데이터 가공 프로젝트 pt1

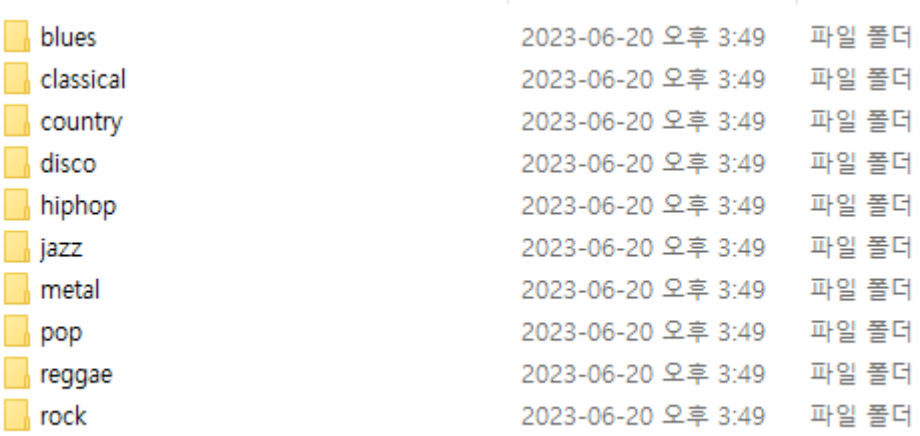
GTZAN Dataset image 프로세싱

전진희

프로젝트 개요:

음성 데이터에서 MelSepctrogram, STFT, waveshow 이미지를 추출하고, Augmentation 기법을 적용하여 데이터를 증강하는 작업을 수행한다. 또한, 원본 이미지와 Augmentation된 이미지를 비율을 유지하면서 255x255 사이즈로 리사이즈한다.

사용 데이터는 총 10가지 종류의 음악으로 구성되어 있으며, 각 음악 종류당 100가지의 데이터 파일로 구성되어 있다. 단, "jazz.00054" 파일은 음원이 손상되어 "jazz.00053"파일을 하나 더 복사하여 "jazz.00054"를 대체하였다. (jazz.00053와 jazz.00054는 같은 파일이다.)



사용할 원본 데이터

WaveShow 이미지 추출

조건1. 음성 데이터의 0초~10초 구간 WaveShow 변환 이미지 추출

```
# 함수 정의

def waveShowOrigin(audio_path, output_path):
    start_time = 0
```

```
    end_time = 10

    data, sr = librosa.load(audio_path, sr=22050)
    start_sample = int(sr * start_time)
    end_sample = int(sr * end_time)
    data_section = data[start_sample:end_sample]

    plt.figure(figsize=(12, 4))
    librosa.display.waveshow(data_section, color='red')
    plt.axis('off')

    os.makedirs(os.path.dirname(output_path), exist_ok=True)
    plt.savefig(output_path, bbox_inches='tight', pad_inches=0)
    plt.close()



# 반복문 사용
audio_folder = './raw_data/classical/'
output_folder = './waveshow/classical_original/'

for i in range(100):
    audio_path = os.path.join(audio_folder, f'classical.{str(i).zfill(5)}.wav')
    output_path = os.path.join(output_folder, f'waveshow_classical.{str(i).zfill(5)}_original.png')

    waveShowOrigin(audio_path, output_path)
```

조건2. WaveShow에 대해 augmentation 진행 : noise

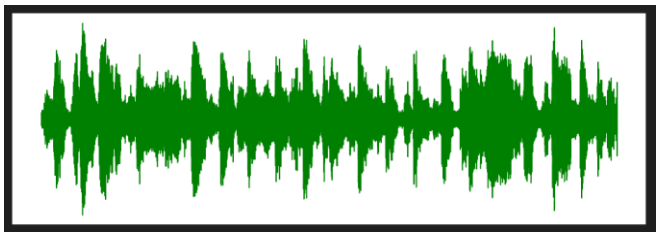
```
def waveShowNoise(audio_path, output_path):
    start_time = 0
    end_time = 10

    data, sr = librosa.load(audio_path, sr=22050)
    start_sample = int(sr * start_time)
    end_sample = int(sr * end_time)
    data_section = data[start_sample:end_sample]

    os.makedirs(os.path.dirname(output_path), exist_ok=True)
    plt.savefig(output_path, bbox_inches='tight', pad_inches=0)
```

```
    plt.close()

    # 노이즈 추가
    noise = 0.08 * np.random.rand(*data_section.shape)
    data_noise = data_section + noise

    # 노이즈 추가된 그래프
    plt.figure(figsize=(12, 4))
    librosa.display.waveshow(data_noise, color='red')
    plt.axis('off')

    # 결과 이미지 저장
    plt.axis('off')
    plt.savefig(output_path, bbox_inches='tight', pad_inches=0)
    plt.close()


# 사용
audio_folder = './raw_data/classical/'
output_folder = './waveshow/classical_noise/'

for i in range(100):
    audio_path = os.path.join(audio_folder, f'classical.{str(i).zfill(5)}.wav')
    output_path = os.path.join(output_folder, f'waveshow_classical.{str(i).zfill(5)}_noise.png')

    waveShowNoise(audio_path, output_path)
```

조건2. WaveShow에 대해 augmentation 진행 : stretch

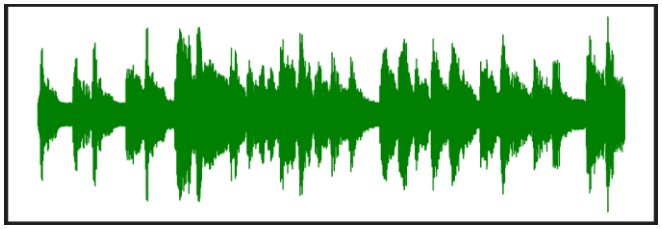
```
def waveShowNoise(audio_path, output_path):
    start_time = 0
    end_time = 10

    data, sr = librosa.load(audio_path, sr=22050)
    start_sample = int(sr * start_time)
    end_sample = int(sr * end_time)
    data_section = data[start_sample:end_sample]
```

```
        os.makedirs(os.path.dirname(output_path), exist_ok=True)
        plt.savefig(output_path, bbox_inches='tight', pad_inches=0)
        plt.close()

        # 스트레치 저장
        data_stretch = librosa.effects.time_stretch(data_section, rate=0.8)

        # 이미지 저장
        plt.figure(figsize=(12, 4))
        librosa.display.waveshow(data_stretch, color='red')
        plt.axis('off')
        plt.savefig(output_path, bbox_inches='tight', pad_inches=0)
        plt.close()



# 사용
audio_folder = './raw_data/classical/'
output_folder = './waveshow/classical_stretch/'

for i in range(100):
        audio_path = os.path.join(audio_folder, f'classical.{str(i).zfill(5)}.wav')
        output_path = os.path.join(output_folder, f'waveshow_classical.{str(i).zfill(5)}_stretch.png')

        waveShowNoise(audio_path, output_path)
```



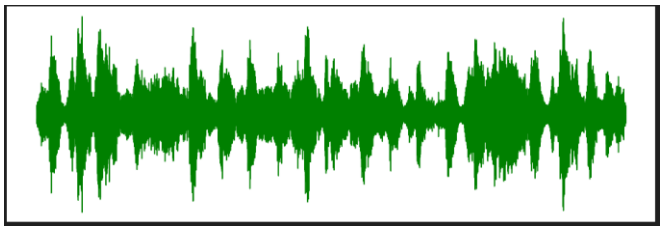
오리지날 데이터 예시

Waveshow_blues.00000_original.png



Noise로 증강한 데이터 예시

Waveshow_blues.00000_noise.png



Stretch로 증강한 데이터 예시

Waveshow_blues.00000_noise.png

STFT 이미지 추출

조건1. 음성 데이터의 0초~10초 구간 STFT 변환 이미지 추출

```
import matplotlib.pyplot as plt

import librosa
import librosa.display

import os
import glob
import numpy as np
import IPython
import random
```

```
def STFTImages(audio_folder, output_folder):
    os.makedirs(output_folder, exist_ok=True)

    for i in range(100):
        audio_path = os.path.join(audio_folder, f'classical.{str(i).zfill(5)}.wav')



        output_path = os.path.join(output_folder, f'stft_classical.{str(i).zfill(5)}_original.png')

        data, sr = librosa.load(audio_path, sr=22050)
        start_time = 0
        end_time = 10
        start_sample = int(sr * start_time)
        end_sample = int(sr * end_time)
        data_section_stft = data[start_sample : end_sample]

        # STFT 계산
        stft_temp = librosa.stft(data_section_stft)

        # STFT => dB 변환
```

```
            stft_db_temp = librosa.amplitude_to_db(abs(stft_temp))

            # 이미지 저장
            plt.figure(figsize=(12, 4))
            librosa.display.specshow(stft_db_temp, sr=sr, x_axis='time', y_axis='hz')
            plt.axis('off')
            plt.savefig(output_path, bbox_inches='tight', pad_inches=0)
            plt.close()


audio_folder = './raw_data/classical/'
output_folder = './STFT/classical_original/'

STFTImages(audio_folder, output_folder)
```

조건2. STFT에 대해 augmentation 진행 : noise

```
def addNoise(audio_folder, output_folder):
    os.makedirs(output_folder, exist_ok=True)

    for i in range(100):
        audio_path = os.path.join(audio_folder, f'blues.{str(i).zfill(5)}.wav')
        output_path = os.path.join(output_folder, f'stft_blues.{str(i).zfill(5)}_noise.png')

        data, sr = librosa.load(audio_path, sr=22050)
        start_time = 0
        end_time = 10
        start_sample = int(sr * start_time)
        end_sample = int(sr * end_time)
        data_section_stft = data[start_sample : end_sample]

        # 노이즈 추가
        noise = 0.005 * np.random.randn(*data_section_stft.shape)
        augmented_data_section = data_section_stft + noise

        # STFT 계산
        augmented_stft = librosa.stft(augmented_data_section)

        # STFT 결과 -> dB로 변환
        augmented_stft_db = librosa.amplitude_to_db(abs(augmented_stft))
```

```
        # 이미지 저장
        plt.figure(figsize=(12, 4))
        librosa.display.specshow(augmented_stft_db, sr=sr, x_axis='time', y_axis='hz')
        plt.axis('off')
        plt.savefig(output_path, bbox_inches='tight', pad_inches=0)
        plt.close()


audio_folder = './raw_data/blues/'
output_folder = './STFT/blues_noise/'

addNoise(audio_folder, output_folder)
```

조건3. STFT에 대해 augmentation 진행 : stretch

```
def addStretch(audio_folder, output_folder):
    os.makedirs(output_folder, exist_ok=True)

    for i in range(100):
        audio_path = os.path.join(audio_folder, f'country.{str(i).zfill(5)}.wav')
        output_path = os.path.join(output_folder, f'stft_country.{str(i).zfill(5)}_stretch.png')

        data, sr = librosa.load(audio_path, sr=22050)
        start_time = 0
        end_time = 10
        start_sample = int(sr * start_time)
        end_sample = int(sr * end_time)
        data_section_stft = data[start_sample : end_sample]

        # Stretching 기법 적용
        rate = 0.8 + np.random.random() * 0.4    # 0.8 ~ 1.2 사이의 랜덤한 비율로 time
stretching
        stretched_data_section = librosa.effects.time_stretch(data_section_stft, rate=rate)

        # STFT 계산
        stretched_stft = librosa.stft(stretched_data_section)

        # STFT 결과 -> dB로 변환
        stretched_stft_db = librosa.amplitude_to_db(abs(stretched_stft))
```

```
        # 이미지 저장
        plt.figure(figsize=(12, 4))
        librosa.display.specshow(stretched_stft_db, sr=sr, x_axis='time', y_axis='hz')
        plt.axis('off')
        plt.savefig(output_path, bbox_inches='tight', pad_inches=0)
        plt.close()


audio_folder = './raw_data/country/'
output_folder = './STFT/country_stretch/'

addStretch(audio_folder, output_folder)
```

MelSepctrogram 이미지 추출

조건1. 음성 데이터의 0초~10초 구간 MelSepectrogram 이미지 추출

```
import matplotlib.pyplot as plt

import librosa
import librosa.display

import os
import glob
import numpy as np
import IPython
import random
```

```
def MELImages(audio_folder, output_folder):
    os.makedirs(output_folder, exist_ok=True)

    for i in range(100):
        audio_path = os.path.join(audio_folder, f'blues.{str(i).zfill(5)}.wav')
        output_path = os.path.join(output_folder, f'mel_blues.{str(i).zfill(5)}_original.png')

        data, sr = librosa.load(audio_path, sr=22050)
        start_time = 0
```

```
            end_time = 10
            start_sample = int(sr * start_time)
            end_sample = int(sr * end_time)
            data_section_stft = data[start_sample : end_sample]

            # STFT 계산
            stft_temp = librosa.stft(data_section_stft)

            # 멜 스펙트로그램 계산
            mel_spec = librosa.feature.melspectrogram(S=abs(stft_temp))

            # dB 스케일로 변환
            mel_spec_db = librosa.amplitude_to_db(mel_spec, ref=np.max)

            # 이미지 저장
            plt.figure(figsize=(12, 4))
            librosa.display.specshow(mel_spec_db, sr=sr, x_axis='time', y_axis='mel')
            plt.axis('off')
            plt.savefig(output_path, bbox_inches='tight', pad_inches=0)
            plt.close()


audio_folder = './raw_data/blues/'
output_folder = './MelSpectrogram/blues_original/'

MELImages(audio_folder, output_folder)
```

조건2. MelSepectrogram 이미지에 대해 augmentation 진행 : noise

```
def addNoise(audio_folder, output_folder):
    os.makedirs(output_folder, exist_ok=True)

    for i in range(100):
        audio_path = os.path.join(audio_folder, f'blues.{str(i).zfill(5)}.wav')
        output_path = os.path.join(output_folder, f'stft_blues.{str(i).zfill(5)}_noise.png')

        data, sr = librosa.load(audio_path, sr=22050)
        start_time = 0
        end_time = 10
        start_sample = int(sr * start_time)
```

```
            end_sample = int(sr * end_time)
            data_section_stft = data[start_sample : end_sample]

            # 0초~10초 구간 STFT 계산
            stft_temp = librosa.stft(data_section_stft)

            # 멜 스펙트로그램 계산
            mel_spec = librosa.feature.melspectrogram(S=abs(stft_temp))

            # dB 스케일로 변환
            mel_spec_db = librosa.amplitude_to_db(mel_spec, ref=np.max)

            # 노이즈 추가
            noise = 0.005 * np.random.randn(*mel_spec_db.shape)
            augmented_spec = mel_spec_db + noise

            # db 스케일로 변환
            augmented_spec_db = librosa.amplitude_to_db(augmented_spec, ref=np.max)

            # 이미지 저장
            plt.figure(figsize=(12, 4))
            librosa.display.specshow(augmented_spec_db, sr=sr, x_axis='time', y_axis='hz')
            plt.axis('off')
            plt.savefig(output_path, bbox_inches='tight', pad_inches=0)
            plt.close()


audio_folder = './raw_data/blues/'
output_folder = './MelSpectrogram/blues_noise/'

addNoise(audio_folder, output_folder)
```

조건3. MelSepectrogram 이미지에 대해 augmentation 진행 : stretch

```
def addStretch(audio_folder, output_folder):
    os.makedirs(output_folder, exist_ok=True)

    for i in range(100):
        audio_path = os.path.join(audio_folder, f'country.{str(i).zfill(5)}.wav')
        output_path = os.path.join(output_folder, f'mel_country.{str(i).zfill(5)}_stretch.png')
```

```
        data, sr = librosa.load(audio_path, sr=22050)
        start_time = 0
        end_time = 10
        start_sample = int(sr * start_time)
        end_sample = int(sr * end_time)
        data_section_stft = data[start_sample : end_sample]



        # Stretching 기법 적용
        rate = np.random.uniform(low=0.8, high=1.2)    # 0.8 ~ 1.2 사이의 랜덤한 비율로 time stretching
        stretched = librosa.effects.time_stretch(data_section_stft, rate=rate)



        # 0초~10초 구간 STFT 계산
        stft_stretched = librosa.stft(stretched)

        # 멜 스펙트로그램 계산
        mel_spec_stretched = librosa.feature.melspectrogram(S=abs(stft_stretched))

        # dB 스케일로 변환
        mel_spec_stretched_db = librosa.amplitude_to_db(mel_spec_stretched, ref=np.max)

        # 이미지 저장
        plt.figure(figsize=(12, 4))
        librosa.display.specshow(mel_spec_stretched_db, sr=sr, x_axis='time', y_axis='hz')
        plt.axis('off')
        plt.savefig(output_path, bbox_inches='tight', pad_inches=0)
        plt.close()


audio_folder = './raw_data/country/'
output_folder = './MelSpectrogram/country_stretch/'

addStretch(audio_folder, output_folder)
```

리사이즈 진행

```
from PIL import Image
import os
import glob
```

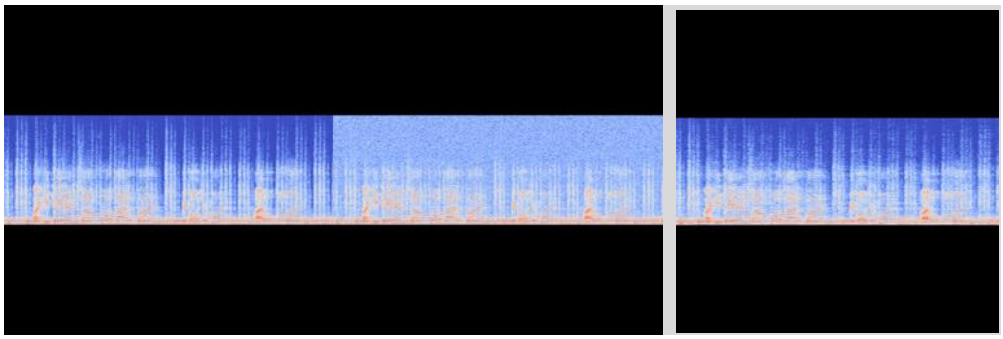
```
# 이미지를 정사각형으로 확장하는 함수
def expand_to_square(pil_img, background_color):
    width, height = pil_img.size

    if width == height:  # 이미지가 정사각형이면 그대로 반환
        return pil_img

    elif width > height:  # 가로가 더 긴 경우
        result = Image.new(pil_img.mode, (width, width), background_color)
        result.paste(pil_img, (0, (width - height) // 2))
        return result

    else:  # 세로가 더 긴 경우
        result = Image.new(pil_img.mode, (height, height), background_color)
        result.paste(pil_img, ((height - width) // 2, 0))
        return result


# 이미지를 주어진 크기로 리사이즈하고 패딩을 추가하는 함수
def resize_with_padding(pil_img, new_size, background_color):
    img = expand_to_square(pil_img, background_color)
    img = img.resize(new_size, Image.ANTIALIAS)
    return img


img_path_list = glob.glob(os.path.join("./waveshow/blues_original/", "*.png"))


# 결과 이미지를 저장할 폴더 생성
os.makedirs("./waveshow_resize/blues/", exist_ok=True)

for i, image_path in enumerate(img_path_list):
    print(image_path)
    img = Image.open(image_path)
```

```
    img_new = resize_with_padding(img, (255, 255), (0, 0, 0))

    # 결과 이미지를 파일로 저장
    save_file_name = f"./waveshow_resize/blues/{str(i).zfill(4)}_resize_blues_original.png"
    img_new.save(save_file_name, "png")
```

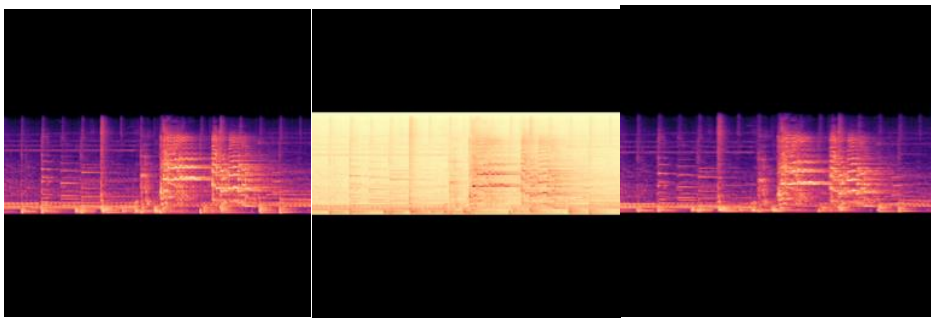
사용시 변수 재설정 필요

결과 예제:

Classical.00008의 waveshow이미지, noise 추가본, stretch 추가본



Country.00016의 stft, noise 추가본, stretch 추가본



Blues.00099의 MelSpectrogram, noise추가본, stretch 추가본

본 프로젝트를 통해 원본 음성 데이터에서 추출한 MelSepctrogram, STFT, waveshow 이미지를 다양한 Augmentation 기법을 적용하여 데이터를 확장하고, 비율을 유지한 리사이징을 수행. 이를 통해 보다 다양한 학습 데이터를 구성하고, 음성 데이터의 특징을 시각적으로 확인할 수 있었다.

의문점:

Stft와 melspectrogram이미지 작업에서 기본 이미지와 stretch 이미지의 차이가 너무 미세해서 뭔가 잘못된 건지 확인 필요