

# ***Dog · Cat · Bird Classify Sound***

Deep Learning project photofolio

**3조**

김수영, 김해원, 유하나, 이한별, 전근호

## Team

[이한별] 조장(프로젝트 총괄) / 발표 / GUI 백업

[김수영] 외부 데이터 수집 / 데이터 정제

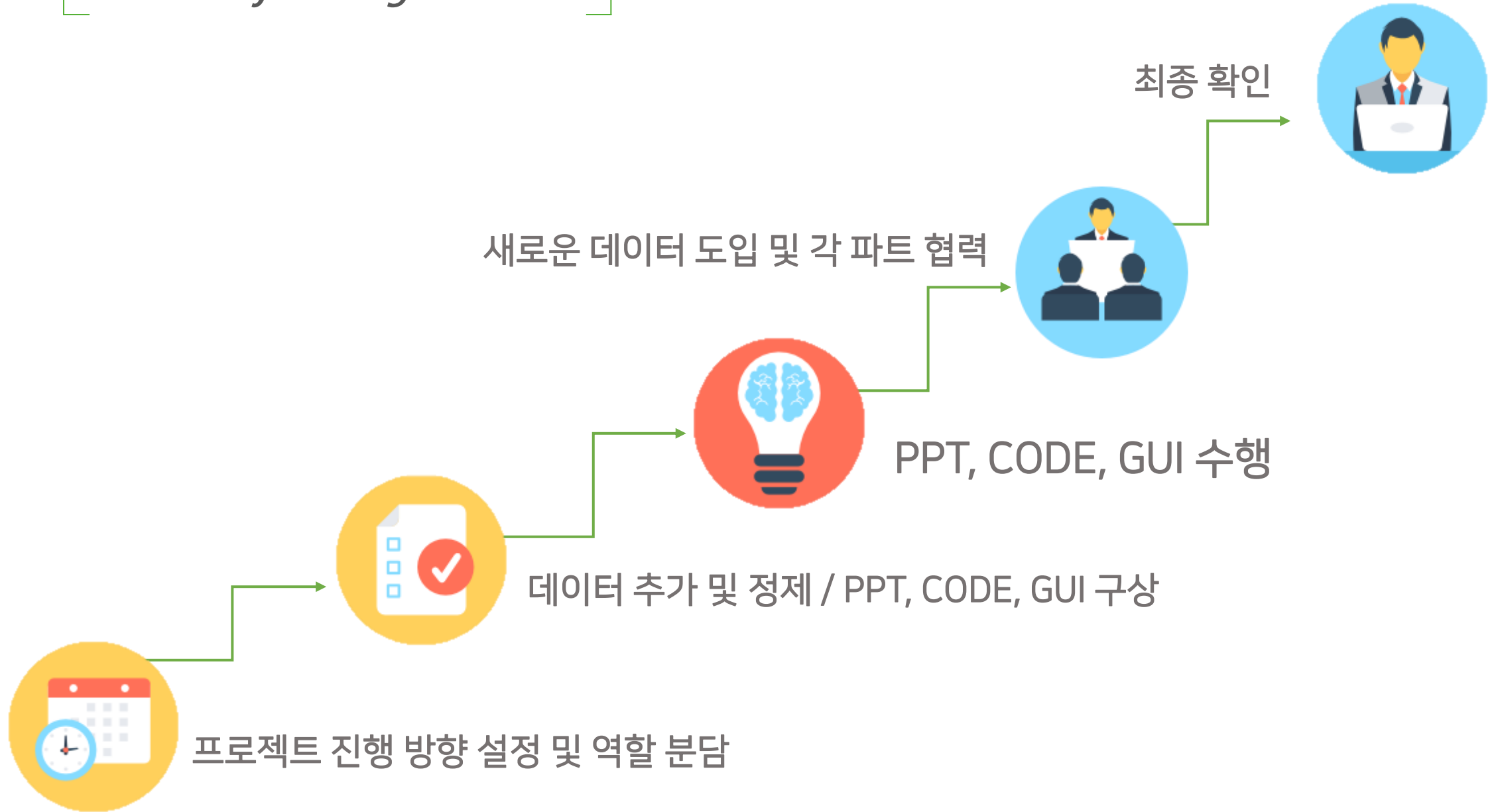
[유하나] 이론 / 데이터 추출 코드 / ppt

[김해원] GUI 총괄 개발 / 디자인

[전근호] 인공 신경망 코드 / 모델구현



## Project Progress



## 개발환경



Window 10 Pro



GPU Intel(R)  
Iris(R) Xe  
Graphics



Python 3.10.0

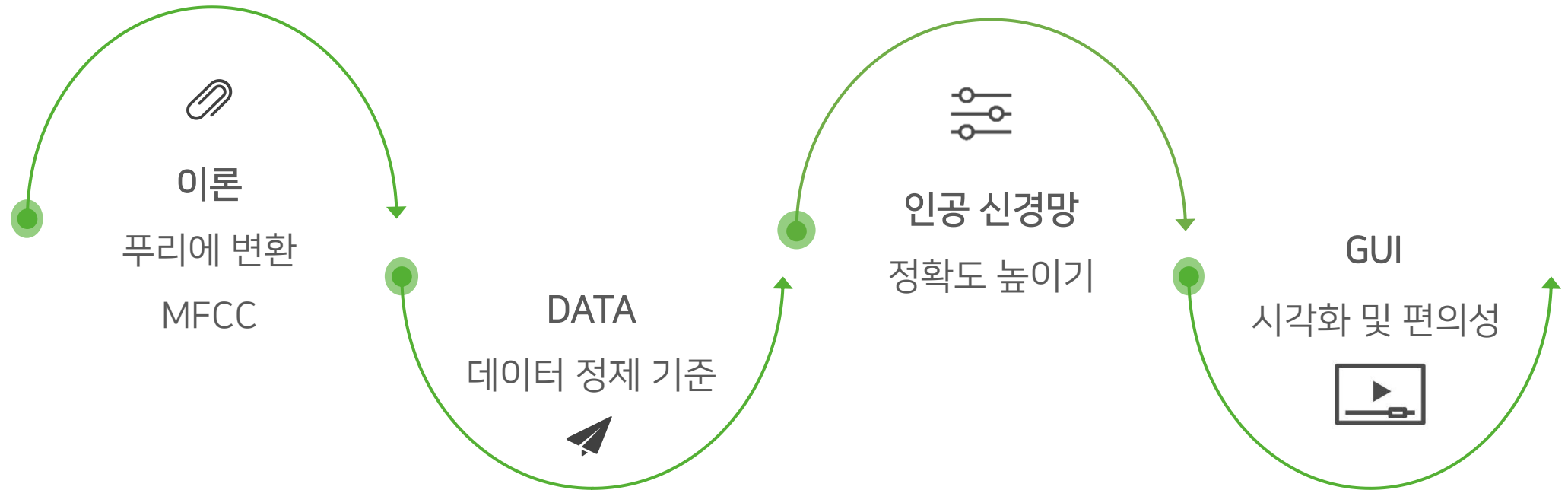


tensorflow 2.8.0



Keras 2.8.0

# Table of Contents



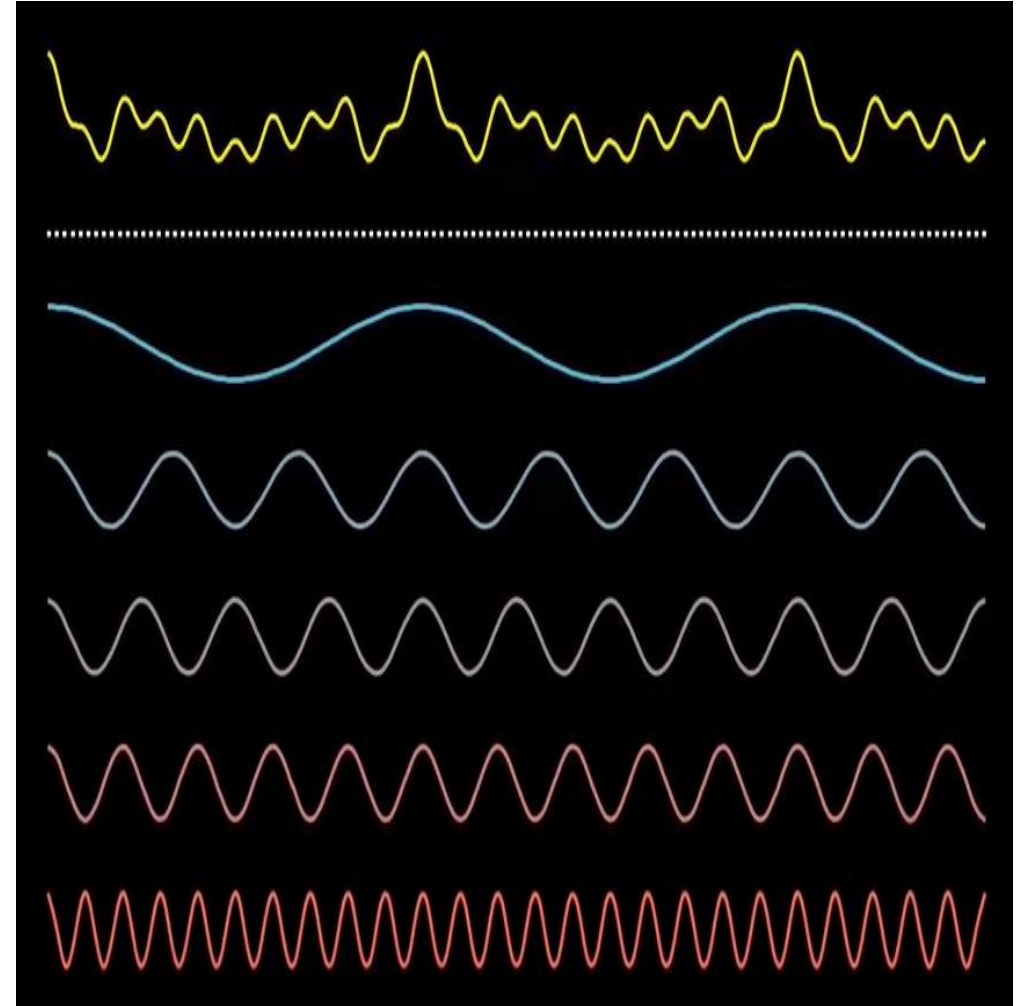


# 1.0이론

## 분류기의 기본 원리 ?

소리의 녹음은  
여러 개의 시점의 기압을 측정하여 이루어진 소리들의 최종 합계  
즉 순수 신호 여러 개의 총 합으로 구성

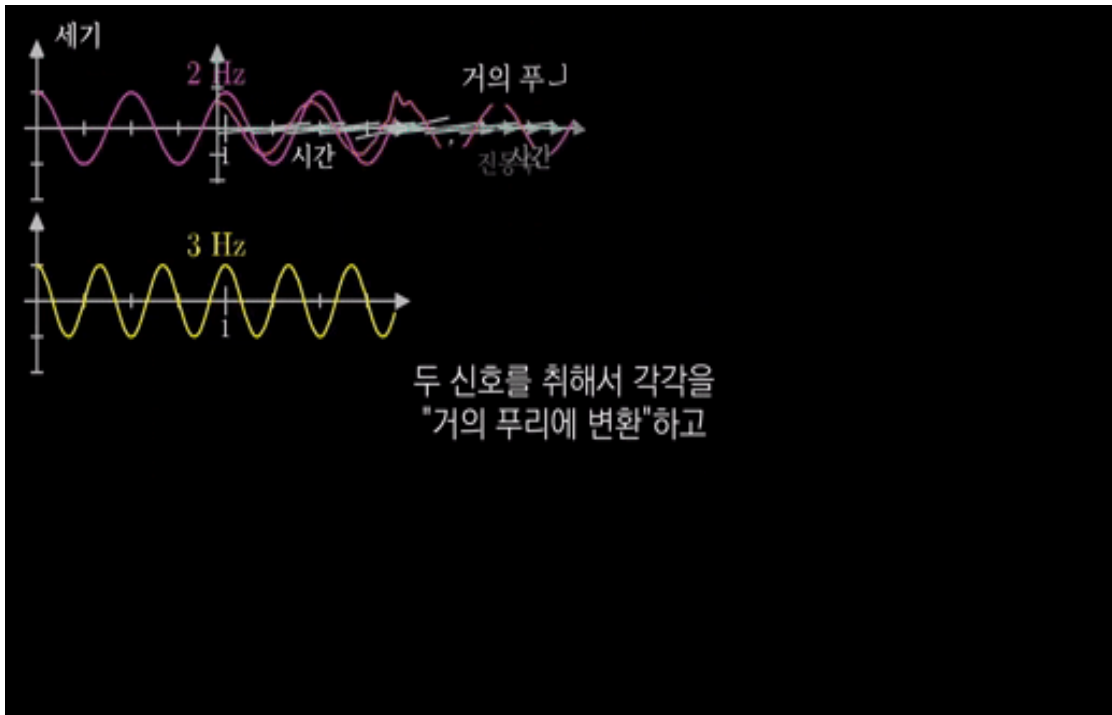
여러 진동수가 뒤섞인 소리에서 원래 신호들을 뽑아내고  
그를 이용해 분류하는 것



# 푸리에 변환 기본 원리 : 시간이 아닌 진동 수(주파수) 관점으로 보자

신호를 진동수의 성분으로 분해하는 수학적 기법

단순한 파동으로 분리하여  
특정 진동수의 세기를 조절할 수 있게 된다.





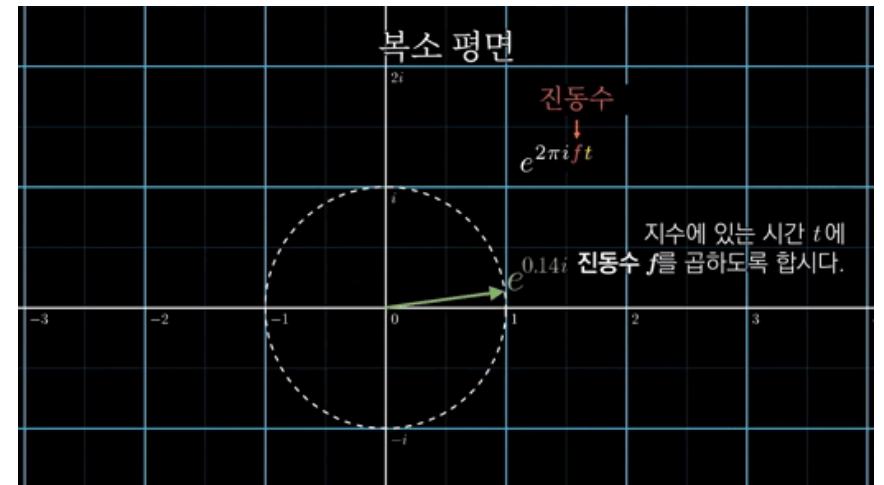
## 1. 이론

### • 푸리에 변환 공식

$$X(f) = \int_{t_1}^{t_2} x(t) e^{-2\pi i f t} dt$$

$e^{-2\pi i f t}$  : 복소수( $e^{-2\pi i t}$ ) 와 진동수( $f$ )를 곱합니다.  
복소수는 회전을 나타내는데 탁월하므로 사용합니다.  
 $2\pi$  는  $2\pi \text{ rad}$ 로 반지름이 1인 단위원의 둘레의 길이를 말합니다.  
 $t$  는 이때 흐른 시간입니다.

$f$  는 진동수로 1/10일 경우 10초동안 1바퀴를 돕니다.

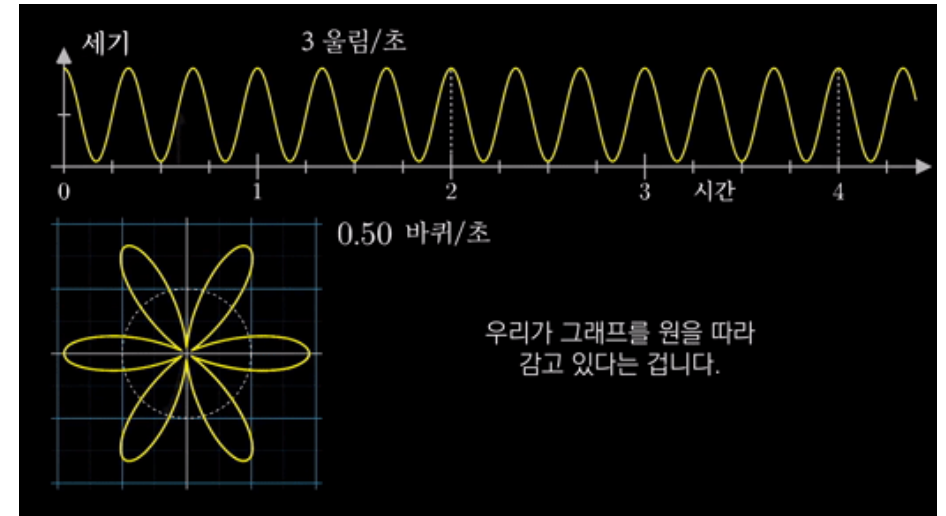


## 1. 이론

- 푸리에 변환 공식

$$X(f) = \int_{t1}^{t2} x(t)e^{-2\pi ift} dt$$

$x(t)e^{-2\pi ift}$  : 위 지수함수에 진동수 그래프를 곱하여 회전 그림으로 만듭니다.

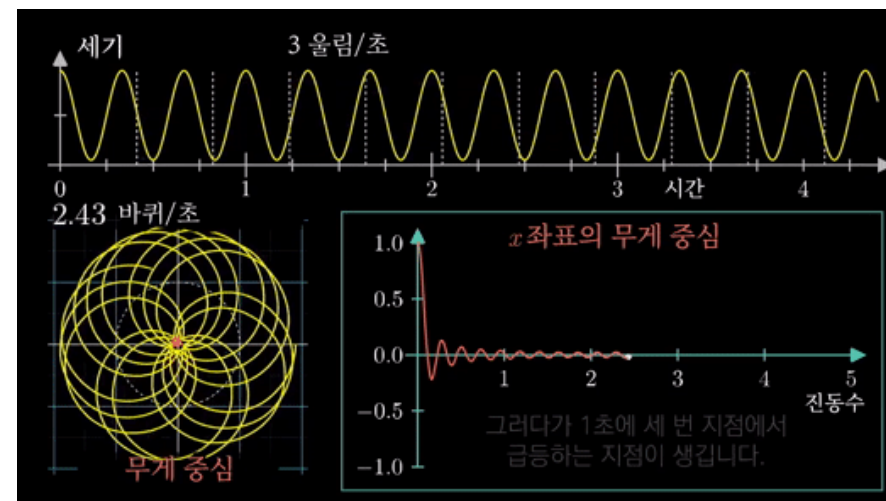


## 1. 이론

- 푸리에 변환 공식

$$X(f) = \int_{t_1}^{t_2} x(t)e^{-2\pi ift} dt$$

진동수에 의해 그려지는 회전원의 한점(x좌표의 무게중심)을 미분하는데  
이는 한점이 주어진 진동수에 따라서 얼마나 신호가 강한지를 알려줍니다.



## • 푸리에 변환 사용 예시 (1)



목소리 구별 : 다른 소리의 성분들을 추출한 뒤 가장 진폭이 큰 성분의 진동수부터 시작해서 2번째, 3번째 ... 순서로 비교해 가다 보면 어느 부분부터는 그 성분의 진동수가 달라진다. 이를 통해 목소리를 구분하거나 흡사하게 만들 수 있다.



잡음 제거 : 여러가지 소리가 섞이는 경우 푸리에 변환을 통해 특정 소리의 주파수를 제거함으로써 잡음 제거에 사용할 수 있다.



다양한 소리를 내는 전자 악기 : 악기에는 몇 가지 주파수의 단순한 파동 발생 회로만 갖고 있으며 연주자가 악기의 종류를 선택하면 그에 해당하는 성분만 적당한 비율로 합쳐서 소리를 만들어 낼 수 있다.

### • 푸리에 변환 사용 예시 (1)



영상 자동 인식  
(지문 저장)

: 주민등록증에 있는 지문을 저장할 때도 푸리에 변환이 활용된다. 먼저 지문 영상을 푸리에 변환을 통해 성분을 추출한 뒤, **다른 사람의 지문과 구별할 수 있는 정도의 성분만 남기고** 더 자세한 성분들은 제거하여 필요 저장 공간은 줄일 수 있다.

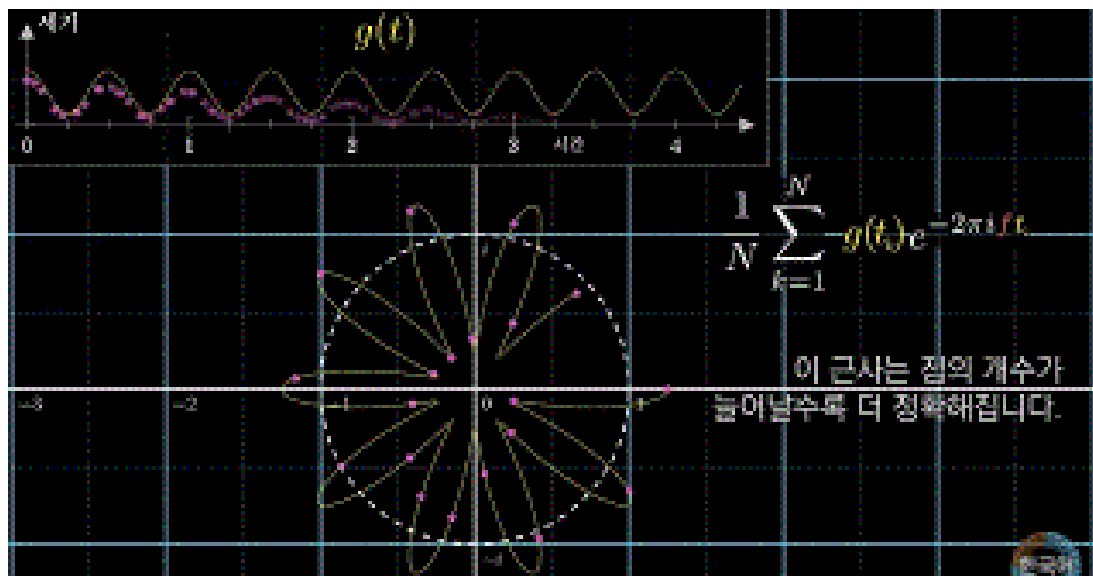


영상 노이즈 제거:

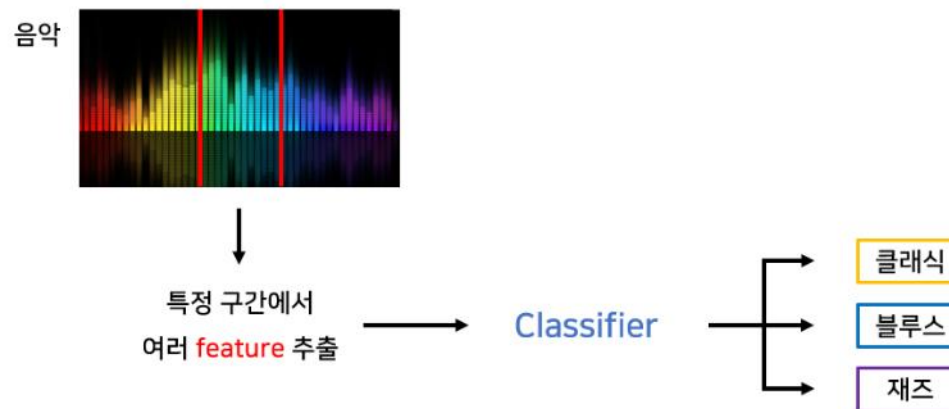
영상 신호도 원하지 않는 데이터가 섞이는 경우가 있다. 이를 노이즈라고 하는데 사진을 확대할 때 흐릿하게 번져 보이는 것도 노이즈 때문이다. 이때 동영상의 앞뒤 장면을 각각 푸리에 변환해 비교하면 **노이즈에 해당하는 성분들을 제거할 수 있다.**

# • MFCC (Mel-Frequency Cepstral Coefficient)

오디오 신호에서 추출할 수 있는 feature로, 소리의 고유한 특징을 나타내는 수치

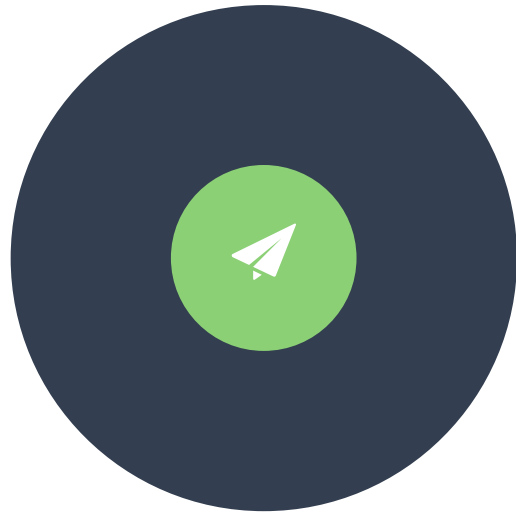


## ❖ 음악 장르 분류(Music Genre Classification):



## "동물 음성 분류"





## 2. Data

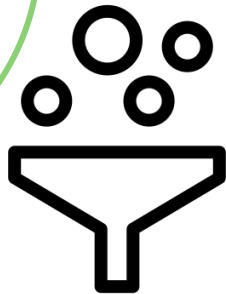


## 2. Data

### Data 목표

#### 1. Refine

데이터를  
깨끗하게 정제

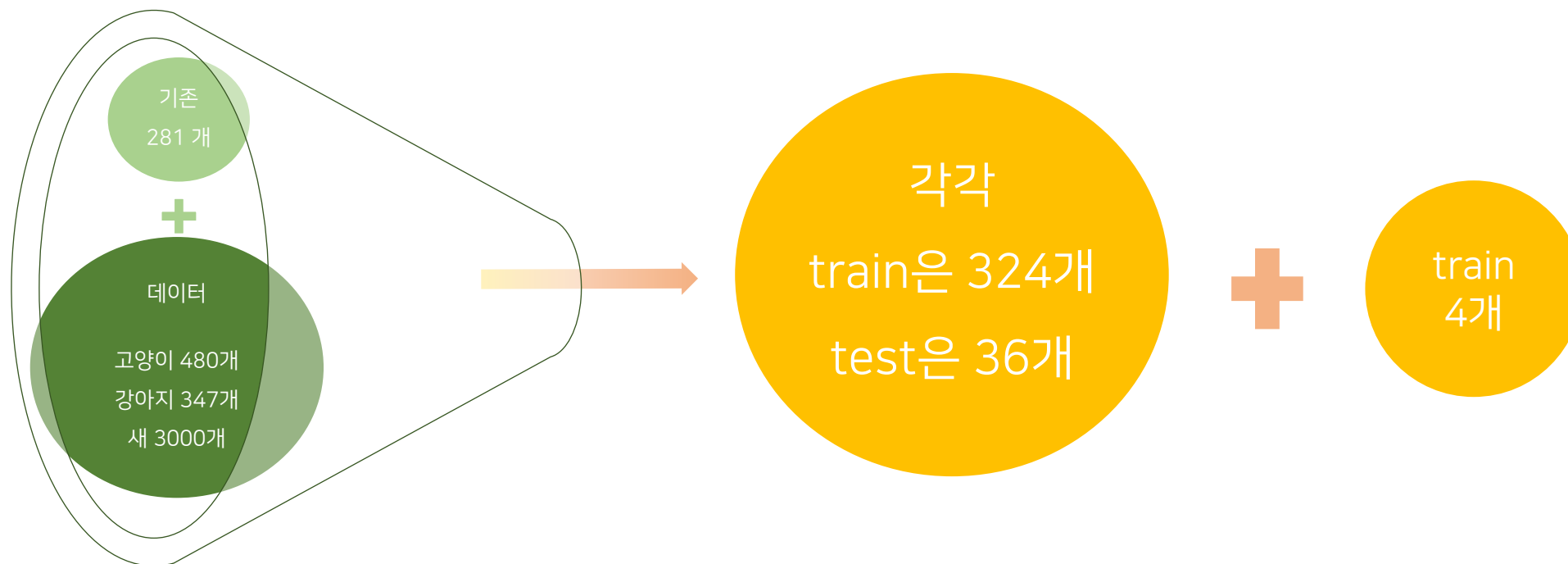


#### 2. equal

모든 종의  
데이터 수집 갯수를  
똑같이 맞추기



## 2. Data



: 대형견 / 소형견 / 노견의 짖는소리, 우는소리, 화난소리 등



: 우는소리, 화난소리, 그르릉소리 등






: 32종의 새 울음소리

(물떼새, 까마귀, 딱새, 종달새, 벌새, 도요새, 독수리, 꾀꼬리, 제비, 참새 등)

# 데이터 정제 기준


### 1. 주변 소음이 큰 데이터 : 노래, 잔디, 자동차, 물 소리 등


- (cat\_112) 
- (cat\_114) 
- (bird\_230) 



# 데이터 정제 기준

## 2. 해당 동물 외에 다른 동물의 소리가 겹쳐있는 데이터

- (dog\_104) 

- (cat\_81) 

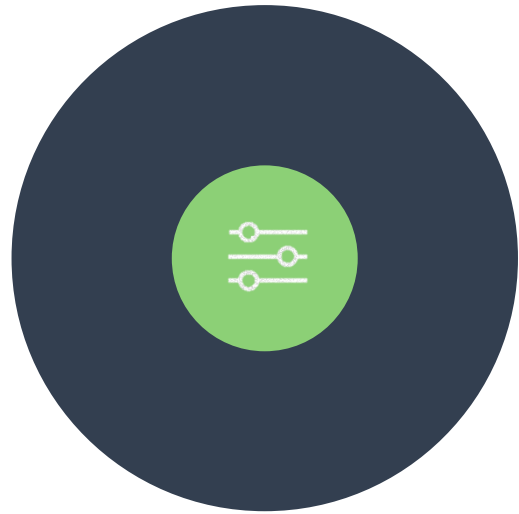
## 3. 15초 이내의 데이터로만 구성하기 위해 긴 음성 데이터는 잘라서 사용

(Dog 수정 전 ; 1분)



(Dog 수정 후 ; 15초)





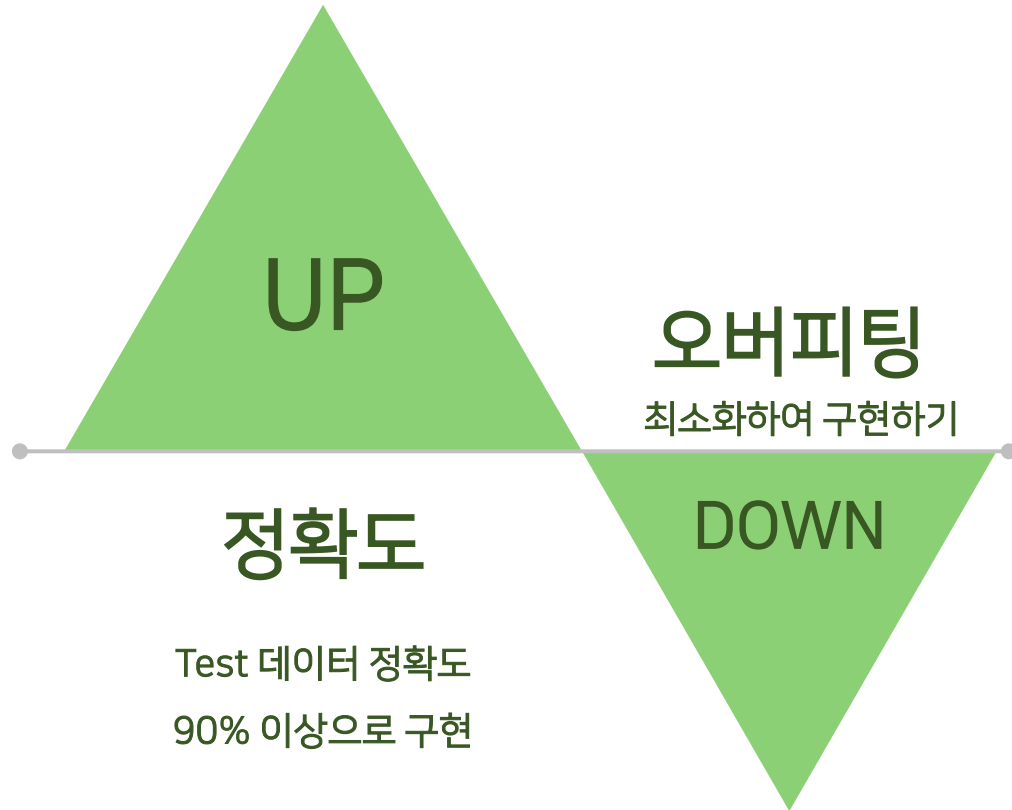
## 3. Code

### 3. Code

[모델 구현 목표]



새로운 데이터를  
올바르게 분류 분석



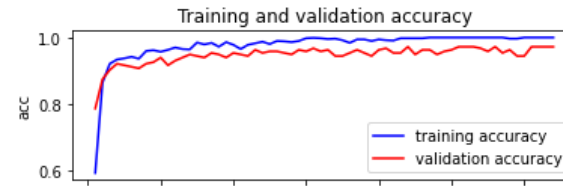
### 3. Code

## 최적의 정확도를 가진 인공지능구현 과정

check

첫번째 시도

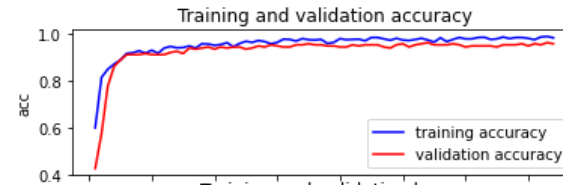
총 3층짜리 신경망(relu사용)  
Optimizer='adam' ,  
loss='categorical\_crossentropy'



훈련데이터 정확도: 100%  
검증데이터 정확도 : 96.07%

두번째 시도

- 배치 정규화, Dropout을 추가  
model.add( BatchNormalization() )  
model.add( Dropout(0.2) )



각각 0.62%, 3.24% 감소

세번째 시도

- 배치정규화, L2 정규화  
kernel\_regularizer='l2'  
model.add( BatchNormalization() )



각각 1.16%, 4.64% 감소

### 3. Code

문제점

고양이 데이터 '만' 오분류 하는 상황



Dog : 99.999%



Bird : 100.0%



새로운 고양이 데이터 (1),(2)를  
넣었을 때 높은 확률로  
Dog 또는 Bird로 예측함



해결방안

## 고양이 데이터 분류 정확도 높이기

	변경 사항	시도한 방법
첫번째 시도	데이터의 비율 8:2 → 9:1	훈련 비율을 높여 학습 훈련을 더 많이 시켜보기
두번째 시도	데이터 추가	틀린 데이터를 훈련데이터에 추가
세번째 시도	데이터 비율 + 데이터 추가	위 두가지 시도를 함께 시도
네번째 시도	GUI의 test데이터를 불러오는 코드 수정	고정적이었던 sample_rate를 수정
다섯번째 시도	하이퍼 파라미터 mfcc 조절	리턴 될 mfcc 갯수 변경

### 3. Code

해결방안

## 고양이 데이터 분류 정확도 높이기

	변경 사항	시도한 방법
첫번째 시도	데이터의 비율 8:2 → 9:1	훈련 데이터 비율을 높여 학습 훈련을 더 많이 시켜보기

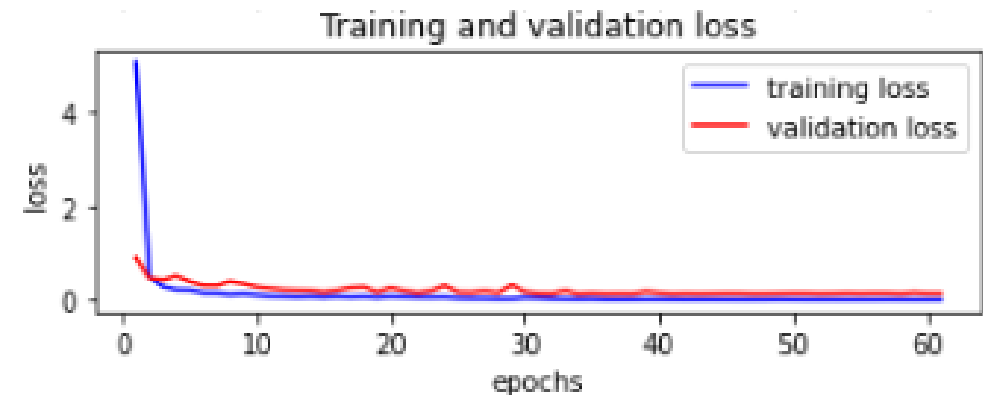
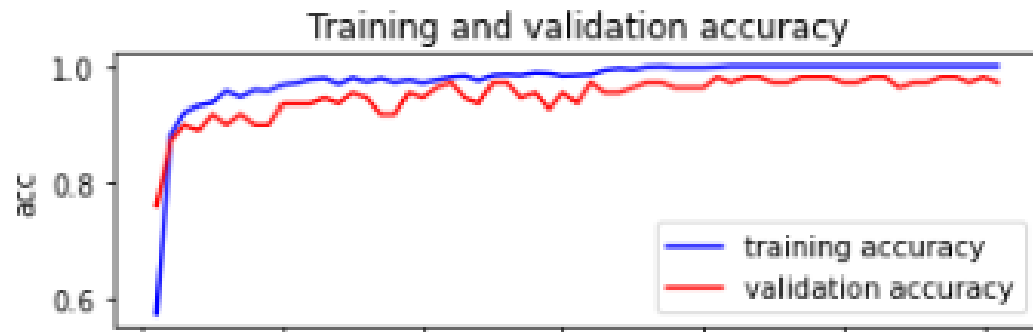
Epoch 61/200

22/31 [=====>.....] - ETA: 0s - loss: 0.0015 - accuracy: 1.0000

Epoch 61: val\_accuracy did not improve from 0.98148

31/31 [=====] - 0s 3ms/step - loss: 0.0019 - accuracy: 1.0000 - val\_loss: 0.1374 - val\_accuracy: 0.9722

Epoch 61: early stopping



### 3. Code

해결방안

## 고양이 데이터 분류 정확도 높이기

	변경 사항	시도한 방법
두번째 시도	데이터 추가	틀린 데이터를 훈련데이터에 추가

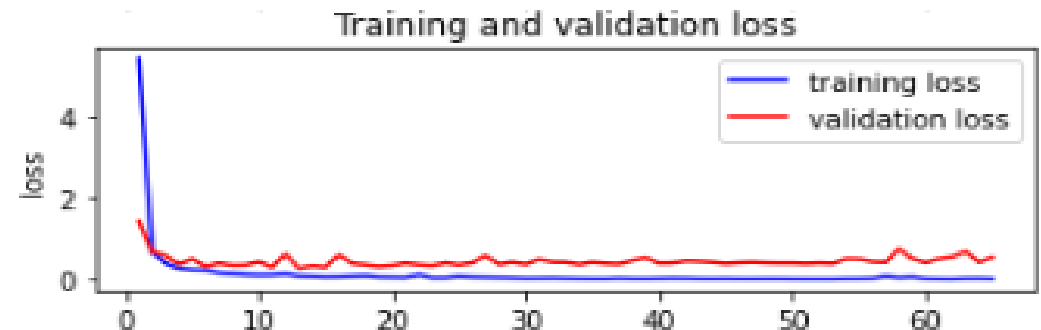
Epoch 65/200

24/28 [=====>.....] - ETA: 0s - loss: 0.0094 - accuracy: 0.9974

Epoch 65: val\_accuracy did not improve from 0.95392

28/28 [=====] - 0s 3ms/step - loss: 0.0097 - accuracy: 0.9965 - val\_loss: 0.5370 - val\_accuracy: 0.9401

Epoch 65: early stopping



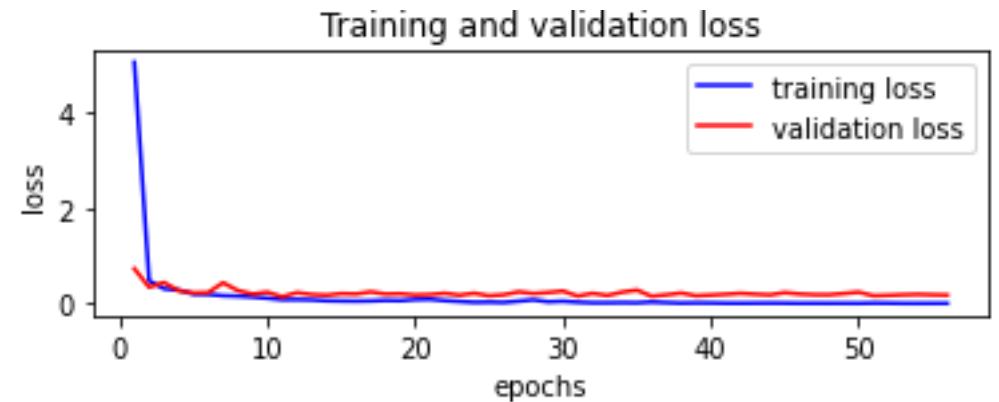
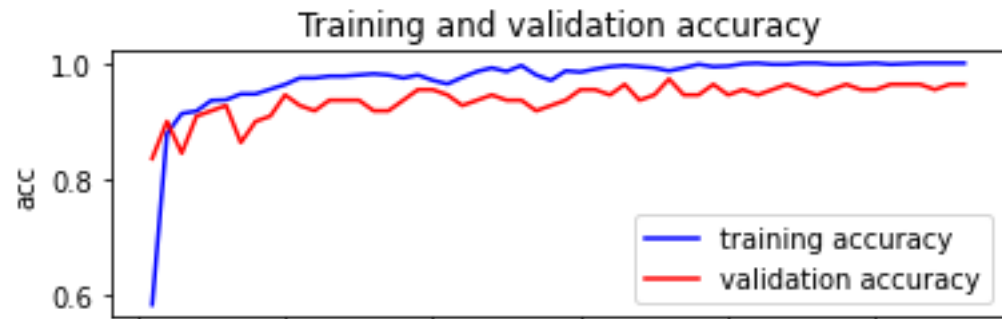
### 3. Code

해결방안

## 고양이 데이터 분류 정확도 높이기

check	변경 사항		시도한 방법
	세번째 시도	데이터 비율 + 데이터 추가	위 두가지 시도를 함께 시도 합니다. (9:1로 변경, 틀린데이터 4개 추가)

```
Epoch 56/200
25/31 [=====>.....] - ETA: 0s - loss: 0.0043 - accuracy: 1.0000
Epoch 56: val_accuracy did not improve from 0.97248
31/31 [=====] - 0s 3ms/step - loss: 0.0043 - accuracy: 1.0000 - val_loss: 0.1779 - val_accuracy: 0.9633
Epoch 56: early stopping
```



해결방안

## 고양이 데이터 분류 정확도 높이기

	변경 사항	내용
네번째 시도	GUI의 test데이터를 불러오는 코드 수정	고정적이었던 sample_path를 self.sr로 수정

```
# sample_rate 가져오기
self.sample_path = 'sample_rate.wav'
self.sample_rate = librosa.load(self.sample_path)[1]
```

# 샘플 파일을 집어넣으면 음성의 특징을 추출하는 함수(불러온 모델에 넣어 예측할 수 있도록 변형)

```
def extract_features(self):
```

```
    self.sample, self.sr = librosa.load(self.file_path)
```

```
    extracted_features = np.empty((0, 61, ))
```

```
    zero_cross_feat = librosa.feature.zero_crossing_rate(self.sample).mean()
```

```
    self.mfccs = librosa.feature.mfcc(y=self.sample, sr=self.sr, n_mfcc=60)
```

```
    mfccsscaled = np.mean(self.mfccs.T, axis=0)
```

```
    mfccsscaled = np.append(mfccsscaled, zero_cross_feat)
```

```
    mfccsscaled = mfccsscaled.reshape(1, 61, )
```

```
    self.test_sample = np.vstack((extracted_features, mfccsscaled))
```

# 불러온 오디오 파일의 진폭을 담기  
# 61개의 값을 받을 비어있는 리스트 생성

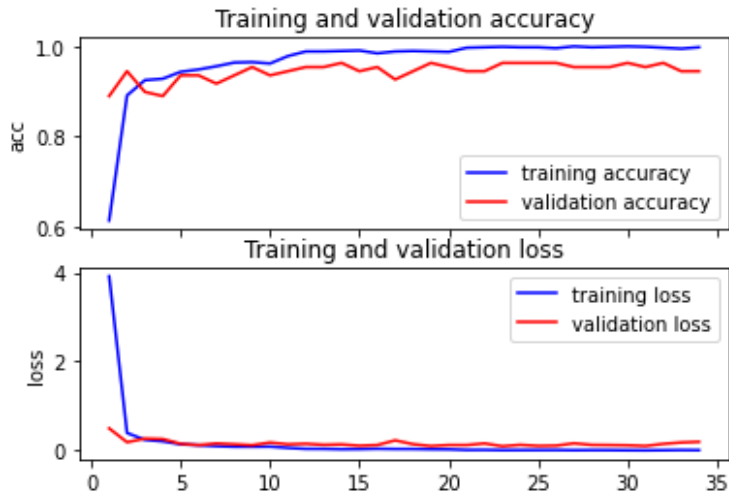
### 3. Code

추가

## PETPULS LAB 제공 강아지 소리 데이터(+노이즈)

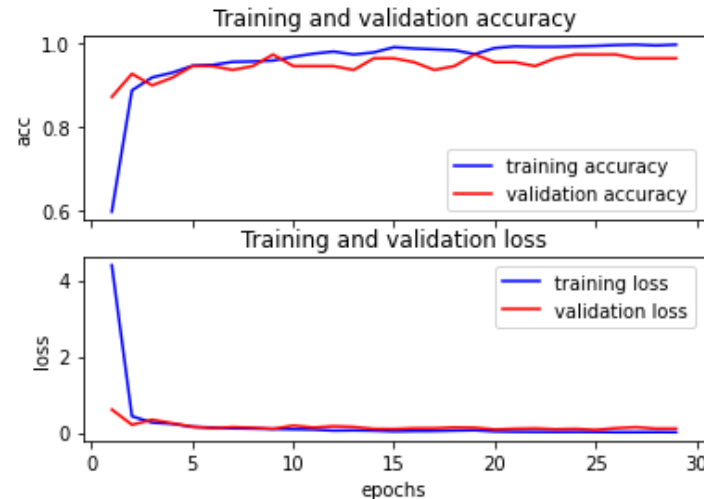
	변경 사항	시도한 방법
다섯번째 시도	리턴 될 mfcc 갯수 변경	<code>mfcc = librosa.feature.mfcc(audio, sr=16000, <b>n_mfcc= ?</b>, n_fft=400, hop_length=160)</code>

100



정확도 99.79%, 테스트정확도 :94.50%

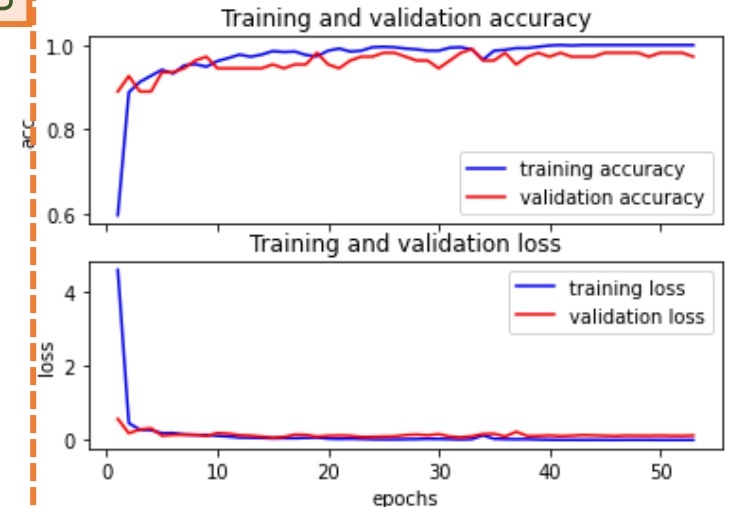
70



정확도 : 99.59%, 테스트 정확도 : 96.33%

check

60



정확도 100%, 테스트정확도 97.25%

### 3. Code

문제점  
해결!!!



고양이 데이터 '만' 오분류 하는 상황 해결 0

+ 강아지와 새 소리도 97.25 %정확도로 분류 0



# 4. GUI

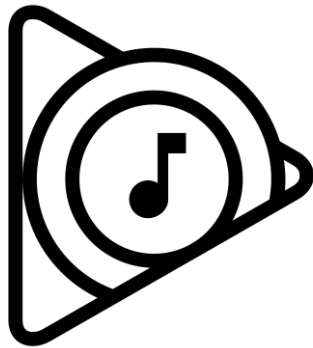
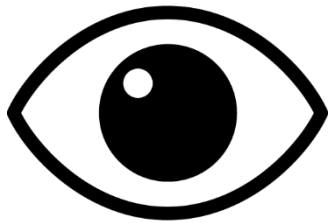


## 4. GUI



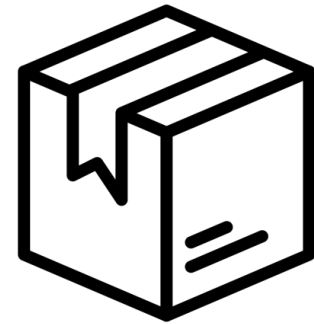
Check  
Point

Waveform  
mel-spectrogram  
그래프를 통한 시각화



Play / pause / stop  
버튼으로 구현

패키지를 통한 편의성 확보  
(파일명: SoundGUI)



99.215

정확한 정답 비율 표시

## 4. GUI

음성 파일 불러오기



예측 결과 비율 출력



그래프 시각화

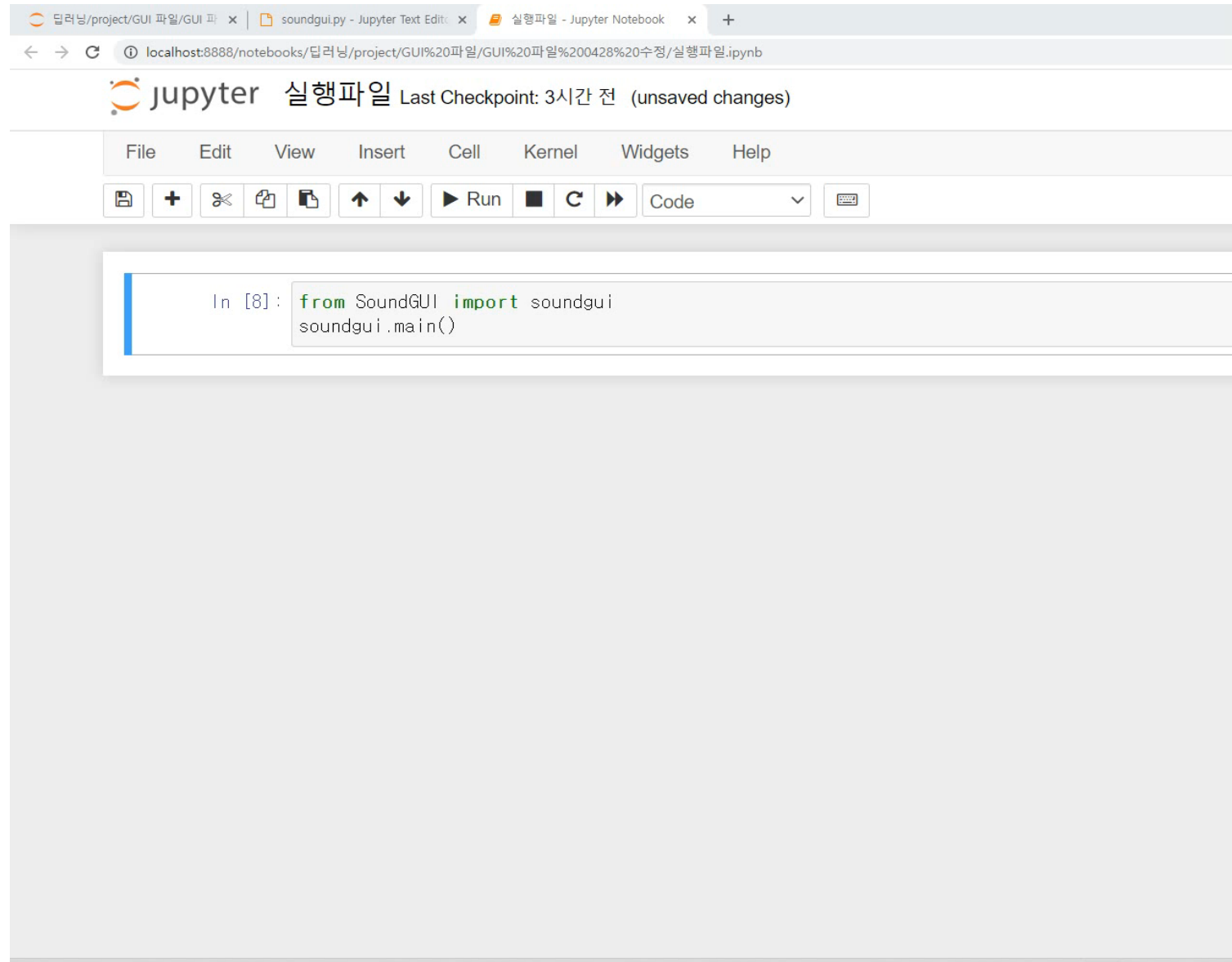


그래프 저장



## 4. GUI

시연 영상



### • Load\_audio()

```
def load_audio(self):  
    try:  
        pygame.mixer.music.load(self.file_path)  
    except:  
        data, samplerate = soundfile.read(self.file_path)  
        # wav 파일을 읽을 수 있도록 변환  
        soundfile.write(self.file_path, data, samplerate,  
                        subtype='PCM_16')  
        pygame.mixer.music.load(self.file_path)
```

### • Destroy()

```
def destroy(self):  
    try:  
        self.label.destroy() # 모델 예측 결과 출력 초기화  
        self.a1.clear()      # 그래프 초기화  
        self.a2.clear()  
        self.a3.clear()  
        self.spec_colorbar.remove() # colorbar 초기화  
        self.mfccs_colorbar.remove()  
        # 없을 시 새로운 file_path 경로가 들어왔을 때 그래프 초기화가 되지 않는다.  
        self.canvas.draw_idle()  
    except:  
        pass
```

### • PLAY ()

```
def play(self):
```

```
    pygame.mixer.music.play() # 로드한 오디오 파일 재생
```

```
    # file_path에 그려지는 그래프와 모델 분류는 한번만 하고 유지되도록 함
```

```
    if self.cnt_play_event:
```

```
        try:
```

```
            self.classify()    # classify 함수 호출
```

```
            self.set_plot()    # set_plot 함수 호출
```

```
            # 그래프 파일 저장 버튼
```

```
            self.save_button = Button(self.root, text='Save Graph', command=self.save_graph, width=15, height=1)
```

```
            self.save_button.configure(background='#E6DFDF', font=('Arial',12,'bold'))
```

```
            self.save_button.place(x=770, y=930)
```

```
        except:
```

```
            pass
```

```
        finally: # play를 누를 때마다 그래프와 예측 결과가 텍스트가 새로 나타나지 않도록 함(겹치지 않게)
```

```
            self.cnt_play_event = False
```

## 4. GUI

### • Self\_plot()

#### 1. Waveform 그래프

가로축 : 시간

세로축 : Amplitude = 진폭

↓ 푸리에 변환 ; 음의 세기인 magnitude를 dB로 변환하여 색으로 표현

#### 2. Spectrogram 그래프

가로축 : 시간

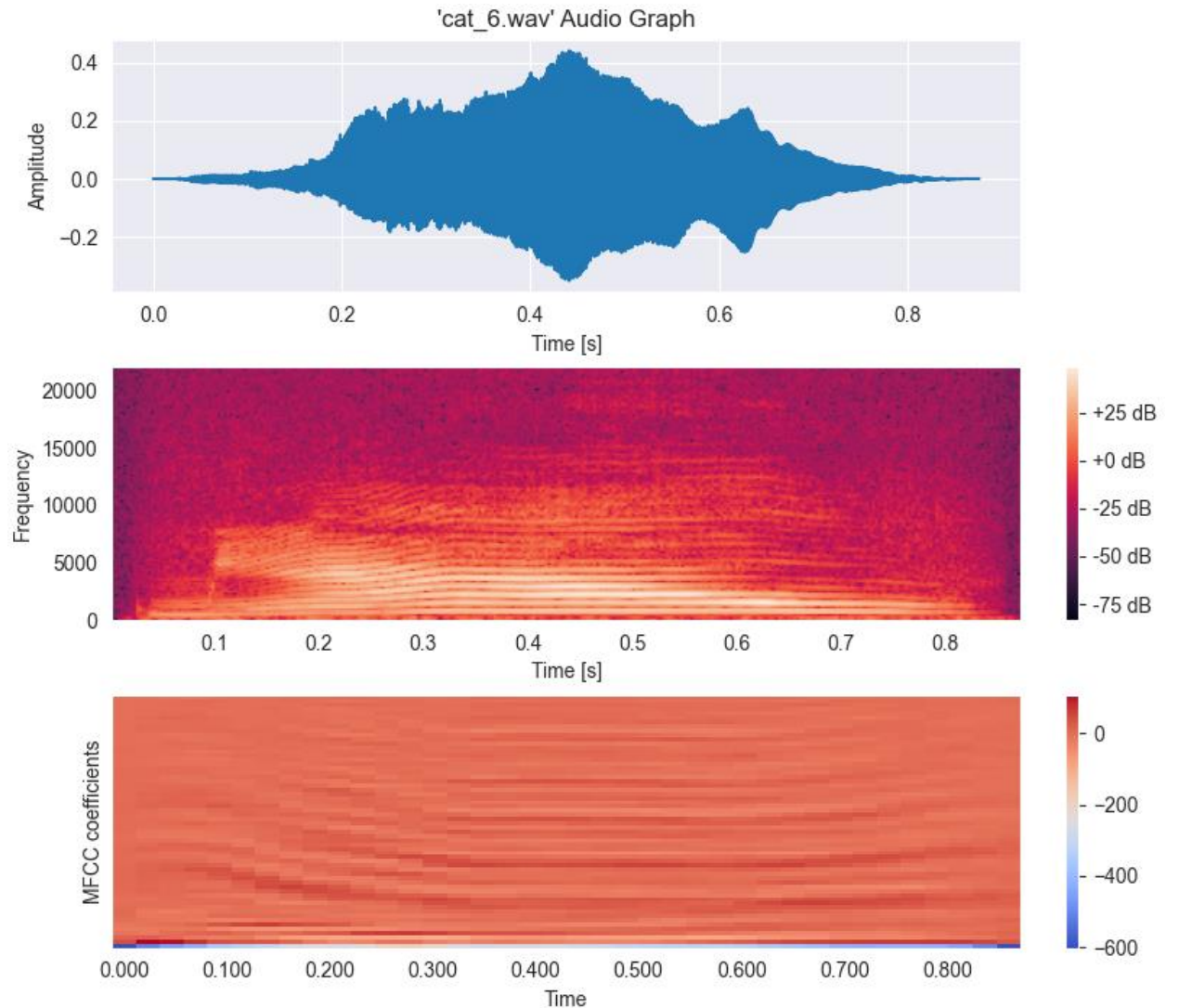
세로축 : Frequency = 주파수 (단위: Hz)

↓ mel-filter bank 통과+log scaling : 압축된 데이터로

#### 3. MFCCs 그래프

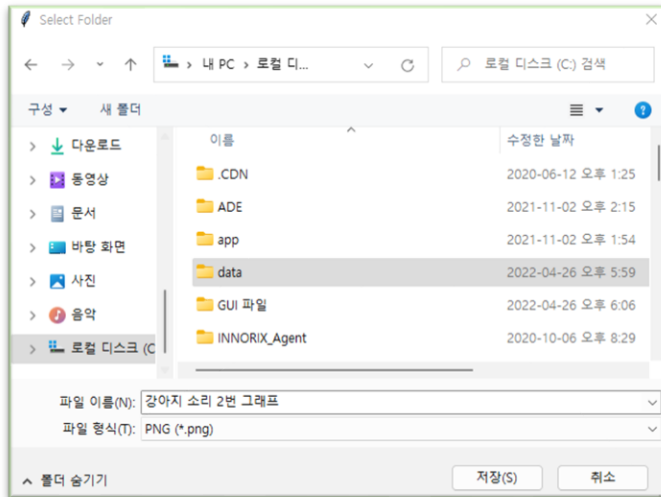
가로축 : 시간

세로축 : MFCC coefficients

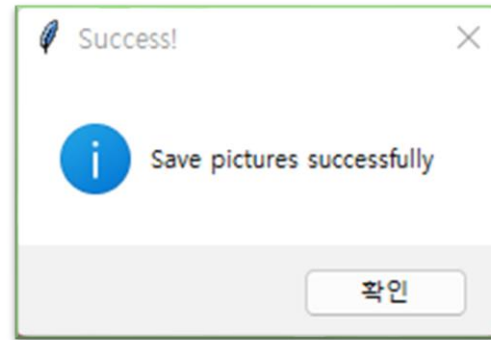


## 4. GUI

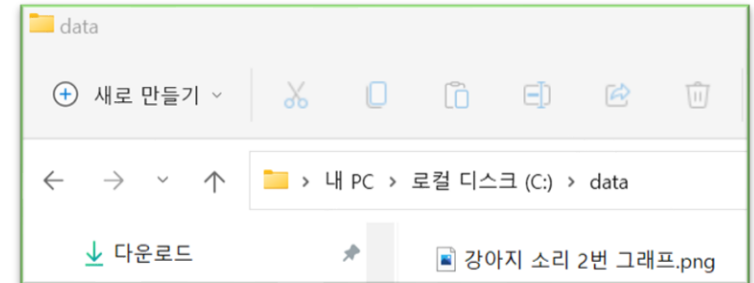
# Save Graph 버튼 구체적인 기능



파일이 지정 될  
경로 , 이름, 형식 설정 가능



파일이 성공적으로  
저장되었다는 팝업창 출력



폴더에서 성공적으로 저장이  
되었음을 확인할 수 있다.

## 4. GUI

### • save\_graph()

# 그래프를 저장하는 함수

```
def save_graph(self):
```

```
    try:
```

```
        files = [('PNG', '.png'), ('JPEG', '.jpg'), ("All Files", "*.*")] # 저장할 수 있는 파일 형식 목록
```

```
        save_file = filedialog.asksaveasfilename(initialdir="/",
```

```
                                                    title="Select Folder",
```

```
                                                    filetypes=files,
```

```
                                                    defaultextension='.png')
```

```
        if save_file:
```

```
            self.f.savefig(save_file)
```

```
            messagebox.showinfo("Success!", "Save pictures successfully")
```

```
            # 파일이 저장될 때만 알림창이 나타나도록 함
```

```
    except:
```

```
        pass
```





## 4. GUI

### • extract\_features()

# 샘플 파일을 집어넣으면 음성의 특징을 추출하는 함수  
(불러온 모델에 넣어 예측할 수 있도록 변형)

```
def extract_features(self):  
    self.sample, self.sr = librosa.load(self.file_path)  
    extracted_features = np.empty((0, 61, ))  
    zero_cross_feat = librosa.feature.zero_crossing_rate(self.sample).mean()  
    self.mfccs = librosa.feature.mfcc(y=self.sample, sr=self.sr, n_mfcc=60)  
    mfccsscaled = np.mean(self.mfccs.T, axis=0)  
    mfccsscaled = np.append(mfccsscaled, zero_cross_feat)  
    mfccsscaled = mfccsscaled.reshape(1, 61, )  
    self.test_sample = np.vstack((extracted_features, mfccsscaled))
```

### • Classify()

```
def classify(self):
```

```
    self.extract_features() # extract_features 함수 호출  
    pred = self.model.predict(self.test_sample.reshape(1, 61,))  
    answer = np.argmax(pred)  
    result_pred = str(pred[0][answer]*100)[:6]  
    if answer == 0:  
        self.sign='Bird : ' + result_pred + '%'  
    elif answer == 1:  
        self.sign='Cat : ' + result_pred + '%'  
    else:  
        self.sign='Dog : ' + result_pred + '%'
```

### 코드 실행 시 발생하는 warning : 그래프 관련



WavFileWarning: Chunk (non-data) not understood, skipping it. sf, sd = wavfile.read(self.file\_path)

non-data : 소리데이터가 아닌 부분(메타데이터에 대한 정보-저작권, 트랙이름)을 이해하지 못하는 것에 대한 경고

untimeWarning: divide by zero encountered in  $\log_{10} Z = 10. * \text{np.log}_{10}(\text{spec})$

소리가 없는 부분 0에 가까운 곳의 data에서 spectrogram으로 그려질 수 없는 것에 대한 문제



## 4. GUI

문제점

gui에서 stop을 넣지 않고 창을 닫을 경우, 소리 재생 문제

해결방안

Tkinter 창을 닫으면 실행되는 함수, 창을 닫으면 재생중인 음악이 정지하는 코드 구현

해결 코드:

```
def on_closing(self):  
    pygame.mixer.music.stop()  
    self.root.destroy()  
  
# tkinter 창을 닫으면 음악이 재생되지 않도록 함  
self.root.protocol("WM_DELETE_WINDOW", self.on_closing)
```



## 4. GUI

문제점

pygame.mixer.music.load(path) 코드 입력시

Unknown WAVE format

원인:

"32-bit floating point WAV File"

pygame으로 32비트 부동 소수점 값을 포함하는 오디오 WAV 파일을 읽을 수 없다.

해결방안

해결 방법:

soundfile 모듈을 사용하여 **16비트 wav 파일로 변환**하여 pygame으로 로드 한다.

해결 코드:

```
def load_audio(self):
    try:
        pygame.mixer.music.load(self.file_path)
    except:
        data, samplerate = soundfile.read(self.file_path)
        soundfile.write(self.file_path, data, samplerate, subtype='PCM_16') # wav 파일을 읽을 수 있도록 변환
        pygame.mixer.music.load(self.file_path)
```





## 데이터 전처리

- ✓ 3종류의 동물 데이터를 수집
- ✓ 수집 데이터를 꼼꼼하게 전처리

## 분류 모델 구현 (코드)

- ✓ 하이퍼 파라미터 조절을 통해 정확도 97% 도달

**GUI**

- ✓ 소리 재생 시 한번에 3가지의 그래프가 그려지도록 구현

## 출처\_1

- PPT 템플릿:

<http://pptbizcam.co.kr/?p=7677>

- 사진 및 영상:

<https://www.youtube.com/watch?v=Mc9PHZ3H36M&list=WL&index=7>

<https://www.pngwing.com>

<https://www.clipartkorea.co.kr/>

- 편집(gif):

<https://gifrun.com/youtube/?v=Mc9PHZ3H36M>

- 이론:

세계일보 & Segye.com - <https://www.segye.com/newsView/20070322001256>

- 데이터 출처:
  - 강아지 : <https://www.kaggle.com/datasets/mmoreaux/audio-cats-and-dogs>  
<https://bigsoundbank.com/detail-0916-barking-dog.html>  
<https://orangefreesounds.com/sound-effects/animal-sounds/dog-sounds/page/2/>  
<https://soundbible.com/tags-dog-bark.html>
  - 고양이 : <https://www.kaggle.com/datasets/andrewmvd/cat-meow-classification>  
<https://www.kaggle.com/datasets/mmoreaux/audio-cats-and-dogs>
  - 새 : <https://www.kaggle.com/datasets/ttahara/birdsong-resampled-train-audio-00?select=aldfly>  
(32종 이름 출처 : <https://birdsoftheworld.org/bow/home>)

*Thank you!*

