인공지능 기말 프로젝트

호흡기 소리 분석을 통한 질병유무 판단과 질병분류

14011890 김기홍
15010958 이가경
15011037 김동익
17013252 이상민

호흡기 소리 분석을 통한 질병유무 판단과 질병분류

01 배경 및 데이터 셋

02 기존 연구

03 CNN을 이용한 학습

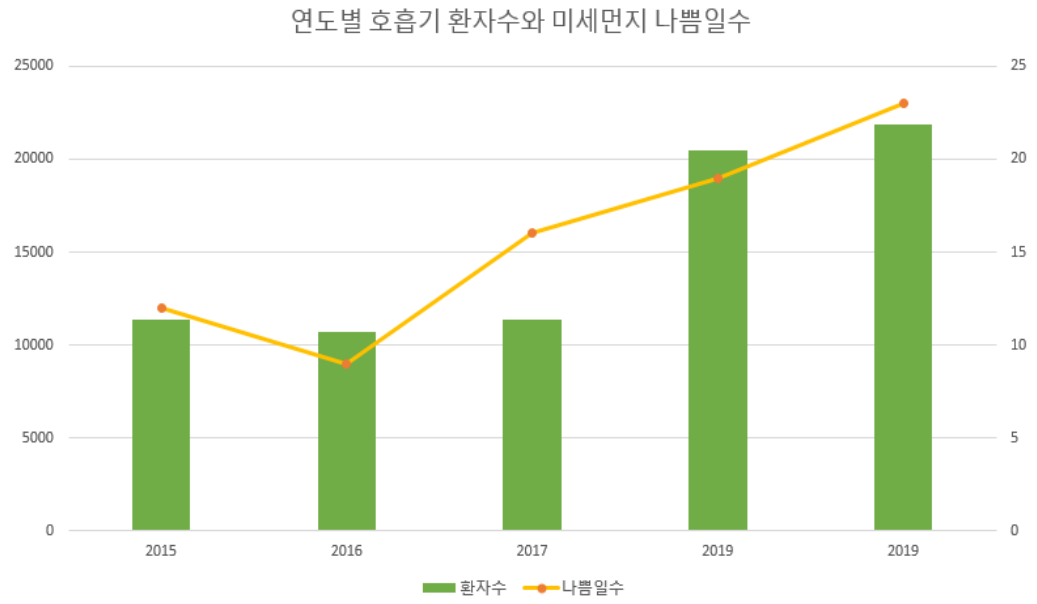
04 SVM을 이용한 학습

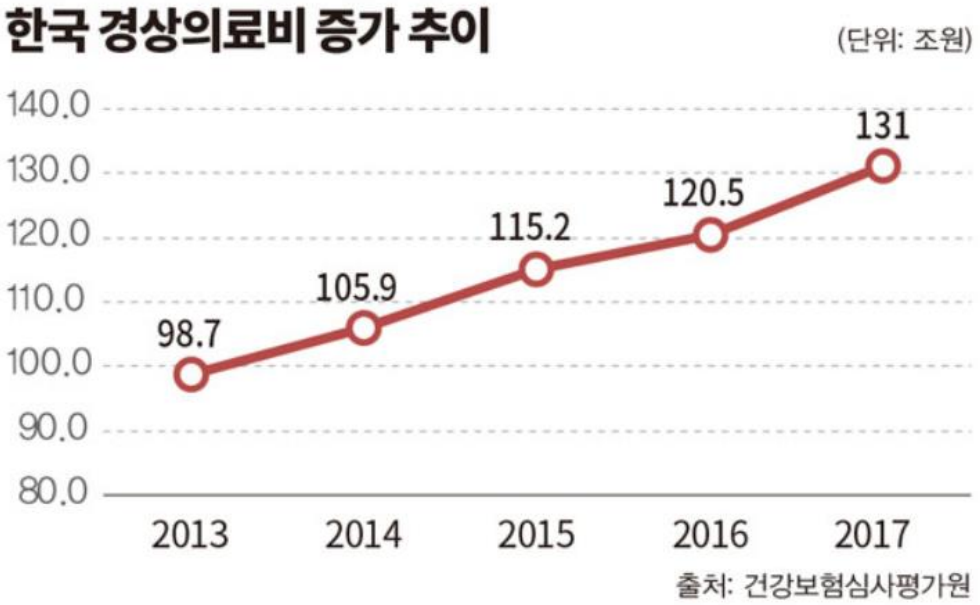
01

배경 및 데이터 셋

01. 배경

호흡기 질환과 디지털 헬스케어





이슈 미세먼지 비상

서울 초미세먼지 농도·나쁨일수 4년 새 2배 급증

고영득 기자 godo@kyunghyang.com

미세먼지 영향?…1분기 119 이용한 호흡기질환자 최근 5년간 최다

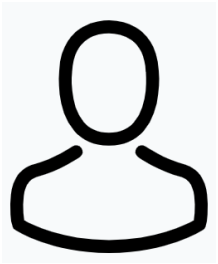
박진규 기자 승인 2019.04.10 13:15 댓글 0

01. 데이터 셋

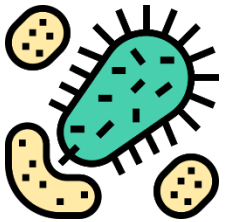
ICBHI Challenge Respiratory Sound Database

ICBHI Challenge

ESSUA 연구실
D. Pedro, Aveiro 병원



126 명



7개 질환

만성 폐쇄성 폐 질환, 상부 호흡기 감염, 하부 호흡기 감염, 천식, 기관지 확장 증, 폐렴, 세기관지염

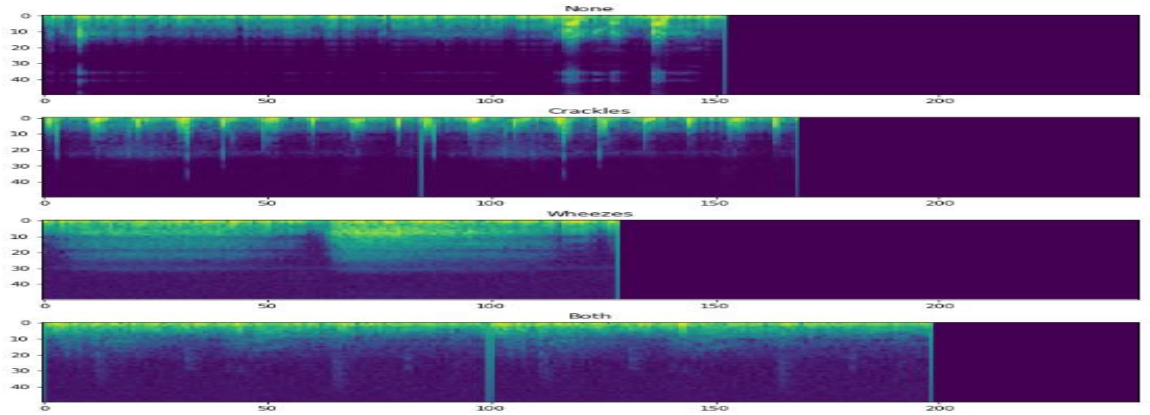
920개 .wav

02

기존 연구

02. 기존연구

Detection of wheezes and crackles with CNN



107_3p2_Pr_mc_AKGC417L_events - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

0.885	0.915	crackle
0.924	0.952	crackle
0.959	1.284	wheeze
1.004	1.026	crackle
1.026	1.048	crackle
1.051	1.073	crackle
2.412	2.444	crackle
3.250	3.270	crackle
3.481	3.940	wheeze

1. None / Others ➡ Accuracy : 84%

2. None / Only Wheeze /

Only Crackle / Wheeze and Crackle

➡ Accuracy : 70%

02. 기존연구

Detection of wheezes and crackles with CNN Problem

Patient 109
COPD

No Wheezing

109_1b1_Lr_sc_Litt3200_events

파일(F)	편집(E)	서식(O)	보기(V)
1.284	2.905		crackle
3.853	4.164		crackle
5.783	6.129		crackle
7.649	7.920		crackle
9.442	9.768		crackle
11.324	11.660		crackle
14.445	14.682		crackle
17.197	17.456		crackle

➢ **COPD : wheezing + crackling + Rhonchi + Mediastinal Crunch**

Problem

03

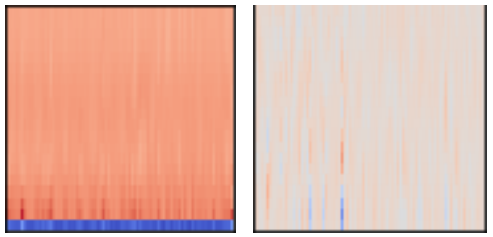
CNN을 이용한 학습

03. CNN을 이용한 학습

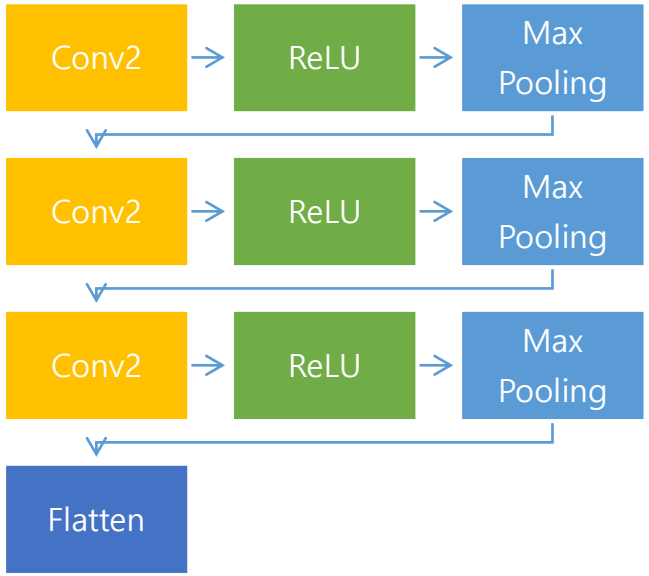
학습 순서

Accuracy
??%

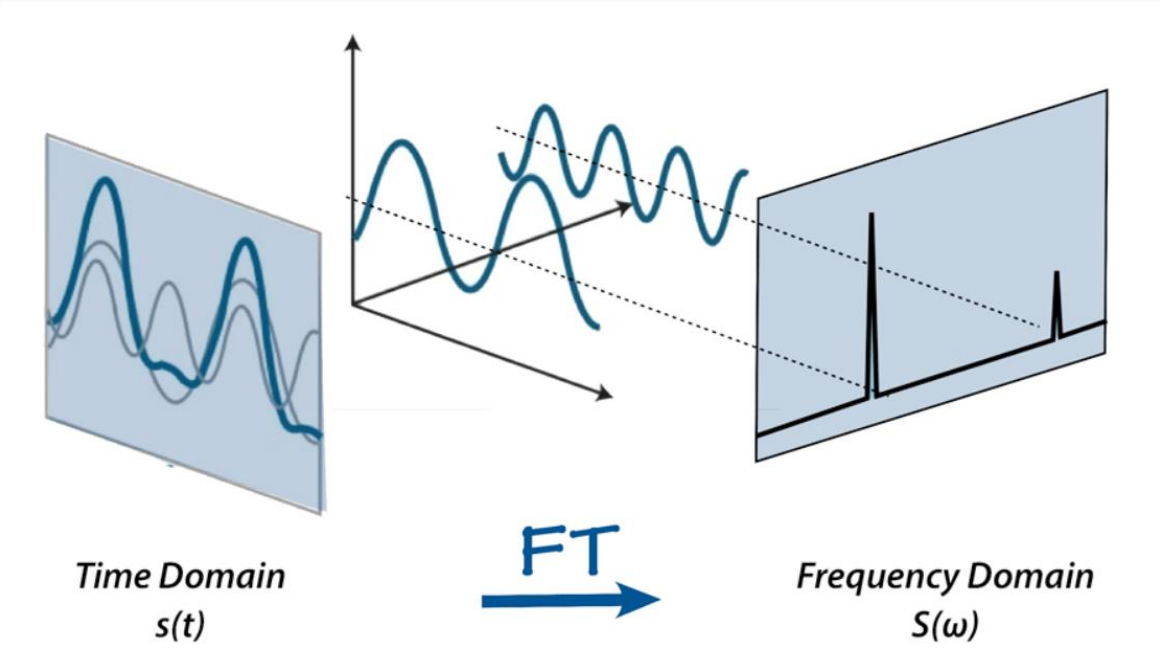
특징 값 추출



CNN 학습



03. FT(Fourier Transform)



- FT(Fourier Transform) 이란?

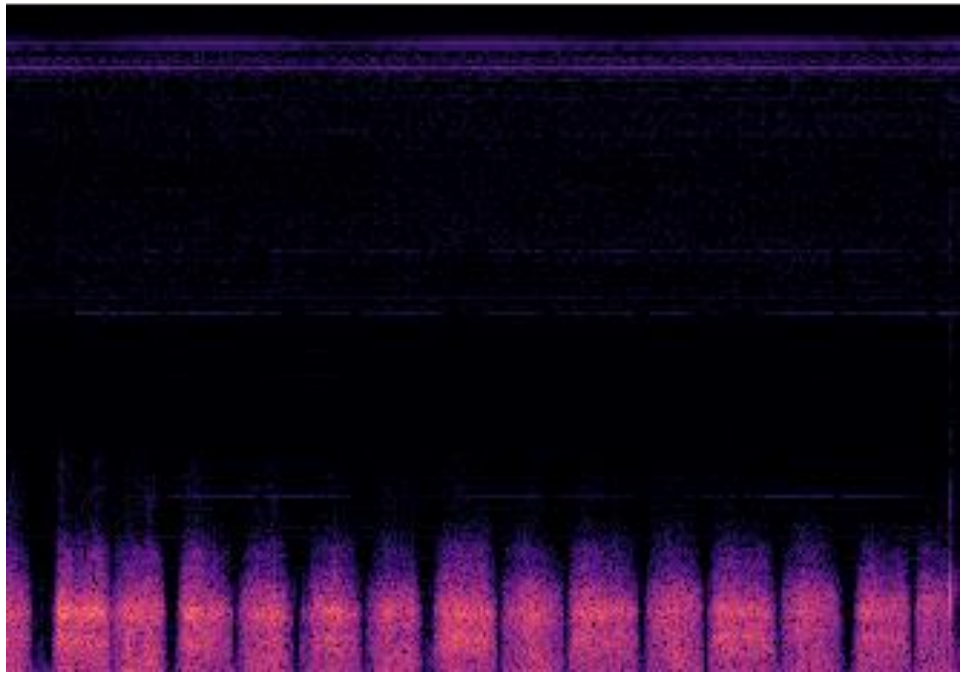
  입력 신호를 대응하는 스펙트럼으로 전환

  연속시간의 아날로그 파형을 infinite Fourier series

- Infinite Fourier series

  특정 amp와 phase를 가지는 사인파로 변환.

03. STFT(Short Time Fourier Transform)



- FT(Fourier Transform)을 실제 녹음된(유한의) 소리에 적용하기 위해 만든 것.
- 데이터에서 시간에 대해 구간을 짧게 나누어 FT (Fourier Transform) 적용.
- 음성파일의 전체길에 비례하여 구간을 나누기 때문에 사용하지 않음

03. MFCC(Mel Frequency Cepstral Coefficient) – 6 steps

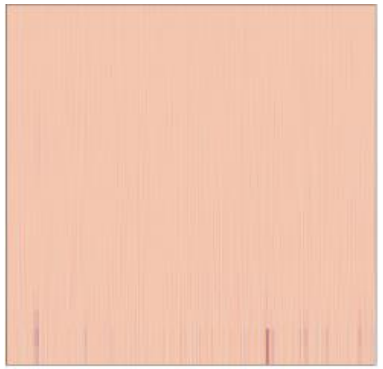
- Divided into small frames
- Calculate Periodogram estimate
- Mel filter bank
- Log
- DCT
- 12~13 Coefficients

03. MFCC+DELTA

MFCC

Once Derivative

Twice Derivative



MFCC + delta

음성의 변화율을
스팩트럼 이미지로 생성

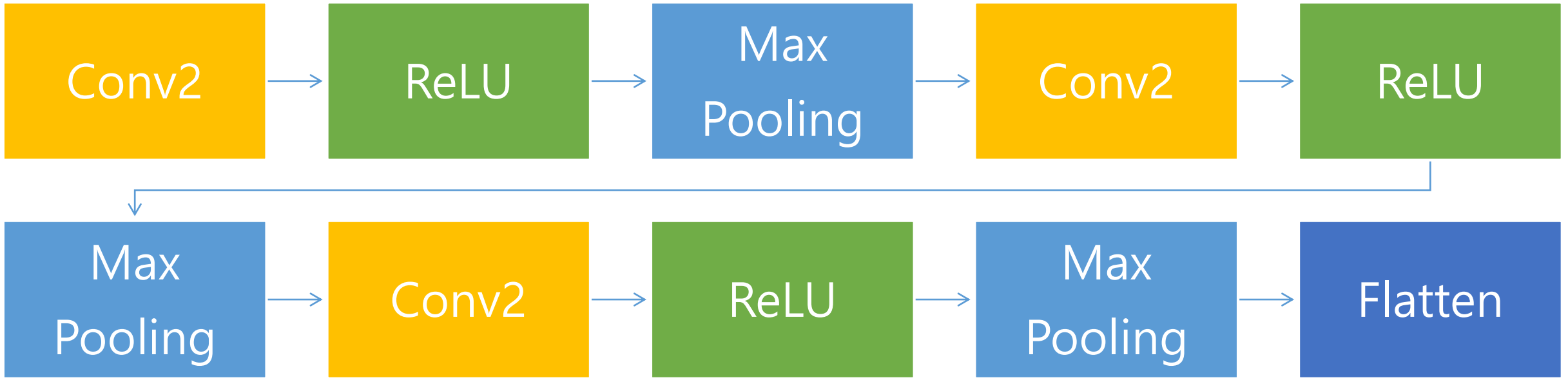
MFCC+delta+delta

음성 데이터 변화의 가속도를
스팩트럼 이미지로 생성

청진 데이터는 일정한 간격으로 반복되기 때문에 MFCC Delta만 적용

03. CNN

분류기


Conv2
ReLU
Max Pooling
Conv2
ReLU
Max Pooling
Conv2
ReLU
Max Pooling
Flatten

03. 모듈화 – 경로설정, 파일

```
test_index_list = []
test_answer_list =[]
train_answer_list = []
size_ = 2
random_len = 0

filename = np.array(pd.read_csv('C:/Users/DONG/Desktop/A.I.data/test_train/data/SC_filename.csv'))
answer = np.array(pd.read_csv('C:/Users/DONG/Desktop/A.I.data/test_train/data/SC_answer.csv'))

path_test='C:/Users/DONG/Desktop/A.I.data/SC_56_version_mfcc/test'
path_train='C:/Users/DONG/Desktop/A.I.data/SC_56_version_mfcc/train'
```

1. 음원 파일명을 담은 filename.csv, 음원 라벨링이 분류된 answer.csv 파일 로드

2. Path는 저장할 위치를 설정

03. 모듈화 – test, train set 설정

```
def make_rand_num(total_count):
    tf.set_random_seed(777)
    test_rate = 0.2
    rand_num = random.randint(0,total_count)
    range_num = int(total_count*test_rate)
    for i in range(range_num):
        while rand_num in test_index_list:
            rand_num = random.randint(0,total_count)
        test_index_list.append(rand_num)
    test_index_list.sort()
```

1. Test와 train 데이터 셋을 랜덤으로 생성하기 위해서 test에 포함될 파일 index test_index_list에 담기
2. 중복 제거

03. 모듈화 – 특징 값 추출 함수

```python
def mfcc(total_count):
    pointer = 0
    step = 0
    print(len(filename))
    for step in range(total_count):
        #mfcc
        audio_path='C:/Users/DONG/Desktop/A.I.data/test_train/data'+filename[step]
        print(audio_path)
        y, sr = librosa.load(audio_path[0])
        mfccs=librosa.feature.mfcc(y=y,sr=sr)
        # labeling
        pointer = labling(step,pointer)
        #save image
        save_image(mfccs, answer[step],step)
```

```python
def mfcc_delta(total_count):
    pointer = 0
    for step in range(total_count):
        #mfcc
        audio_path='./data/'+filename[step]
        y, sr = librosa.load(audio_path[0])
        mfcc=librosa.feature.mfcc(y=y,sr=sr)
        mfccs = librosa.feature.delta(mfcc, order=2)
        # labeling
        pointer = labling(step,pointer)
        #save image
        save_image(mfccs, answer[step],step)
```

1. Mfcc, mfcc_delta를 각각 함수로 구현, 원하는 함수를 호출해서 사용
2. Librosa 라이브러리 이용

03. 모듈화 – 이미지 파일 생성

```
def save_image(mfccs, answer_, step):
    temp_name = filename[step][0].split(".")
    plt.figure(figsize=(0.78*2*size_,0.78*2*size_))
    librosa.display.specshow(mfccs)
    if step in test_index_list:
        if answer[step] == 0:
            plt.savefig(path_test+'/test_0/'+ ''.join(temp_name[0])+'.png')
        #else:
            plt.savefig(path_test+'/test_1/'+ ''.join(temp_name[0])+'.png')
    else:
        if answer[step] == 0:
            plt.savefig(path_train+'/train_0/'+ ''.join(temp_name[0])+'.png')
        else:
            plt.savefig(path_train+'/train_1/'+ ''.join(temp_name[0])+'.png')
```

1. 이미지를 test와 train 따라 지정한 경로에 저장

03. 모듈화 – 라벨링, csv 파일 생성

```
def labling(step,pointer):
    if pointer < random_len:
        if step == test_index_list[pointer]:
            test_answer_list.append(answer[step][0])
            pointer = pointer +1;
        else:
            train_answer_list.append(answer[step][0])
    return pointer
```

```
def save_test_lable_csv(name_file):
    path='.C:/Users/DONG/Desktop/A.I.data/test_train/data/csv_file/'+name_file
    csvfile = open(path,"w",newline="")
    csvwriter = csv.writer(csvfile)
    csvwriter.writerows(map(lambda x: [x], test_answer_list))
    csvfile.close()

def save_train_lable_csv(name_file):
    path='C:/Users/DONG/Desktop/A.I.data/test_train/data/csv_file/'+name_file
    csvfile = open(path, "w",newline="")
    csvwriter = csv.writer(csvfile)
    csvwriter.writerows(map(lambda x: [x], train_answer_list))
    csvfile.close()
```

1. Index 배열에 랜덤으로 지정된 값을 저장한다.

2. 값이 들어간 index배열은 csv파일 형태로 저장한다.

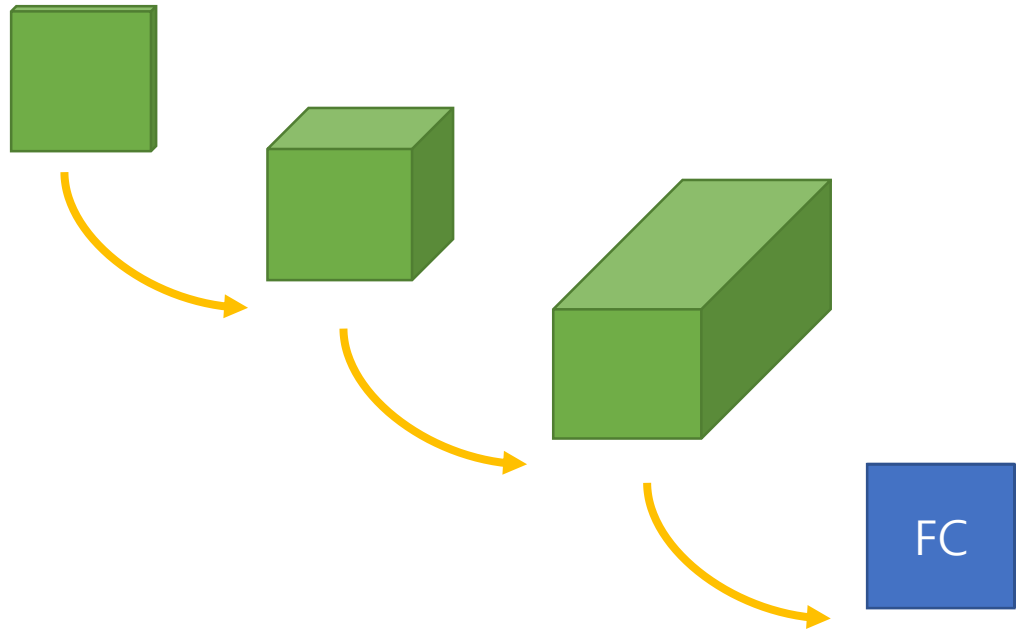
03. 모듈화 – 특징 값 추출 완성

1. 총 920개의 데이터가 설정한 비율에 따라 Train, Test폴더로 나뉘어 라벨링된 값에 따라 적용한 필터의 이미지가 나뉘어 생성

03. 모듈화 – CNN

Processing modulization

```
# hyper parameters
learning_rate = 0.001

# input place holders
X = tf.placeholder(tf.float32, [None, 3136*4])
X_img = tf.reshape(X, [-1, 112, 112, 1])   # img 28x28x1 (black/white) #img 356x238x1
Y = tf.placeholder(tf.float32, [None, 8])

W1 = tf.Variable(tf.random_normal([3,3,1,32],stddev =0.01))
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
tf.layers.batch_normalization(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')


W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
L2 = tf.nn.relu(L2)
tf.layers.batch_normalization(L2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

W3 = tf.Variable(tf.random_normal([3, 3, 64, 128], stddev=0.01))
L3 = tf.nn.conv2d(L2, W3, strides=[1, 1, 1, 1], padding='SAME')
L3 = tf.nn.relu(L3)
tf.layers.batch_normalization(L3)
L3 = tf.nn.max_pool(L3, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

L3_flat = tf.reshape(L3, [-1, 7 * 7 * 512])
W4 = tf.get_variable("W4", shape=[7 * 7 * 512, 8], initializer=tf.contrib.layers.xavier_initializer())
b = tf.Variable(tf.random_normal([8]))
logits = tf.matmul(L3_flat, W4) + b

# define cost/loss &amp;amp;amp; optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=logits, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```



03. 질병 유무 판단, 분류 결과

	MFCC	MFCC + Delta	Average
112X112	97.8	95.9	96.85
56X56	96.7	93.9	95.3
28X28	95.1	98.3	96.7
14X14	97.2	96.1	96.65

CNN을 이용한 질병 유무 판단

Average : 96.38%

	MFCC	MFCC + Delta	Average
112X112	82.2	86.8	84.5
56X56	84.1	88	86.05
28X28	84.7	86.4	85.55
14X14	82.6	86.4	84.5

CNN을 이용한 질병 분류

Average : 85.15%

03. 질병분류 정확도 향상을 위한 조건 변경

	0.7	0.8	0.9	Average
112X112	84.9	84.5	84.2	84.53
56X56	85.45	86.05	89.1	86.87
28X28	85.95	85.55	82	84.5
14X14	86.3	84.5	83.75	84.85
avg	85.65	85.15	84.7625	

CNN을 이용한 질병분류 Training rate 변경

	0.01	0.001	0.0001	Avg
56X56	85.5	85.5	85.5	85.5

CNN을 이용한 질병분류 Learning rate 변경

	100	200	300	400	avg
56X56	84.6	84.6	84.6	84.6	84.6

CNN을 이용한 질병분류 Batch size 변경

	Case1	Case2	Case3	avg
56X56	88	85.8	86.35	86.72

CNN을 이용한 질병분류 training, test set 변경

Average : 85.15% → **86.72**

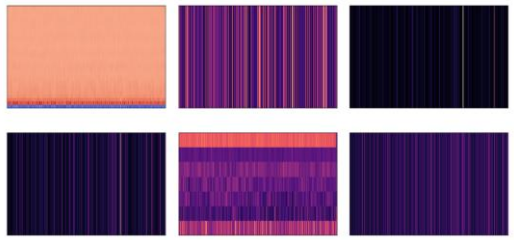
04

SVM을 이용한 학습

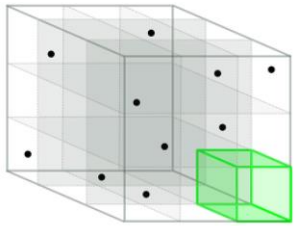
04. SVM을 이용한 학습

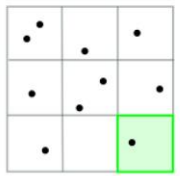
Accuracy
??%

특징 값 추출

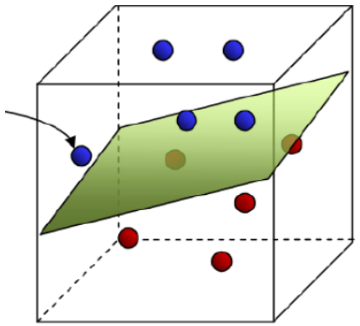


차원 축소

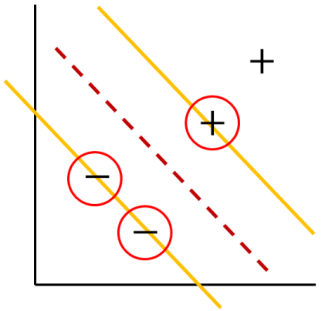




RBF 커널 사용

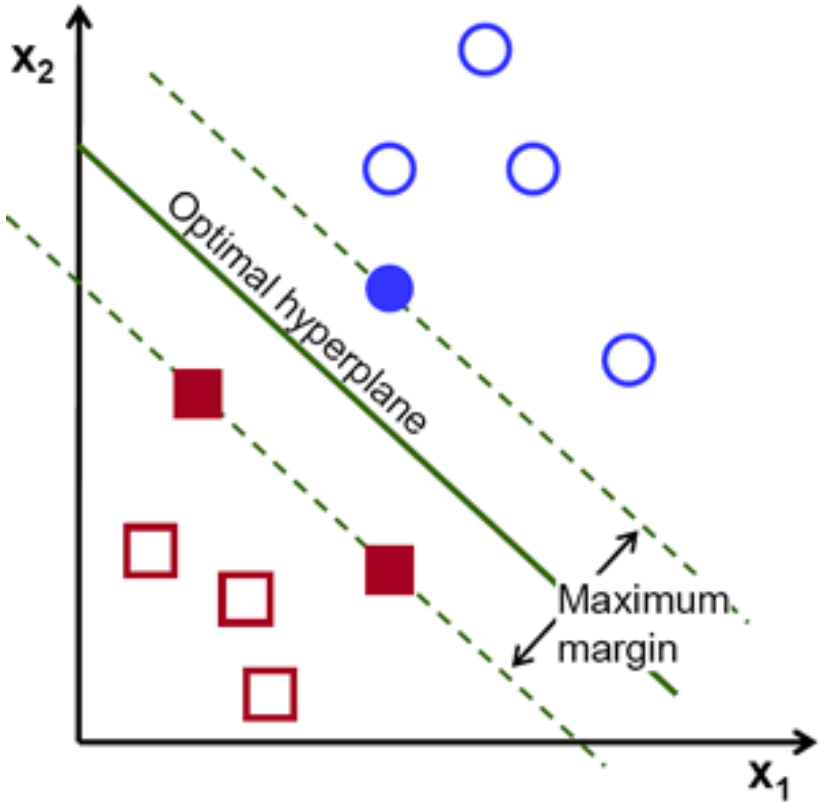


SVM 학습



04. SVM 학습 진행

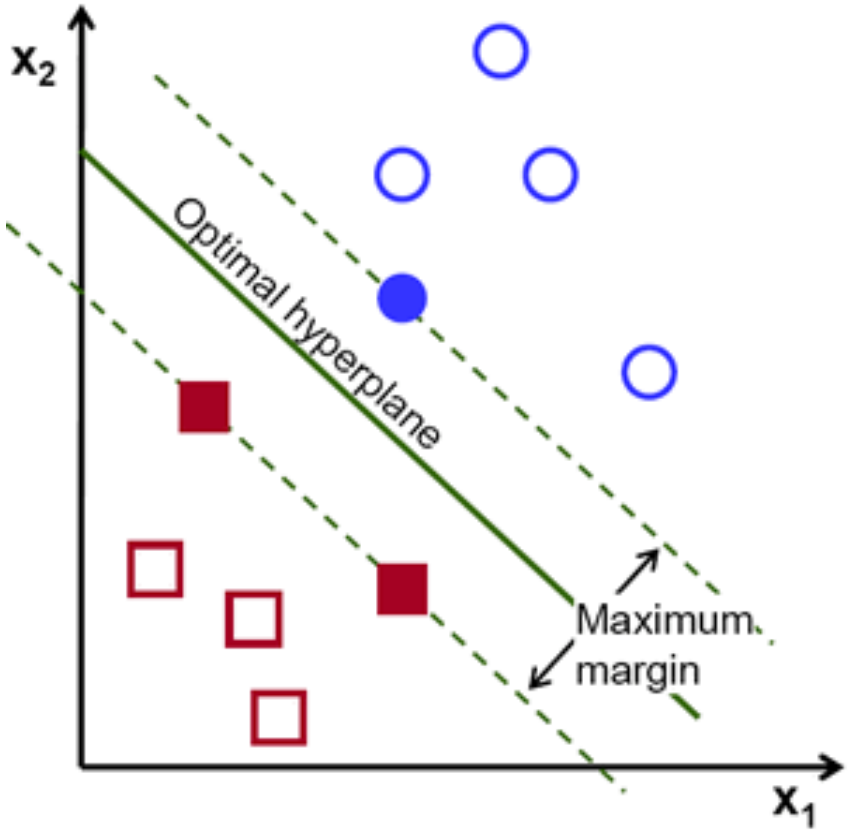
Support Vector Machine



1. 지도 학습 방식의 대표 분류 기법인 SVM(Support Vector Machine) 학습법 적용
2. SVM은 선형분류와 더불어 비선형 분류에도 사용
3. 차원 수 > 데이터 수인 경우 효과적

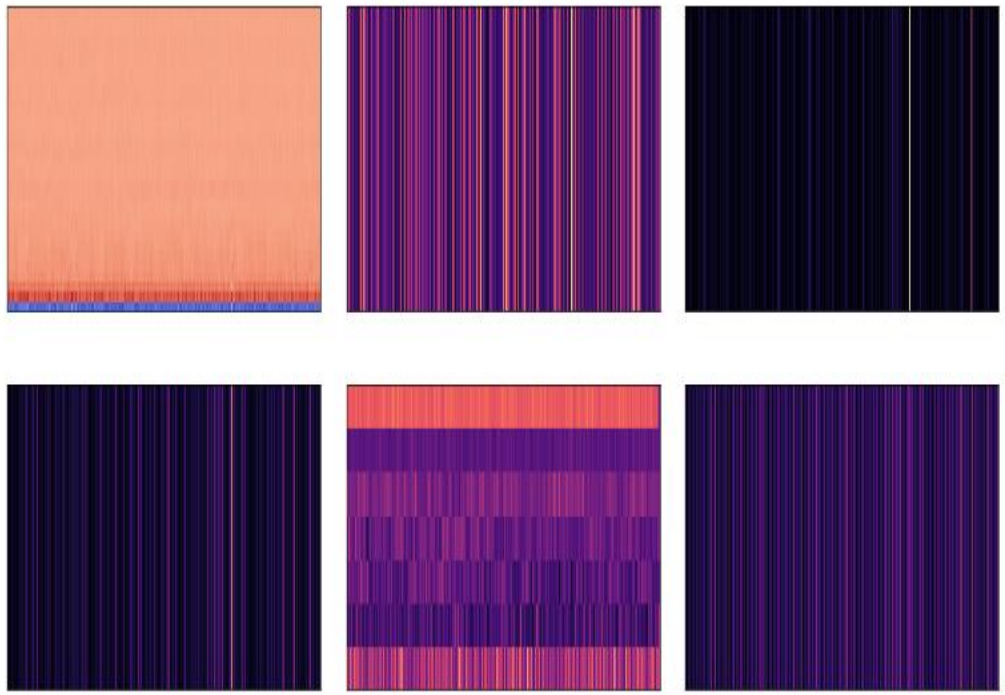
04. SVM 학습 진행

Support Vector Machine



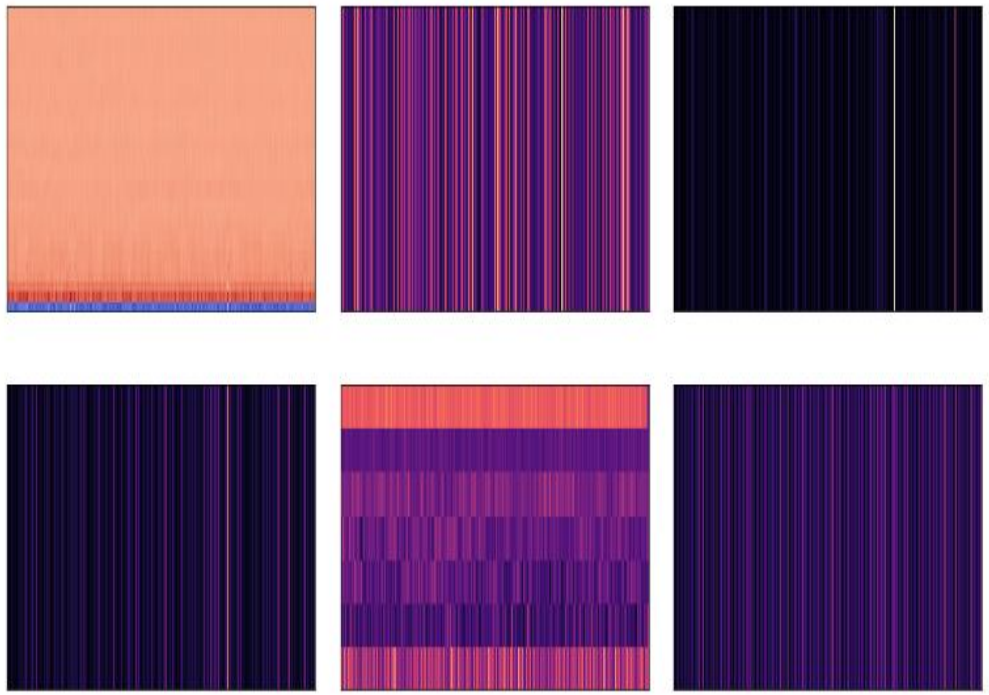
1. SVM은 데이터를 벡터공간으로 표현한 후 서포터 벡터간 거리를 최대화 하는 방식으로 데이터를 분류
2. 비선형 분류의 경우 데이터를 고차원 특징 공간으로 만들어 분류 진행

04. 특징 추출



1. librosa.feature를 통해 음원의 특징 추출
2. 각 음원파일마다 6가지 특징을 뽑아내어 특징을 정규화 시킨 후 특징을 묶음

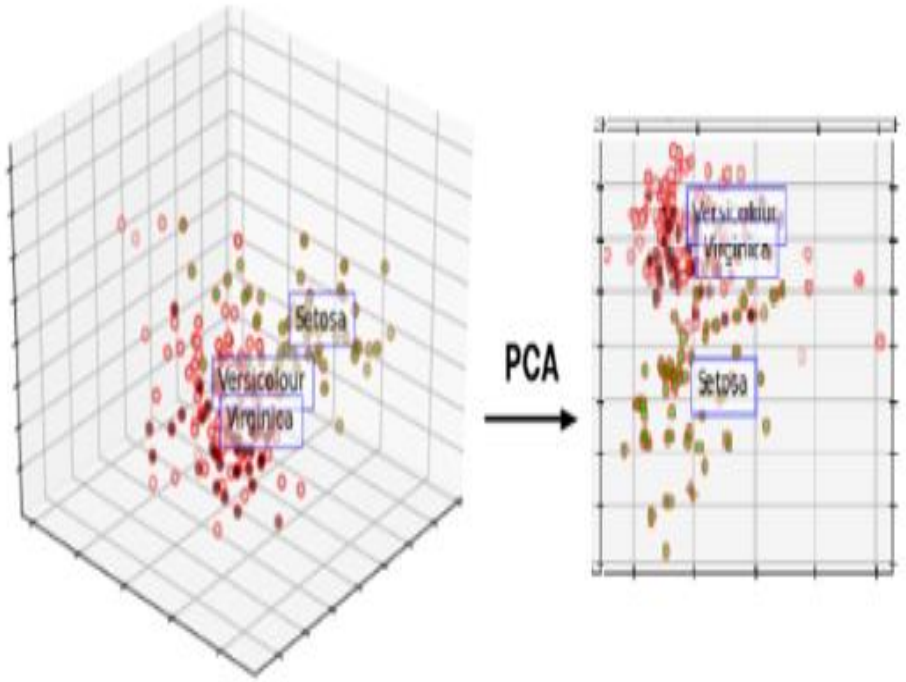
04. 특징 추출



1. librosa.feature를 통해 음원의 특징 추출
2. Mfcc
3. Zero_crossing_rate : 음성 신호의 smoothness를 측정하여 소리 구분
4. Spectral_rolloff : 낮은 주파수 영역에 신호의 에너지가 얼마나 집중되어 있는지 측정
5. Spectral_centroid : STFT의 magnitude 스펙트럼의 중심을 측정
6. Spectral_Contrast : 매 프레임별 6개의 구역으로 나누어 스펙트럼의 peak, valley 차이점 계산
7. Spectral_bandwidth : 주파수의 대역폭을 측정

04. 차원축소

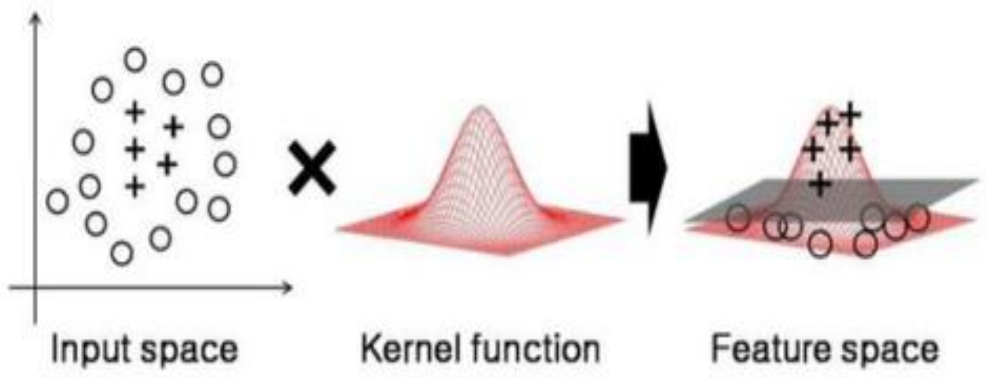
Dimensional Reduction



1. 특징이 많으면 기계학습 모델이 잘 훈련되지 않거나 과적합을 일으키고, 훈련된 모델을 해석하여 용이한 정보를 얻기 어려움
2. 주어진 특징들을 조합하여 새로운 특징 값을 계산하는 주성분분석(PCA)방법 이용

04. RBF 커널 사용

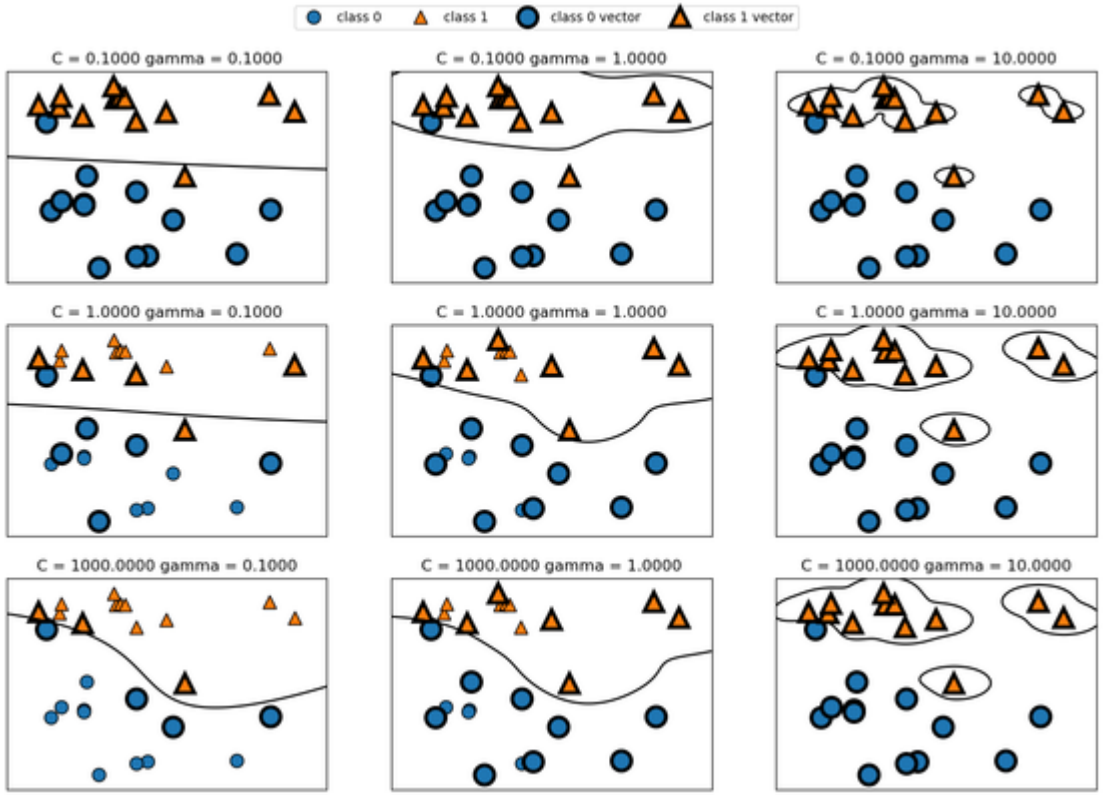
Using RBF(Radial Basis Function) kernel to



1. 선형으로 구분하지 못하는 구조를 RBF 커널을 통해 데이터 변환.
2. RBF 커널은 Cost, Gamma 값들을 parameter로 받으며 Overfitting을 막기 위해 적절한 C, G값이 필요

04. Cost, Gamma 값 설정

Using RBF(Radial Basis Function) kernel to



1. Cost 값이 작을때 제약이 큰 모델을 만들고 각 벡터포인트의 영향력을 적게 받음
2. Gamma 값은 커질수록 하나의 벡터포인트에 더 민감하게 반응
3. Grid 통해 가장 적절한 C, G값을 찾아주는 선택해서 사용

04. 질병 유무 판단, 질병분류 결과

Train Data 수	정확도
100개	95%
200개	95%
300개	98%
400개	97%

SVM을 이용한 질병 유무 판단

97%

Train Data 수	정확도
100개	85%
200개	95%
300개	95%
400개	93%
500개	89%
600개	93%
700개	87%
800개	87%

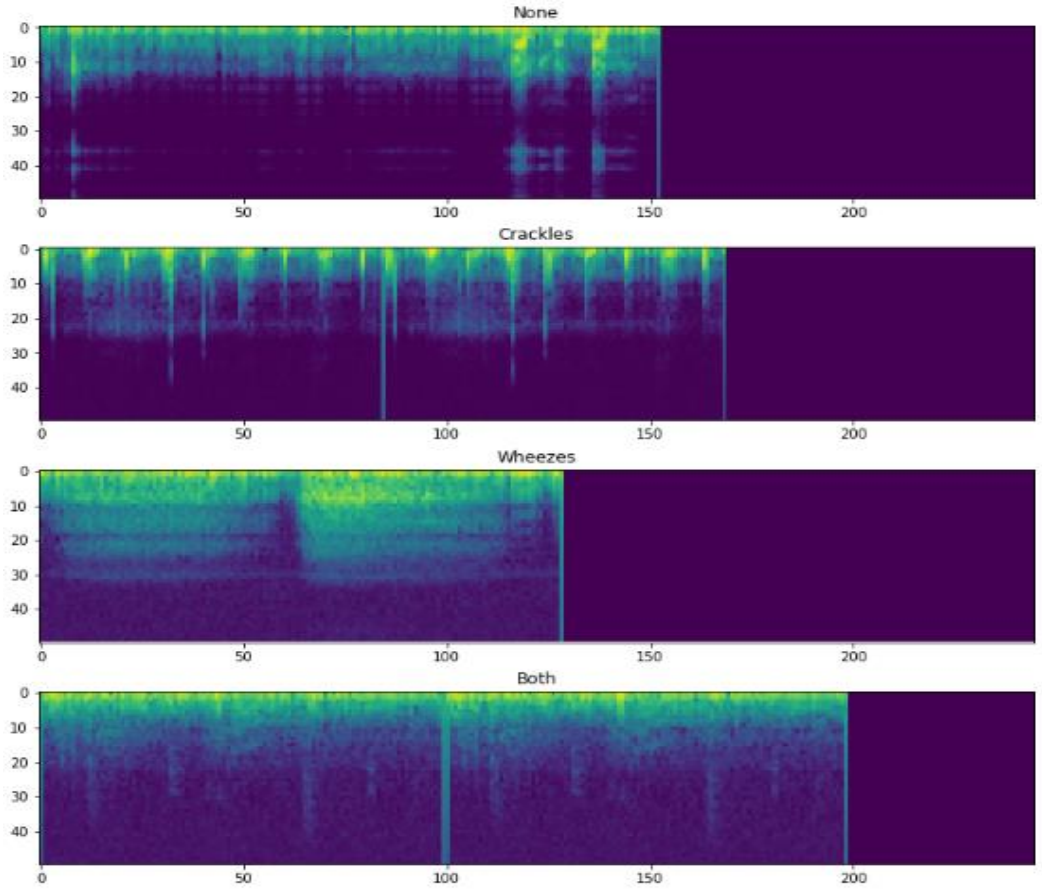
SVM을 이용한 질병 분류

87%

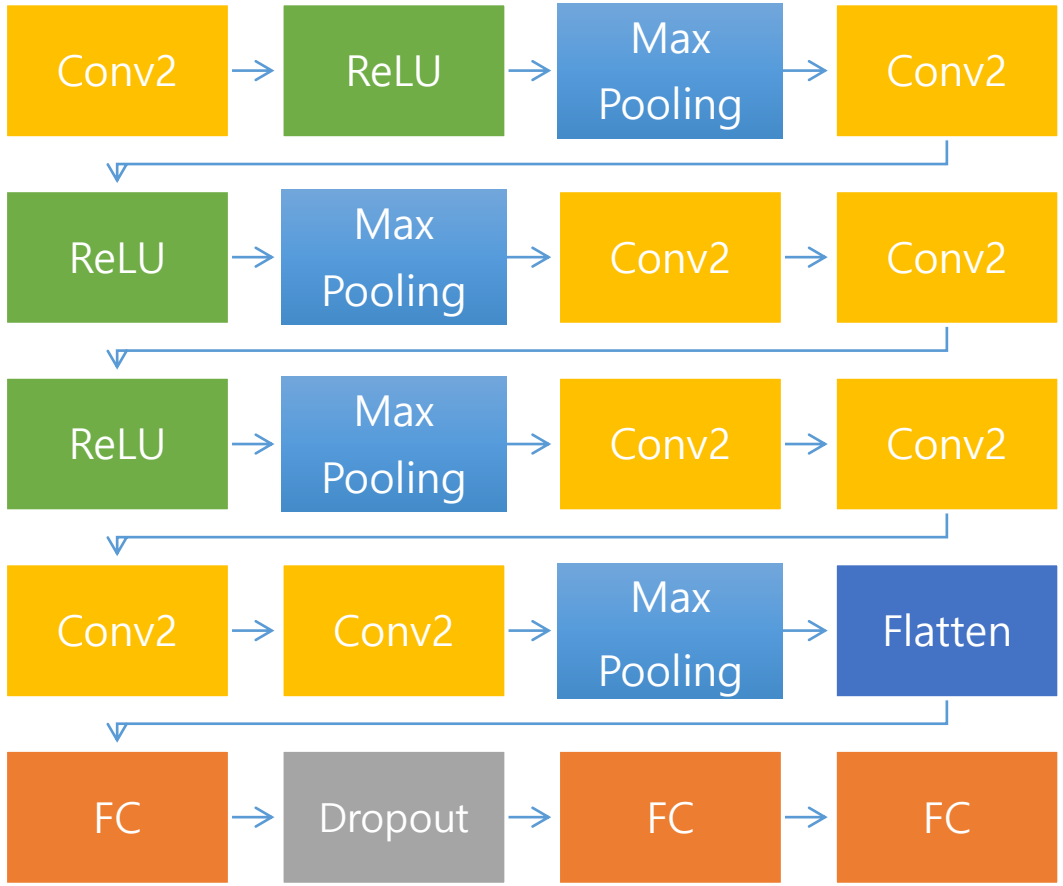
Thank you

02. 기존연구

Detection of wheezes and crackles with CNN



FFT2MelSpectrogram



CNN

02. 기존연구

Detection of wheezes and crackles with CNN

Patient 109
COPD

No Wheezing

109_1b1_Lr_sc_Litt3200_events

서식(O) 보기(V)

1.284	2.905	crackle
3.853	4.164	crackle
5.783	6.129	crackle
7.649	7.920	crackle
9.442	9.768	crackle
11.324	11.660	crackle
14.445	14.682	crackle
17.197	17.456	crackle

- **COPD : wheezing + crackling + Rhonchi + Mediastinal Crunch**
- URTI : wheezing + crackling + Rhonchi + pleural Friction Rub + Mediastinal Crunch
- LRTI : wheezing + crackling + Rhonchi + pleural Friction Rub + Mediastinal Crunch
- Asthma : wheezing
- Bronchinectasis : rhonchi + wheezing
- Pneumonia : wheezing + crackling + pleural Friction Rub + Mediastinal Crunch
- Bronchiolitis : wheezing + crackling + Rhonchi

Problem

04. MFCC(Mel Frequency Cepstral Coefficient) – 6 steps

- Divided into small frames
- Calculate Periodogram estimate
- Mel filter bank
- Log
- DCT
- 12~13 Coefficients