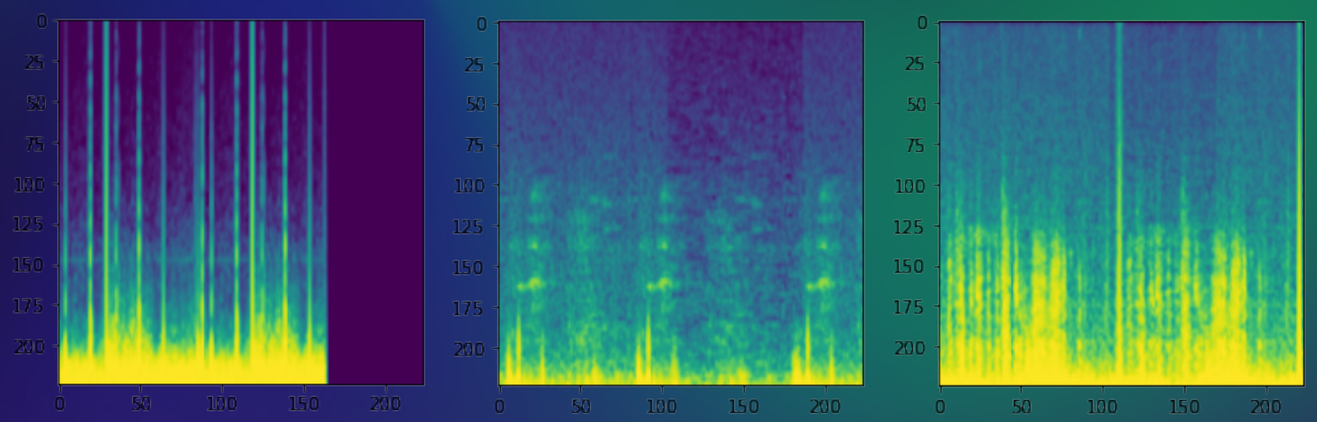


Post-processing
blank region clipping

-
높은 주파수 대역의 빈 영역을 제거하기 위해 적용

라이브러리

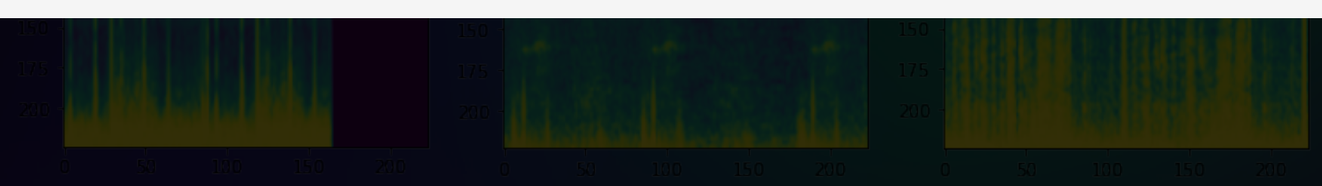
librosa



Post-processing

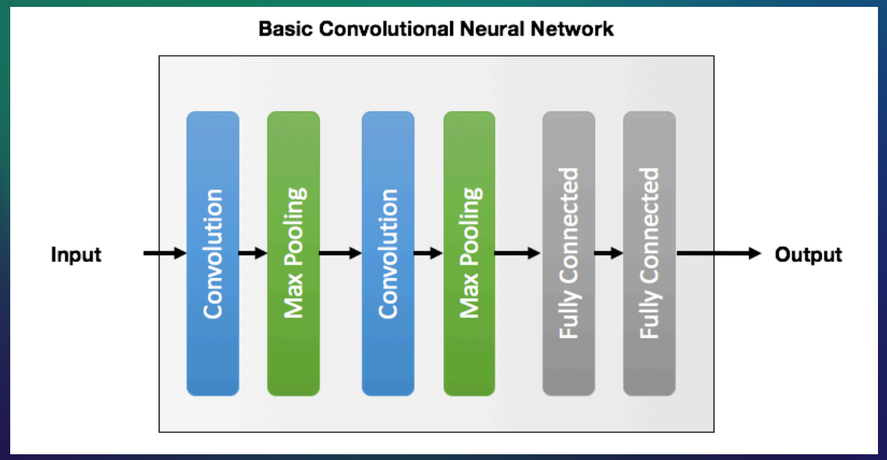
blank region clipping

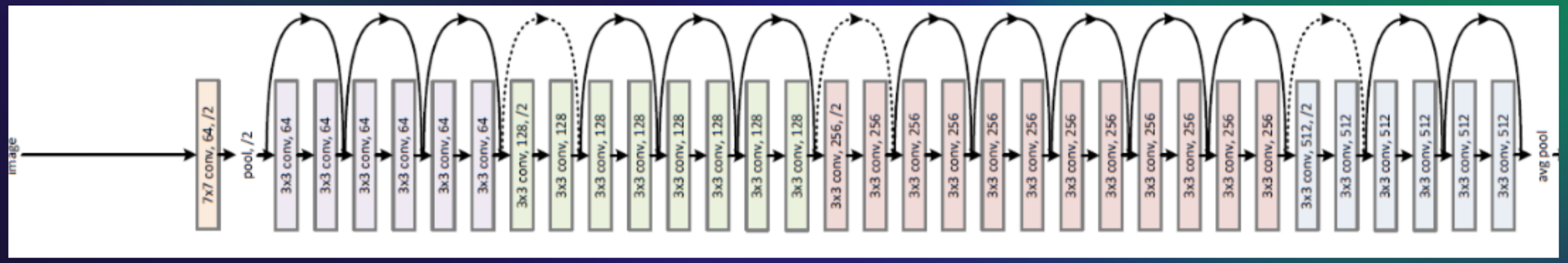
```
#blank region clipping
blank_clipped_aug = []
for idx in range(len(x_train)):
    audio_image = cv2.cvtColor(create_mel_raw(np.array(aug_arr)[idx][0], sr, n_mels, f_min, f_max, nfft, hop, resz=3), cv2.COLOR_BGR2RGB)
    audio_raw_gray = cv2.cvtColor(create_mel_raw(np.array(aug_arr)[idx][0], sr, n_mels, f_min, f_max, nfft, hop), cv2.COLOR_BGR2GRAY)

    audio_raw_gray[audio_raw_gray < 10] = 0
    for row in range(audio_raw_gray.shape[0]):
        black_percent = len(np.where(audio_raw_gray[row,:]==0)[0])/len(audio_raw_gray[row,:])
        if black_percent < 0.80:
            break

if (row+1)*3 < audio_image.shape[0]:
    audio_image = audio_image[(row+1)*3:, :, :]
audio_image = cv2.resize(audio_image, (audio_image.shape[1], n_mels*3), interpolation=cv2.INTER_LINEAR)
audio_image, _, __ = cv2.split(audio_image)
audio_image = cv2.resize(audio_image, (224, 224), interpolation=cv2.INTER_LINEAR)
audio_image = np.array(audio_image, dtype=float)

blank_clipped_aug.append(audio_image)
```



Build Model

simple CNN, ResNet-34





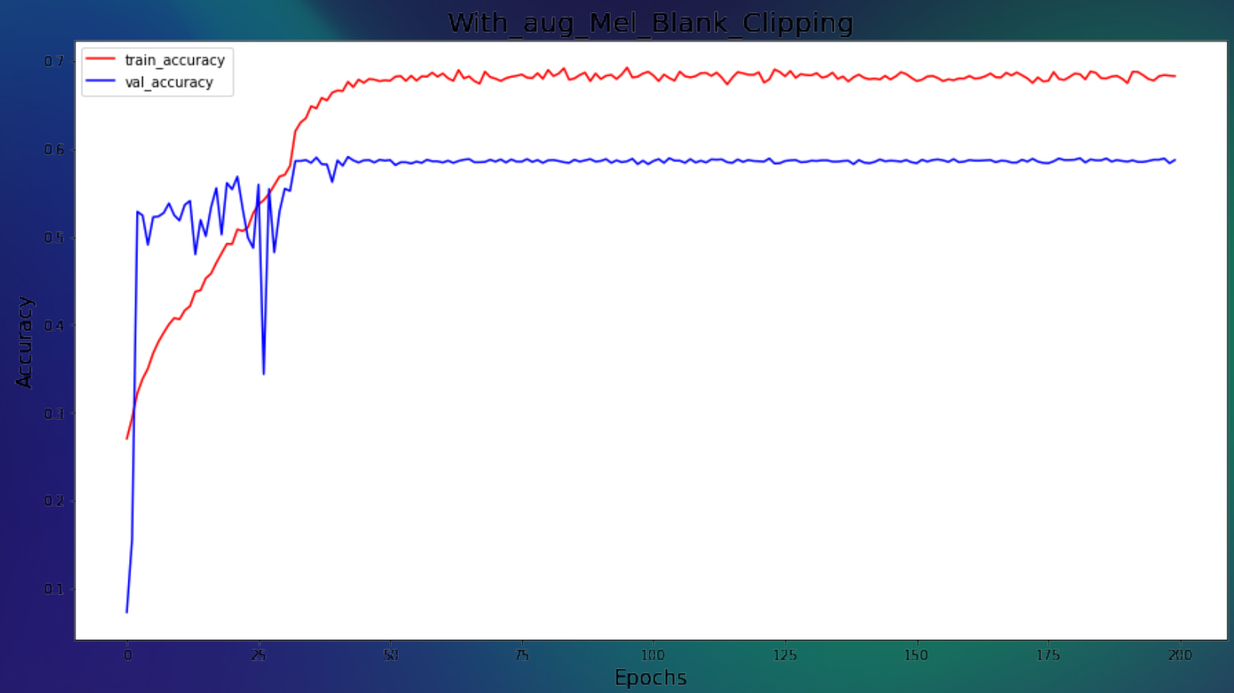
Report

f1 score, recall

-

classification report

	precision	recall	f1-score	support
Normal	0.68	0.73	0.71	1457
Crackle	0.47	0.48	0.47	746
Wheeze	0.52	0.38	0.44	355
Both	0.36	0.32	0.34	202
accuracy			0.59	2760
macro avg	0.51	0.48	0.49	2760
weighted avg	0.58	0.59	0.58	2760



Reference
참고 논문

-

- A Window Width Optimized S-Transform(2008)
- Classification of lung sounds using convolutional neural networks(2017)
- respireNet(2020)
- DC-UNet: Rethinking the U-Net Architecture with Dual Channel Efficient CNN for Medical Images Segmentation(2020)
- LungRN+NL: An Improved Adventitious Lung Sound Classification Using Non-Local Block ResNet Neural Network with Mixup Data Augmentation(2020)
- FILTERAUGMENT: AN ACOUSTIC ENVIRONMENTAL DATA AUGMENTATION METHOD(2021)

감사합니다!