

졸업작품

최종보고서

-Sound Style Transfer Program based on AI-

스마트ICT융합공학과

201912335 이홍석

목차

제 1장 개요.....	3
1.1 문제 인식.....	3
1.1.1 목적.....	3
1.1.2 기대효과.....	3
1.1.3 정보 출처.....	4
제 2장 서비스 소개	5
2.1 서비스 소개.....	5
2.2 사용 방법.....	5
제 3장 개발.....	7
3.1 개발 환경.....	7
3.2 관련 기술.....	7
3.3 시스템 구조.....	8
3.3.1 generator.....	8
3.3.2 discriminator.....	10
3.4 학습 데이터 설계.....	11
3.4.1 train data.....	11
3.4.2 Colab에서의 실제 구현.....	12
3.5 모듈 구현.....	13
3.5.1 클래스 설명.....	13
3.5.2 메인 코드 설명.....	16
제 4장 말은 부분.....	19

제 1장 개요

1.1 문제 인식

“음성 생성에서의 다양한 모델 연구 부족”

최근 생성 모델에 대한 관심이 늘어나면서, Midjourney, DALL.E 등 고성능의 이미지 생성 모델이 등장하고 있다. 특히, Style Transfer에 관한 연구 역시 그 진척도가 상당하여, 초창기 CNN을 활용한 모델부터 pix2pix, CycleGAN, StyleGAN에 이르기까지 다양한 모델이 시도되고 연구되고 있다. 반면, 음성에서의 Style Transfer는 아직 본격적인 연구가 시작된 지 그리 오래되지 않았다. 특히, 그마저도 대부분 사람의 음성을 학습시켜 tts에 적용하는 부분이 더 활발하다. 그러나, 악기 음색의 경우, 특정 단어나 문장을 라벨링하여 사용하기 어렵다는 점 때문에, tts를 사용하기 어렵다. 이에, 악기 음색에 대한 Style Transfer를 이용하여, 하나의 곡에서 다양한 버전의 곡, 예를 들어 클래식 음악의 국악 버전, 밴드 음악의 어쿠스틱 버전 등을 생성해 내는 모델에 관심을 가지게 되었다.

1.1.1 목적

음원 데이터가 시계열 데이터이면서, mel-spectrogram은 이미지 형태로 표현할 수 있다는 점에 착안하여, 시계열 데이터의 예측에 사용되는 LSTM기반 seq2seq, Transformer모델과, 이미지 생성에 효율적으로 사용되는 GAN, CycleGAN모델을 사용하고, 높은 완성도를 보이는 모델이 있는지 확인하고자 하였다.

1.1.2 기대효과

Style Transfer를 활용하여 악기 소리에 대한 음성 생성이 가능하다면, 새로 발매된 음원의 음색을 국악 버전이나 오케스트라 버전으로 듣고 싶어하는

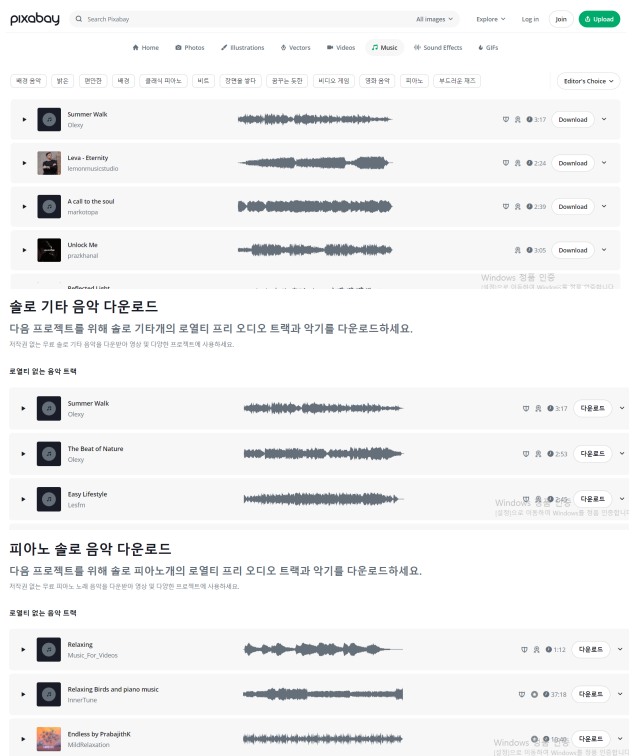
사람들의 니즈를 충족시킬 수 있고, 곡을 리메이크하는 작곡가들에게 있어서도 리메이크하고자 하는 곡의 악기를 선정하거나, 미리 시뮬레이션 해 보는 부분에 있어 도움이 될 것이라 생각한다.

1.1.3 정보 출처

1) 학습 데이터

저작권 프리 음원 다운로드 사이트에서 피아노, 솔로 기타 음원 발취

<https://pixabay.com/music/>



2) 베이스라인 모델

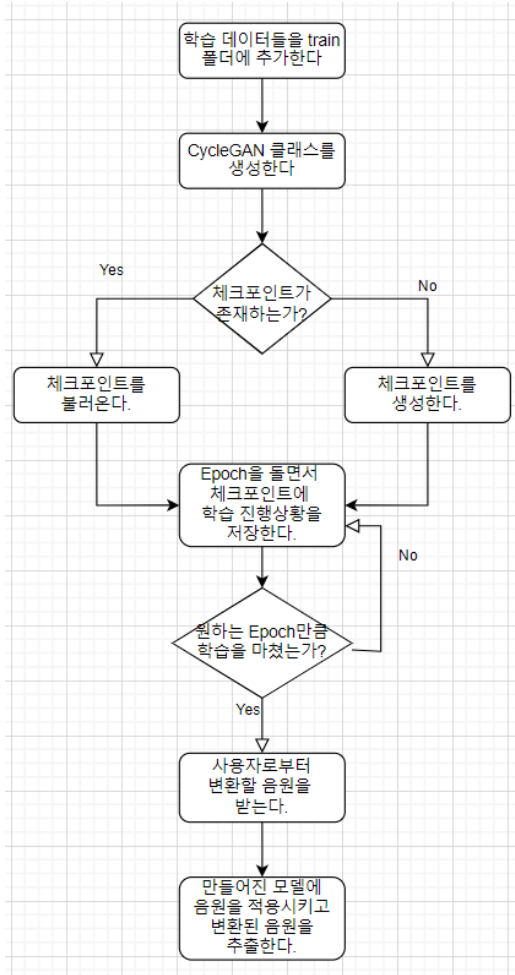
베이스라인 모델 코드의 경우, CycleGAN코드를 참고한 뒤, 음원 데이터에 적용하기 적합한 전처리 코드와 결합하여 사용하였다.

CycleGAN: <https://github.com/eriklindernoren/Keras-GAN/blob/master/cyclegan/cyclegan.py>

제 2장 서비스 소개

2.1 서비스 소개

CycleGAN 모델에 train data를 학습시키는 과정은 다음과 같다.



2.2 사용 방법

우선 학습시킬 데이터를 불러오는 과정을 거친다. 아래 사진과 같이 Google 드라이브에 저장해 놓은 데이터들을 불러 올 수 있도록 Path를 설정해 놓았다.

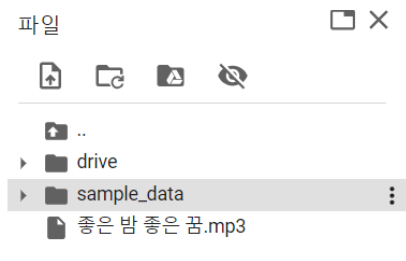
```
path = glob('/content/drive/MyDrive/data/datasets/Custom_data/%s/%s/*' % (self.dataset_name, data_type))
```

다음으로, 데이터를 학습 시키고 저장해놓은 체크포인트를 불러온다. 아래 사진과 같이 학습을 하고 드라이브에 저장해 놓았던 Checkpoint를 불러오는 과정을 거친다.

```
#체크포인트 로드
checkpoint_path = '/content/drive/MyDrive/data/datasets/cyclegan_checkpoint_0601.h5'
checkpoint = ModelCheckpoint(checkpoint_path, monitor='val_loss', verbose=1, save_best_only=False, mode='min')

if os.path.exists(checkpoint_path):
    self.combined.load_weights(checkpoint_path)
    print("Done")
```

그 다음, 변환하고자 하는 음성파일을 Colab환경에서 추가한다. 아래 사진을 보면 좋은 밤 좋은 꿈.mp3라는 음성파일을 추가한 다음 코드에서 해당 파일을 불러오는 작업을 거친다.



```
y, sr = librosa.load("/content/좋은 밤 좋은 꿈.mp3")

min_len = 288000
y = y[:min_len]

n_fft = 2048
n_hop = 512
n_mels = 512

# Mel spectrogram
melspec = librosa.feature.melspectrogram(y=y, sr=sr, n_fft=n_fft, hop_length=n_hop, n_mels=n_mels)
```

Piano에서 Guitar로 변환하고자 할 때는 `model.g_AB.predict(input)`를 사용하고, `out = model.g_AB.predict(input)`

Guitar에서 Piano로 변환하고자 할 때는 `model.g_BA.predict(input)`를 사용하여 변환한다.

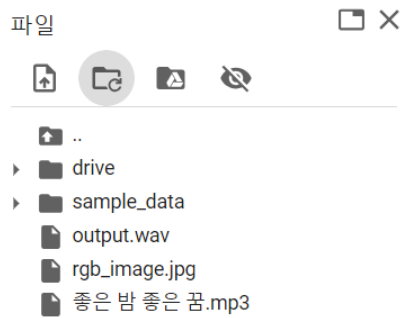
```
out = model.g_BA.predict(input)
```

마지막으로, 변환한 음성파일을 wav형태로 추출한다.

```
import soundfile as sf

# 생성한 오디오 파일(audio)을 WAV 파일로 저장
sf.write('/content/output.wav', audio, sr)
```

위의 과정이 끝나면 output.wav파일로 저장이 된다.



제 3장 개발

3.1 개발환경

Google Colab 환경을 사용하여 PC에서 테스트 및 빌드 진행

3.2 관련 기술

1) Google Colab

- 웹상 공유기능을 통해 팀원들과 공동작업 가능
- Google 서버의 GPU를 통해 원격으로 시간이 오래 걸리는 데이터 학습 진행 가능
- 다양한 라이브러리 사용 가능(librosa, keras, numpy, tensorflow)
- Google Drive 연동을 통한 학습 데이터 쉽게 불러오기

- 체크포인트 기능을 통한 작업상황 저장 가능으로 런타임 해제 문제점 극복

2) Keras

- Tensorflow의 레이어 기능을 보다 단순화시킨 형태
- Convolution, UpSampling Layer 등 기본적으로 제공하는 레이어가 다양하며, 상세한 구현 없이 레이어들을 조립하듯이 연결하여 모델을 구성할 수 있음

3) Librosa

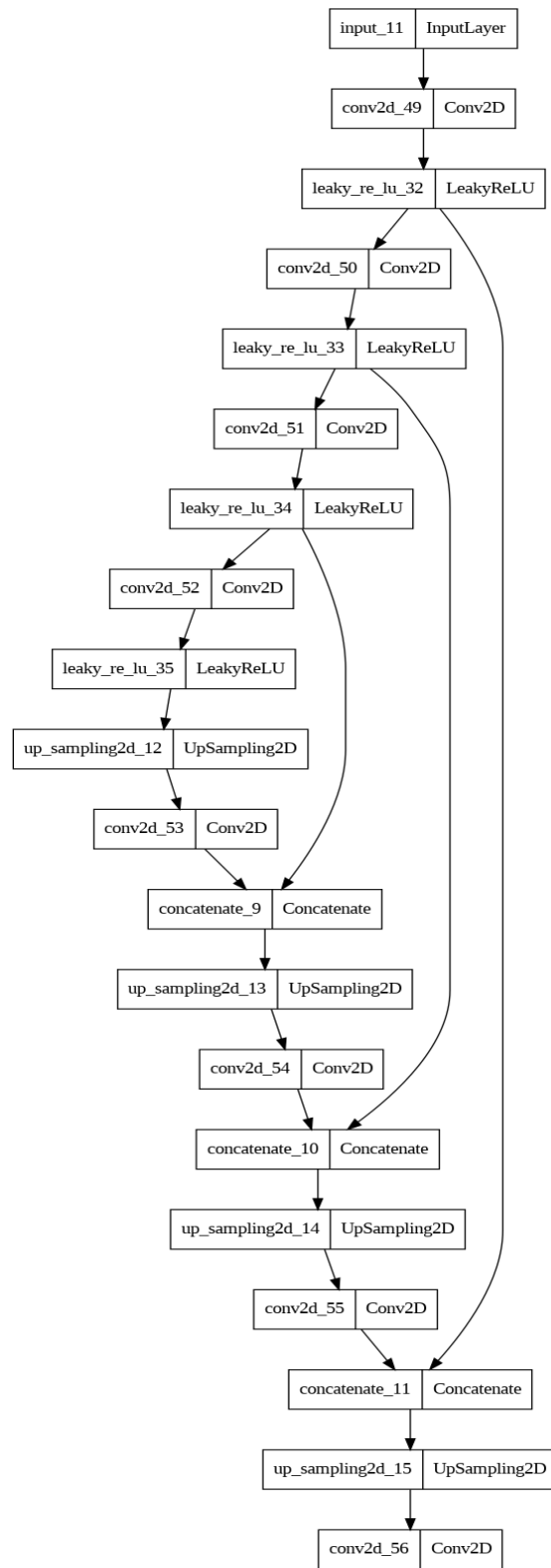
- 음성 데이터를 feature화하여 사용하는 데에 필요한 기능을 제공
- mel-spectrogram과 audio의 상호 전환이 가능하며, mfcc와 같이 주요 feature만 추출하는 기능도 제공

3.3 시스템 구조

시스템 구조는 기본적인 GAN의 구조를 따르며, 학습 데이터를 기반으로 새로운 데이터를 생성하는 generator와, generator가 생성한 데이터가 진짜 데이터와 유사한지를 가려내는 discriminator로 나뉜다.

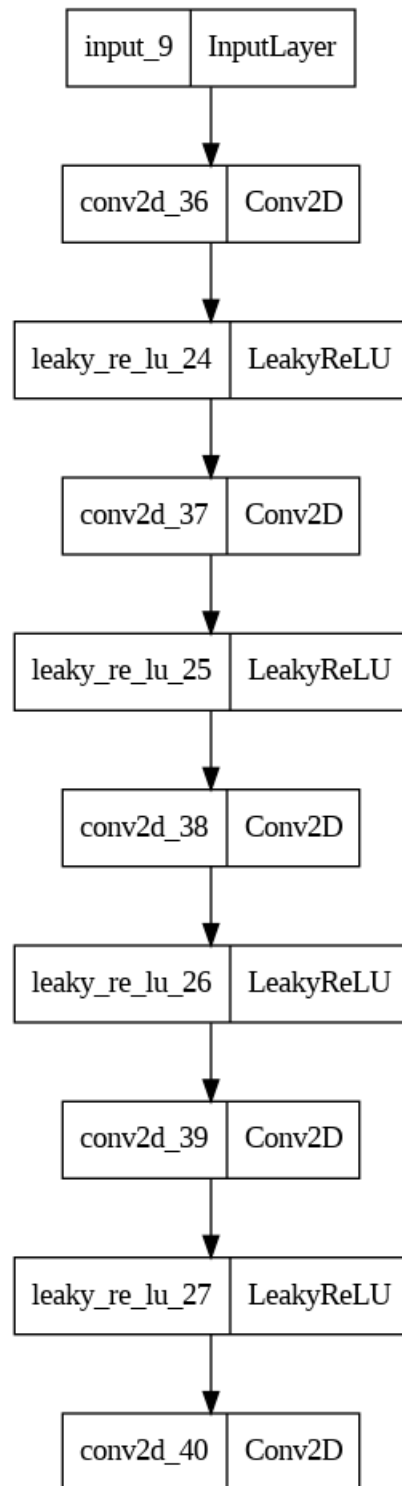
3.3.1 generator

학습 데이터에서 유의미한 특징을 추출하기 위해, Convolution Layer와 UpSampling Layer를 추가한 Encoder - Decoder형태를 추가하였다.



3.3.2 discriminator

Input Layer에는 generator가 생성한 음원 데이터가 들어가고, 이를 다중 Convolution Layer에 통과시킨 후, 최종적으로 생성된 데이터인지 원본 데이터인지 구분하여 결과를 바탕으로 generator를 훈련시킨다.



3.4 학습 데이터 설계

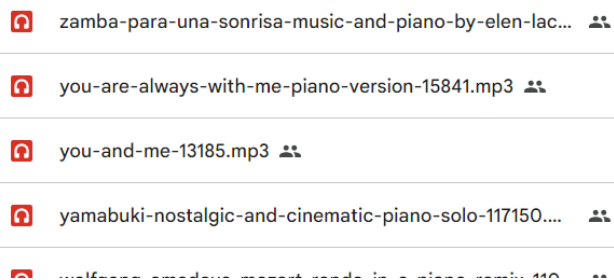
3.4.1 train data

CycleGAN의 훈련을 위한 데이터셋의 구조는 대략적으로 다음과 같이 설계하였다.

```
Custom_data/  
├─ trainA/  
│   ├─ 1.wav  
│   ├─ 2.wav  
│   └─ ...  
├─ trainB/  
│   ├─ 1.wav  
│   ├─ 2.wav  
│   └─ ...  
├─ testA/  
│   ├─ 1.wav  
│   ├─ 2.wav  
│   └─ ...  
└─ testB/  
    ├─ 1.wav  
    ├─ 2.wav  
    └─ ...
```











3.4.2 Colab에서의 실제 구현

Colab에서 Google Drive와 연동 기능을 제공하는 관계로, Google Drive에 데이터 폴더를 구축하고, 이를 불러와 Colab에서 train, test data로 사용하는 방식을 채택하였다. trainA, trainB 폴더를 생성하여 각각 피아노 음원들, 기타 음원들을 저장하였다. 피아노 음원은 877개, 기타 음원은 471개를 저장하였다.



The screenshot shows a list of audio files in Google Drive. Each entry consists of a red square icon with a white 'a' (representing an audio file), the filename, and a small icon of two people (representing sharing permissions). The files listed are:

- zamba-para-una-sonrisa-music-and-piano-by-elen-lac...
- you-are-always-with-me-piano-version-15841.mp3
- you-and-me-13185.mp3
- yamabuki-nostalgic-and-cinematic-piano-solo-117150....
- wolfman-awakens-moment-when-he-is-alone-again-410...

	youth-anna-li-sky-8479.mp3	
	your-love-song-114287.mp3	
	worship-together-20685.mp3	
	working-time-soft-acoustic-guitar-93818.mp3	
	words-unsaid-11540.mp3	

```
for k in range(0, (min(len(y_A), len(y_B))//min_len)):
    #imgs_A, imgs_B = [], []

    y_A_ex=y_A[k*min_len : (k+1)*min_len]
    y_B_ex=y_B[k*min_len : (k+1)*min_len]

    mel_A = librosa.feature.melspectrogram(y=y_A_ex, sr=sr_A, n_fft=2048, hop_length=512, n_mels=512)
    mel_B = librosa.feature.melspectrogram(y=y_B_ex, sr=sr_B, n_fft=2048, hop_length=512, n_mels=512)
```

위와 같은 음원 분할 과정을 거치게 되면서 학습 데이터가 약 3760개로 늘어나게 된다.

3.5 모듈 구현

전체 코드는 각각의 기능별로 구분된 클래스와, 해당 클래스의 객체들을 사용하여 구현된 메인 코드로 구성하였다.

3.5.1 클래스 설명

```
class DataLoader():
    def __init__(self, dataset_name, img_res=(512, 512)):
    def load_data(self, domain, batch_size=1, is_testing=False):
    def load_batch(self, batch_size=1, is_testing=False):
    def load_img(self, path):
    def imread(self, path):
```

데이터를 로드하는 DataLoader 클래스이다. 이 클래스는 이미지 데이터셋을 처리하기 위한 여러 메서드를 포함하고 있다.

`__init__` 메서드에서는 DataLoader 객체를 초기화하는데, 데이터셋의 이름(`dataset_name`)과 이미지 해상도(`img_res`)를 인자로 받는다.

`load_data` 메서드는 도메인(`domain`)과 배치 크기(`batch_size`)를 인자로 받아 데이터를 로드한다. `is_testing`은 테스트 데이터인지 여부를 나타내는 플래그이다. 주어진 도메인과 데이터 유형에 해당하는 파일 경로들을 가져온 후, 배치 크기만큼의 파일을 무작위로 선택한다.

선택된 파일들을 순회하면서 각 파일의 오디오 데이터를 로드하고, mel-spectrogram으로 변환한다. 변환된 mel-spectrogram을 이미지로 변환한 후, 정규화하여 반환한다. 이미지 학습 데이터인 경우에는 이미지를 크기 조정하거나 좌우 반전을 적용할 수 있으나, mel-spectrogram이 좌우 반전되거나 크기가 조정되어 학습에 사용되면 원래 얻고자 했던 형태와 다른 음색이 생성될 위험이 있어 좌우 반전이나 크기 조정을 통한 data augmentation은 배제하였다.

`load_batch` 메서드는 배치 크기와 테스트 여부를 인자로 받아 데이터 배치를 로드한다. 학습 데이터인 경우 "train" 폴더에서 A 도메인과 B 도메인의 파일 경로들을 가져와, 전체 샘플 수를 계산한 후, 경로에서 무작위로 배치 크기만큼의 파일을 선택한다.

선택된 파일들을 순회하면서 각 오디오 데이터를 로드하고, 최소 길이에 맞춰 여러 개의 작은 데이터로 나누어 처리한다. 그 후, 작은 데이터를 mel-spectrogram으로 변환하고 이미지로 변환한 후, 배치 리스트에 추가한다. 배치 크기만큼의 데이터가 모이면 해당 배치를 반환한다.

`load_img` 메서드는 주어진 경로에서 이미지를 로드하고, 크기를 조정하여 반환한다. 본 프로젝트에서는 전처리된 mel-spectrogram을 이미지 형태로

출력하여 올바르게 Style Transfer가 수행되고 있는지를 확인할 수 있게 시각화하는 데에 사용되었다.

`imread` 메서드는 이미지 파일을 읽어들이는 함수이다. 해당 코드에서는 `imread` 함수를 사용하여 이미지를 읽어들이고, 데이터 타입을 float로 변환하여 반환한다.

이 `DataLoader` 클래스를 사용하여 데이터셋을 로드하고 처리할 수 있다.

```
class CycleGAN():
    def __init__(self):
        def build_generator(self):
            def conv2d(layer_input, filters, f_size=4):
            def deconv2d(layer_input, skip_input, filters, f_size=4,
dropout_rate=0):
        def build_discriminator(self):
            def d_layer(layer_input, filters, f_size=4, normalization=False):
        def train(self, epochs, batch_size=1, sample_interval=50):
        def sample_images(self, epoch, batch_i):
```

CycleGAN 모델의 파이썬 클래스 구현이다. CycleGAN은 두 도메인 간의 이미지 변환을 수행하는 생성적 적대 신경망(GAN)으로, 두 개의 생성자(generator)와 두 개의 판별자(discriminator)로 구성되어 있다.

주요 메서드와 속성은 다음과 같다.

`__init__(self)` 메서드는 CycleGAN 클래스의 생성자 메서드로, 모델의 입력 크기, 데이터 로더, GAN 구성 등을 초기화한다.

`build_generator(self)`는 U-Net 형식의 생성자를 구성하는 메서드이다.

`build_discriminator(self)`는 판별자를 구성하는 메서드이다.

`train(self, epochs, batch_size, sample_interval)`: CycleGAN 모델을 훈련하는 메서드로, 주어진 에포크 수와 배치 크기에 따라 모델을 훈련하고, 중간 결과를 출력한다. 이 때, 출력 주기는 `sample_interval`로 한다.(즉, `sample_interval`이 10이라면, 10개의 배치에 대한 `train`을 수행할 때마다 test data에 대한 예측값을 산출하여 옳게 되었는지 확인할 수 있게 한다.)

`sample_images(self, epoch, batch_i)`는 주어진 에포크와 배치 인덱스에 대해 이미지 샘플을 생성하고 저장하는 메서드이다.

이 코드는 이미지 변환 작업을 위해 두 도메인 사이의 이미지를 생성하고 변환 결과를 확인하는 CycleGAN 모델을 구현하는 데 사용된다.

3.5.2 메인 코드 설명

1) 사용한 feature

사용된 feature들은 다음과 같다.

`y`는 input data를 리스트로 변환한 형태의 변수이다.

`melspec`는 일정 길이만큼 자른 input data로부터 mel-spectrogram 형태를 추출한 형태이다.

`input`은 2차원 형식의 melspectrogram에 채널 차원을 추가한 변수이다.

`out`은 input data를 기반으로 모델이 생성한 mel-spectrogram이다.

`audio`는 mel-spectrogram을 audio변환이 가능한 형태로 재변환한 데이터이다.

2) 소스 코드 및 해석

```
import cv2
```



```

import numpy as np

y, sr = librosa.load("/content/좋은 밤 좋은 꿈.mp3")

min_len = 262000
y = y[:min_len]

n_fft = 2048
n_hop = 512
n_mels = 512

# Mel spectrogram
melspec = librosa.feature.melspectrogram(y=y, sr=sr, n_fft=n_fft,
hop_length=n_hop, n_mels=n_mels)

# 2D mel-spectrogram 데이터를 로드합니다.
mel_spectrogram = melspec
mel_spectrogram = mel_spectrogram.astype(np.float32)

# RGB 이미지로 변환합니다.
rgb_image = np.zeros((mel_spectrogram.shape[0],
mel_spectrogram.shape[1], 1), dtype=np.float32)
rgb_image[..., 0] = mel_spectrogram

input = rgb_image

# RGB 이미지를 저장합니다.
cv2.imwrite('rgb_image.jpg', rgb_image)

input = np.array([input])
out = model.g_BA.predict(input)
out = out.squeeze()
audio = librosa.feature.inverse.mel_to_audio(
    out, sr=sr, n_fft=2048, hop_length=512, win_length=None, window='hann',
center=True, pad_mode='reflect')

import soundfile as sf

```

```
# 생성한 오디오 파일(audio)을 WAV 파일로 저장
```

```
sf.write('/content/output.wav', audio, sr)
```

이 코드는 음원 파일을 Mel 스펙트로그램으로 변환한 후, CycleGAN 모델을 사용하여 Mel 스펙트로그램을 다른 도메인으로 변환하고, 다시 원래 도메인으로 복원하는 과정이다.

주요 단계는 다음과 같다:

우선 음악 파일을 로드하고, 필요에 따라 잘라내는 작업을 수행한다.

그 후, librosa 라이브러리를 이용하여 음원의 mel-spectrogram을 계산한다.

계산된 mel-spectrogram에 단일 channel GrayScale 이미지로 변환한다.

변환된 GrayScale 이미지를 저장한다.

변환된 GrayScale 이미지를 CycleGAN 모델의 `model.g_BA.predict()`에 입력으로 넣고 반환값을 `out`에 저장한다.

`out`을 오디오 신호로 복원한다.

복원된 오디오 신호를 WAV 파일로 저장한다.

제 4장 말은 부분

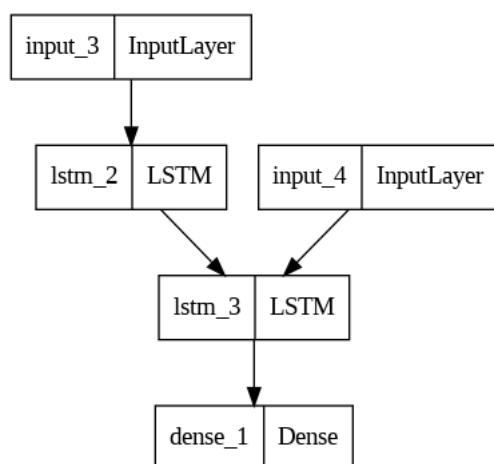
전체 프로젝트의 역할 분담은 모델링 파트와 데이터 전처리 파트로 나누어 진행하였다.

AI 모델링, 차원 확장(이홍석), 데이터 전처리 및 수집(박민재)

1) LSTM을 활용한 Decoder – Encoder 모델

음원 데이터의 시계열적 특성을 활용하는 방법으로, 동일 음원의 피아노 편곡 버전을 내려받아 paired-learning 방법으로 훈련을 진행하였다.

(ex. 황혼 기타 버전 – 황혼 피아노 버전 음원을 각각 다운로드 후, 라벨링하여 1대1로 훈련)

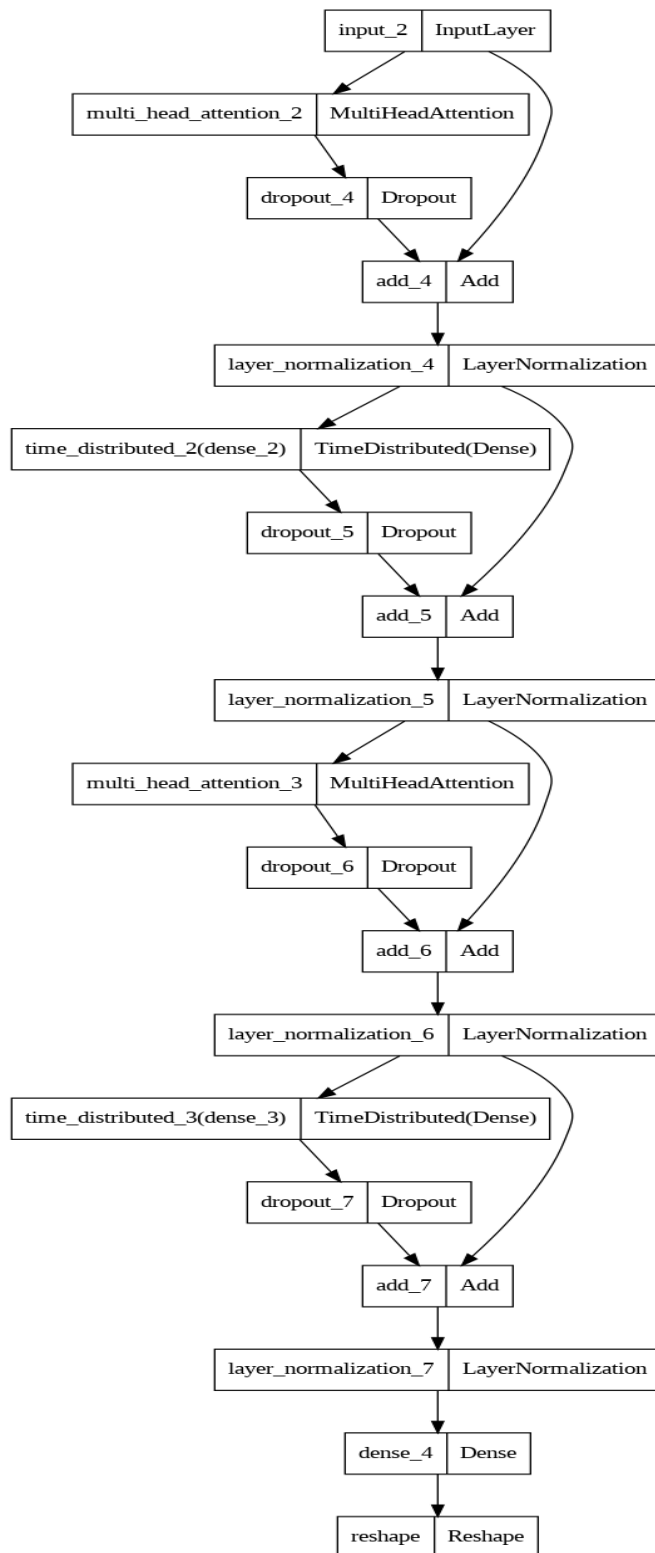


그 결과, 음원 데이터의 편곡 버전의 박자 및 타이밍이 완전히 일치하지 않음으로 인해 시계열적 특성을 모델이 잘 알아차리지 못했으며, LSTM을 활용한 모델의 특성 상, 음원이 조금만 길어지더라도 예측력이 급격히 감소함을 확인하였다.

2) Transformer를 활용한 시계열 예측 모델

LSTM의 길이 측면에서의 한계를 극복하기 위해, 보다 긴 시계열 예측에서도 높은 효율을 보이는 Transformer 모델을 사용하였다. 그 결과, 이미 학습된 데이터에서는 비교적 긴 길이에서도 양호한 예측력을

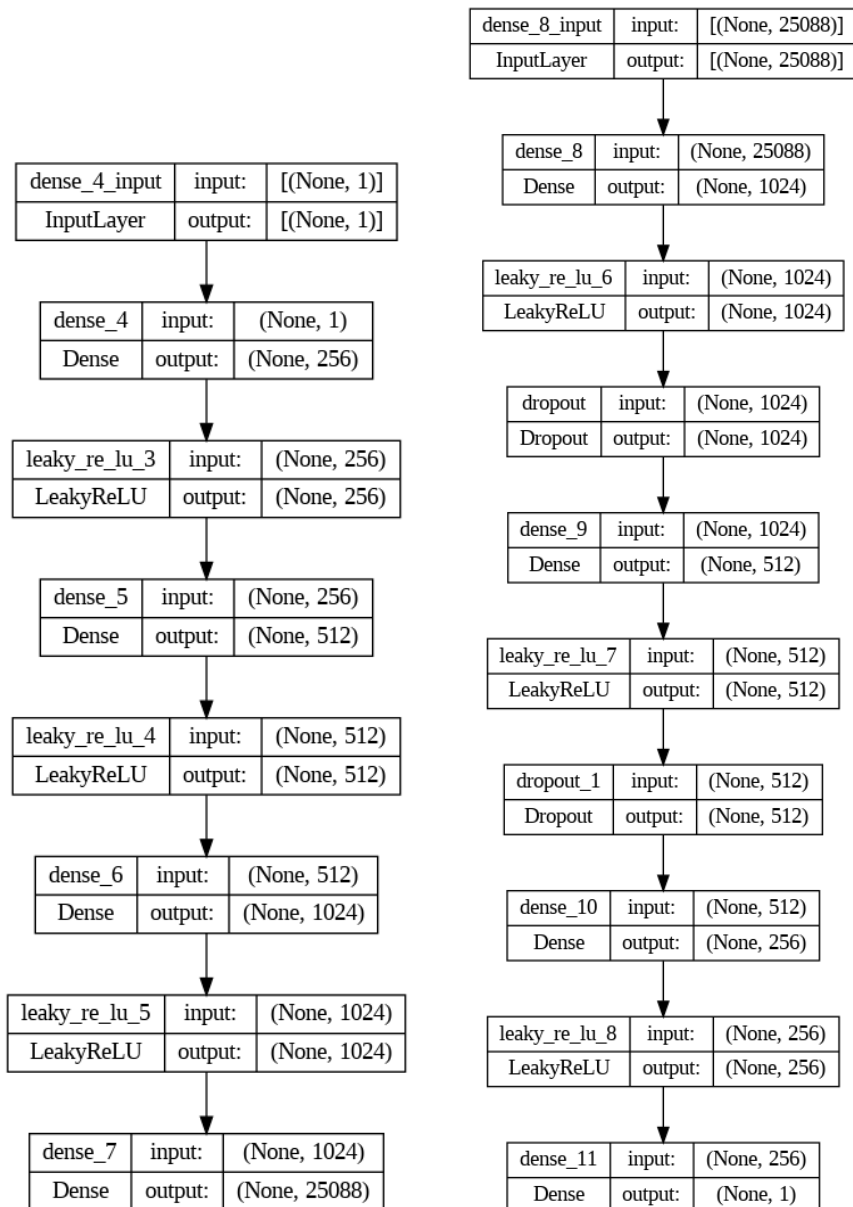
보였으나, 이 역시 학습 데이터 외에 test data에서의 성능이 아주 좋지 않았다.



위 모델들을 실행한 결과, GAN모델이 음성 생성에 적합하다고 판단하여, GAN모델을 선정하여 작업하였다.

3) GAN모델

GAN모델 중, paired learning을 사용하는 GAN모델을 사용한 결과, 마찬가지로 패턴을 잘 읽어내지 못했으며, 노이즈에 가까운 음성들이 주로 생성되었다.



4) 데이터 전처리(차원 확장)

```
import cv2
import numpy as np

# 2D mel-spectrogram 데이터를 로드합니다.
mel_spectrogram = melspec

# mel-spectrogram 데이터를 0~255의 정수값으로 변환합니다.

#mel_spectrogram = mel_spectrogram.astype(np.float32)
'''
mel_spectrogram -= np.min(mel_spectrogram)
mel_spectrogram /= np.max(mel_spectrogram)
mel_spectrogram *= 255
'''

mel_spectrogram = mel_spectrogram.astype(np.float32)

# RGB 이미지로 변환합니다.
rgb_image = np.zeros((mel_spectrogram.shape[0], mel_spectrogram.shape[1], 1),
dtype=np.float32)
rgb_image[..., 0] = mel_spectrogram
'''
rgb_image[..., 1] = 0
rgb_image[..., 2] = 0
'''

input = rgb_image

# RGB 이미지를 저장합니다.
cv2.imwrite('rgb_image.jpg', rgb_image)
```

CycleGAN은 이미지 데이터에 최적화되어 있어, Input data는 channel차원을 확보해야 한다. 따라서, 강제로 2차원의 mel-spectrogram의 차원을 확장하여, CycleGAN의 input으로 사용할 수 있도록 하였다.