JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 22, No. 3, pp. 375-381, Mar. 2021

Check for updates

OpenAI Jukebox의 VQ-VAE 알고리즘을 이용한 실시간 알고리즘 작곡 시스템 제작연구

전 명 준[1] · 김 준[2*]
[1]동국대학교 영상대학원 멀티미디어학과 박사과정
[2*]동국대학교 영상대학원 멀티미디어학과 교수

A Study on the Production of Real-Time Algorithmic Composition System Using the VQ-VAE Algorithm of OpenAI Jukebox

Myong-Jun Jeon[1] · Jun Kim[2*]
[1]D.M.A. Candidate, Department of Multimedia, Graduate School of Digital Image and Contents, Dongguk University, Seoul, Korea
[2*]Professor, Department of Multimedia, Graduate School of Digital Image and Contents, Dongguk University, Seoul, Korea

[요 약]

기존의 컴퓨터를 이용한 알고리즘 작곡 사용자 제작 환경들은 일정 수준의 코딩이나 프로그래밍을 거쳐 선언된 변인을 기반으로 사전정의된 함수들을 이용하여 계산된 데이터를 기다려야 사용할 수 있어, 실시간으로 알고리즘 작곡을 통한 작품 제작에 어려움이 있다. OpenAI의 Jukebox를 통해 실시간으로 녹음된 오디오 데이터를 이용하여 알고리즘 작곡이 가능하며 Max/MSP와 같은 인터랙티브 미디어아트 제작환경에서 다양하게 사용되는 프로그램과의 유기적인 결합이 이루어질 경우 최소한의 개입으로 우연성에 의한 실시간 알고리즘 멀티미디어 작품의 제작이 가능해진다.

[Abstract]

Algorithm composition user production environments using a computer can be used only by waiting for data calculated using predefined functions based on variables declared through a certain level of coding or programming, making it difficult to produce works through algorithmic composition in real time. Algorithm composition is possible using audio data recorded in real time through OpenAI's Jukebox, and when organic combination with various programs used in interactive media art production environments such as Max/MSP is achieved, it is accidental with minimal intervention. Jukebox is an AI that takes musical quality, consistency, audio sample length, and ability to control artists, genres and lyrics to the next level, but it is quite different from human-made music. However, with the least human intervention that could not be done in other AI programs, real-time algorithm works can be produced by chance, which will allow them to more be free in a tool called a program for creating real-time multimedia works and produce a wide range of results, not limited ones.

색인어 : 알고리즘작곡, 컴퓨터음악, OpenAI Jukebox, Max/MSP
Key word : Algorithmic composition, Computer music, OpenAI Jukebox, Max/MSP

http://dx.doi.org/10.9728/dcs.2021.22.3.375

Received 07 December 2020; **Revised** 20 January 2021
Accepted 20 January 2021

***Corresponding Author; Jun Kim**
Tel: ※ 개인정보 표시제한
E-mail: myongjun.jeon@gmail.com

Ⅰ. 서 론

Amazon의 DeepComposer[1], 스탠포드 대학교 CCRMA에서 개발하고 일리노이 대학에서 Rick Taube가 연구 중에 있는 GRACE[2]와 같은 알고리즘 작곡 사용자 제작환경들은 모티브 또는 제어가 필요한 기초적인 변인을 선언해주고 계산을 기다려야 사용할 수 있어 실시간으로 알고리즘 작곡을 통한 작품 제작에 어려움이 있다.

Jukebox[3]는 압축된 오디오 파형과 다양한 가사를 훈련시킨 작곡 AI이다. 원래의 음원 소스를 이용하여 모델을 훈련시키고 모델에 새로운 음원을 생성할 수 있는 기능을 제공한다. 다른 음악 생성 프로그램은 실제 음원이 아닌 음표, 피치에 대한 정보와 같은 한정된 정보를 통해 수치화된 데이터를 출력하는데 반해, Jukebox는 실제 들을 수 있는 음원 형태로 추출할 수 있다. 실제 음원 파일은 기호화된 데이터보다 순서의 진행(시퀀스)이 훨씬 길어지는 특징이 있다. 수치화된 데이터에 비해 직접 입력시키기가 상대적으로 어렵다.

본 연구에서는 알고리즘이 인식 가능한 부호로 직접 입력시키기 어려운 실제 연주되고 있는 음원의 입력을 해결할 수 있는지에 대해 탐구하고 이를 통해 실시간 알고리즘 멀티미디어작품의 제작 시스템을 만들어 보고자 한다.

Ⅱ. 본 론

2-1 VQ-VAE의 음악 분석

기존의 기호화된 데이터만을 생성하는 프로그램에는 한계가 있다. 오디오 데이터에 비해 인간의 목소리나 음악에 필수적인 보다 미묘한 시간, 역동성, 표현성을 포착할 수 없다. 보통의 오디오 데이터 수준으로 음악을 생성하는 것은 쉬운 일이 아니다. 44100 Hz의 샘플 레이트(Sample Rate)와 16Bit의 비트 뎁스(Bit-depth)를 가진 CD 음질의 전형적인 4분짜리 곡은 천만 개가 넘는 타임 스텝(time step)[4]을 가지고 있다. 따라서 오디오 데이터가 가진 높은 수준의 의미론을 배우기 위해서는 모델이 극도로 장기적인 의존성을 다루어야 한다.

VQ-VAE[5](Vector Quantized Variative Auto Encoder)는 구글 DeepMind사에서 개발한 표현학습 알고리즘 중 하나로서, 기계의 신경 이산 표현 학습(Neural Discrete Representation Learning)법 중 하나이다. 오디오 데이터의 높은 수준의 의미를 분석하기 위해 Jukebox는 음악에서 지각적으로 인지되는 관련 정보를 파기하고 원시 오디오[6][7][8][9]만을 회선 신경망으로 압축하여 분석한다.

Jukebox는 오디오를 연속적이지 않은 수많은 부분으로 나눈다. 예를 들어 음원을 0.0078125 (1/128)초 길이의 비트(beat)로 나누는 방식들로 오디오의 연속적인 특성을 다룬다. Jukebox 연구팀은 VQ-VAE를 단순화시킨 변종인 VQ-VAE-2를 이용하여 피드포워드(feed-foward) 인코더와 디코더만을 사용하여 높은 충실도의 사운드 이미지를 생성하였다.

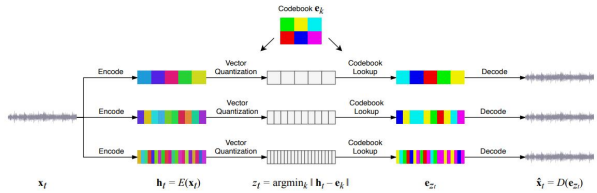

그림 1. VQ-VAE를 이용한 오디오 데이터 분석[7]
Fig. 1. Using VQ-VAE for analyzing audio data[7]

VQ-VAE에 의해 인코딩된 음악 코드의 진행을 배우고 단순화된 데이터 속에서 음악을 생성하는 것을 목표로 모델들을 훈련시킨다. 모델들은 음악의 장거리 구조와 오디오 음질은 낮지만 노래와 멜로디와 같은 높은 수준의 의미론들을 포착한다. 모델들의 재교육을 통해 음색과 같은 현지 음악 구조를 추가해 오디오 음질을 대폭 개선시킨다. 일단 모든 모델들이 훈련을 받으면, 최상위 레벨에 위치한 모델은 단순화된 데이터를 오디오 데이터로 변환시킬 수 있는 코드를 생성하고 업샘플러를 사용하여 그것들을 업샘플링한 다음, VQ-VAE 디코더를 사용하여 다시 오디오 공간으로 디코딩하여 새로운 곡 진행을 샘플링한다[10].

Jukebox 연구팀은 VQ-VAE 모델에 웹을 탐색하여 리릭위키(LyricWiki)의 해당 가사와 메타데이터와 짝을 지어 120만 곡(이 중 60만 곡은 영어로 되어 있음)의 새로운 데이터 집합을 큐레이션하여 모델에 입력했다. 메타데이터에는 아티스트, 앨범 장르, 곡의 연도 등이 수록되어 있으며, 각 곡과 관련된 공통적인 분위기나 재생목록 키워드 등이 수록되어 있다. 32Bit의 비트 뎁스, 44.1kHz 샘플 레이트를 가진 원시 오디오로 훈련하고, 좌우 채널을 무작위로 믹스하여 모노 오디오를 만들어 비교 데이터를 입력시켰다(그림1).[11]

2-2 Jukebox와 Max/MSP의 연동을 통한 실시간 알고리즘 작곡 시스템

본 연구에서는 실시간 오디오를 Jukebox에 입력시키고 입력받은 오디오를 다양한 방법으로 사운드 프로세싱이 가능한 이점을 가진 Max/MSP[12][13]를 함께 활용하기로 한다. Max/MSP는 노드기반의 프로그래밍 언어 혹은 개발 환경으로써, 노드 기반 특유의 작동 방식으로 인해 활용 범위가 매우 넓어 인터랙티브 작품을 창작하는 아티스트들이 선호하는 프로그램이다. 악기나 혹은 오실레이터를 사용하여 실시간으로 연주되며 발생하는 소리를 Max/MSP에서 디지털 파일 형태로 전환시켜 Jukebox에 입력하고, 입력한 파일을 Jukebox가 분석하고 신경망을 통해 새로운 형태의 음원으로 전환시킨다. 전환된 음원을 다시 파일 형태로 Max/MSP로 전송시켜 사운드 프로세싱 과정을 거친 후 재생한다(그림2).

Jukebox를 제어하기 위한 API 소스 코드는 파이선 (Python)의 형태로 제공된다. 그러나 독립적인 프로그램으로서 실행하고자 할 경우 의존하는 외부 라이브러리가 많아 파이선과 Jukebox API만 가지고는 정상적인 실행이 어렵다.

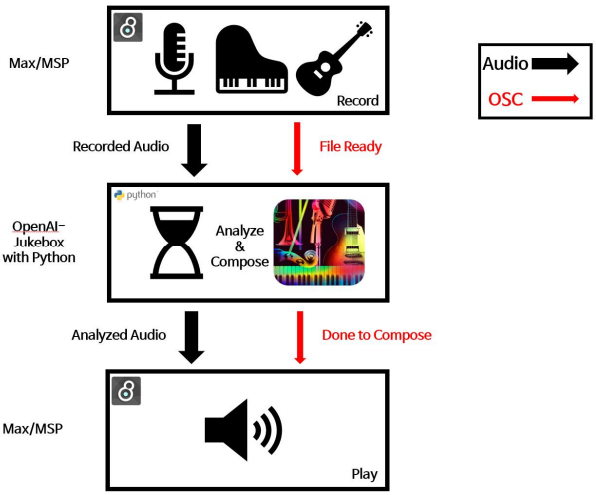

그림 2. 실시간 알고리즘 작곡 시스템 도식
Fig. 2. Real-time Algorithmic Composition System

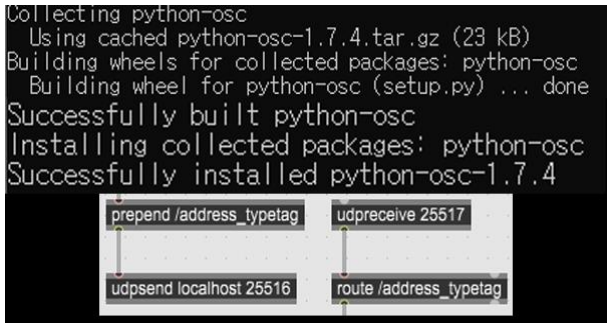

그림 3. 파이선과 Max/MSP의 OSC통신 기능 설치
Fig. 3. Installing OSC protocol on Max/MSP and Python

따라서 Anaconda라는 별도의 디펜던시 라이브러리와 Nvidia사의 GPU에 탑재된 CUDA코어를 계산에 활성화시킬 수 있는 툴킷(Toolkit)을 파이선에 설치하여 사용해야 한다. 이후 동일 컴퓨터에 Max/MSP를 설치하면 기본적인 소프트웨어 설정은 완료되나, 만일 Max/MSP와의 연동 외에 독립적으로 Jukebox를 활용할 경우 위 필수 설치 라이브러리만 설치하면 된다.

Max/MSP와 Jukebox(정확히는 파이선)과의 연결을 위해서는 OSC(Open Sound Control)를 사용하기로 한다. OSC는 네트워크 기반의 프로토콜로써 개별 프로그램이 지원만 한다면 상호 데이터 교환이 가능하며, OSC 통신을 통한 타 프로그램의 실행 과정을 제어할 수 있다. Max/MSP에서는 네이티브로 OSC 통신을 지원하지만 파이선의 기본 환경에서는 지원하지 않는다. 그러나 외부 라이브러리인 python-osc를 설치하면 파이선 또한 OSC 통신이 가능해진다(그림3).

OSC 통신이 상호 가능하게된 상태에서, 파이선과 Max/MSP 사이에 통신에 필요한 포트 번호와 입력주소를 통일시켜준다. Max/MSP와 Jukebox API가 로드된 파이선 프로그램 사이에 다음과 같은 코드를 입력하여 상호간의 통신을 가능하게 한다(그림4).

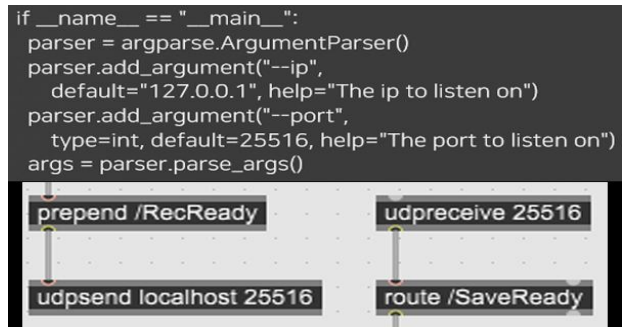

그림 4. 파이선과 Max/MSP 간의 상호연결
Fig. 4. Interactive connection into Python and Max/MSP

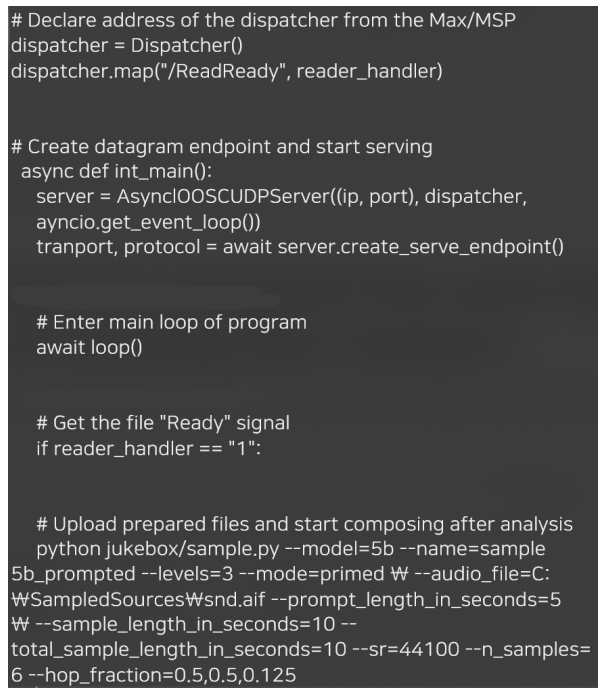

그림 5. 파이선 상에 로드된 Jukebox와 OSC 통신 코드 일람
Fig 5. Loaded on Python as OSC source-code with Jukebox API

OSC는 다음과 같은 상황을 처리했을 때 서로 상호 통신하기 위해 사용한다. Max/MSP의 buffer~ 오브젝트 및 실시간 소리 녹음 기능을 이용하여 연주되는 악기 혹은 발현되는 소리를 sfplay~ 오브젝트에 실시간으로 녹음하여 RAM 혹은 하드디스크에 파일 형태로 저장한다. Max/MSP에서 녹음 할 경우 오디오 파일 포맷이 snd.aif 형태로 저장된다. 파일이 저장됨을 완료하였다는 신호를 OSC를 이용하여 파이선에 전송시킨다(그림5).

송신확인 이후 파이선은 Jukebox에 해당 오디오 파일을 업로드 하고, 업로드 된 파일을 기반으로 분석 작업을 요청한다. 완료된 분석 파일의 실시간 변환 작업을 거친 뒤 나오는 파일을 저장한 뒤, OSC로 저장을 완료하였다는 신호를 Max/MSP에 전송시킨다(그림6).

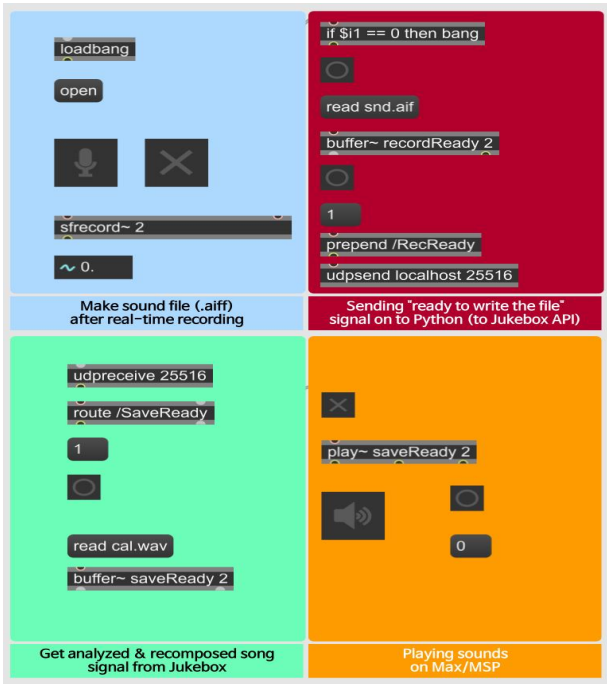

그림 6. Max/MSP 실시간 오디오 녹음/재생 시스템
Fig. 6. Real-time recording and playing system using Max/MSP

```
# Clean up serve endpoint
transport.close()

  # Send file save ready message
  cliend.send_message("/SaveReady", 1)
```

그림 7. Jukebox가 탑재된 파이선 코드에서 분석이 완료됨을 OSC로 전송
Fig 7. Sending the 'ready' message to the Max/MSP using OSC protocol

완료신호와 함께 Max/MSP에서 로드시키고 재생한다(그림 7). 장르별, 악기별 사전분석된 모델과 컴퓨터 자원에 따라 분석에 소요되는 시간은 차이가 발생하였으며, 빠른 분석과 계산을 얻기 위해 클라우드 컴퓨팅을 이용하는 것을 제안한다.

Ⅲ. 적용 예시

3-1 적용 예시

Jukebox에 탑재된 다양한 샘플 분석 신경망 모델 중 3개(5b, 5b_lyrics, 1b_lyrics)를 기반으로 훈련시킨 모델을 토대로 예시 작품에 각각 적용하여 보고 모델 별 차이를 알아보았다.

피아노를 이용하여 다음과 같은 악보를 실시간으로 연주하고, 이를 Max/MSP로 실시간 녹음하였다(그림8).

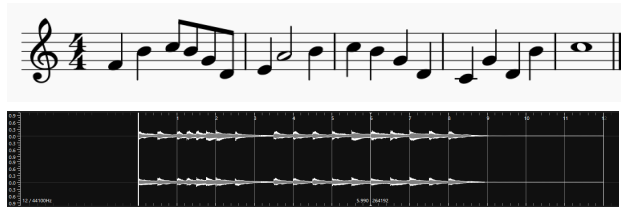

그림 8. 예시 작품 음원의 파형 및 악보
Fig. 8. The waveform and notation of sample song

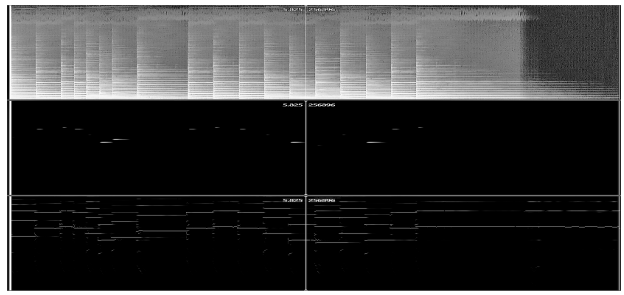

그림 9. 예시 작품의 스펙트로그램
Fig. 9. The spectrogram of sample song

Max/MSP에서 디지털 오디오 형태로 변환된 파일의 스펙트로그램을 살펴보면, 기음과 각 음의 배음이 뚜렷하게 나타나는 것을 볼 수 있었다(그림9).

먼저 입력받은 오디오를 가지고 각 신경망 모델에서 도출된 결과를 살펴보고 예시 작품에 가장 맞는 신경망 모델을 찾고자 하였다. 먼저 5b_lyrics 신경망 모델을 이용하여 녹음한 음원을 분석하였다. 5b_lyrics 모델은 록, EDM, 힙합과 같은 대중음악을 재구성하는 것에서 장점이 있는 모델로서, 각 음원의 사람의 음성을 찾아내고 음성에 포함된 가사를 분석하여 기존 음원의 단어나 어절이 포함된 멜로디를 재배치할 수 있다. 첫 분석에는 계산에 필요한 CUDA 코어가 탑재된 RTX 2080 그래픽 카드를 이용 하였으며, 7초의 음원을 분석하는 데에 약 3시간 43분의 시간이 소요되었다. RTX 2080의 전용 비디오 메모리는 통상적으로 8GB인데, 5b_lyrics 모델이 20여 초의 음원을 완전히 분석하는데 통상적으로 필요한 전용 비디오 메모리가 11GB에 달해 부족한 전용 비디오 메모리를 보충하기 위하여 일반적인 RAM의 메모리 용량과 전용 비디오 메모리를 통합하여 사용하였다. 이로 인해 두 메모리 간의 데이터 처리에 있어 병목 현상이 생겨 짧은 음원임에도 불구하고 처리 시간이 기하급수적으로 증가하는 문제가 생겼다.

5b_lyrics 모델을 이용하여 분석한 예시 작품의 결과는 다음과 같다. 음원에 사람의 음성과 관련된 정보가 없었기 때문에 '가사'를 개사하듯이 음성이 포함된 멜로디 부분을 재배치할 수 있는 부분이 존재하지 않았다. 이로 인해 멜로디가 재배치된 부분의 변화 폭이 기존의 곡의 진행과는 거의 차이가 없었고 재배치된 부분의 음질이 원래 음원에 비하여 심각하게 열화되어 들렸으며, 실제 악기로 연주하였던 원래 음원과의 진행과는 큰 차이를 보이지 않고 있었다. 전반적인 음원의 음질 또한 다소 열화되어 들렸다(그림10).

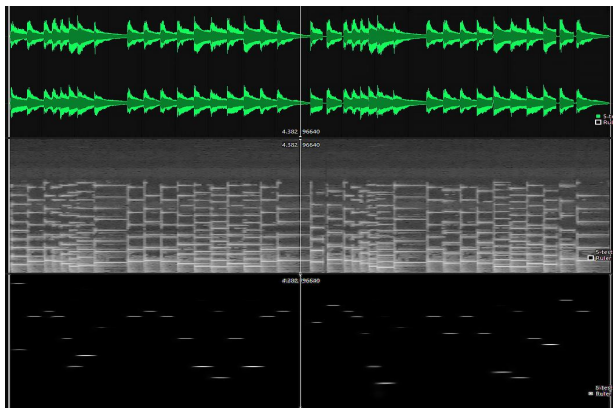

그림 10. 5b_lyrics 모델을 통해 멜로디가 재배치된 음원
Fig. 10. The spectrogam with waveform of the song using 5b_lyrics model as recomposed melody

5b_lyrics 모델의 소요 시간을 줄이고자 아마존 사(Amazon inc.)의 클라우드 컴퓨팅 서비스를 이용하여 클라우드 서버의 자원을 활용하였다. 클라우드 컴퓨팅 서버의 GPU 자원 중 하나인 A100 GPU를 사용하여 RTX 2080 그래픽 카드 대비 12배 더 많은 전용 비디오 메모리와 3배 더 많은 CUDA 코어를 분석에 투입하였다. 7초의 음원을 분석하는 데에 기존 약 3시간 43분에서 29분으로 획기적으로 줄일 수 있었다.

그러나 29분이라는 시간 또한 작품이 진행되는 시퀀스에서는 매우 긴 시간이라는 문제가 있다. 따라서 5b_lyrics 모델은 클라우드 컴퓨팅의 매우 방대한 자원을 사용하여도 실시간 시스템 제작이라는 의미에서는 적합하지 않음을 알 수 있었다.

다음으로 5b 신경망 모델을 이용하였다. 5b 모델은 5b_lyrics 모델에서 가사를 분석하여 재배치하는 기능이 제거되고 멜로디와 코드가 재배치되는 기능이 강화된 모델이다. 똑같이 그림 8에 제시된 음을 피아노로 연주하고 이를 녹음하여 5b 모델에 입력시켰다. 첫 분석 역시 5b_lyrics 모델의 첫 분석과 동일한 RTX2080 그래픽 카드를 이용하여 7초의 음원을 분석하였고 걸리는 시간 또한 5b_lyrics 대비 2시간이 단축된 약 1시간 40여 분의 시간이 소요되었다.

5b 모델 또한 전용 비디오 메모리 부족으로 인해 일반적인 RAM을 통합시켜 통합 비디오 메모리로 분석하였으나, 데이터 처리에 있어 두 메모리 간의 병목 현상은 여전하였다. 특히 5b_lyrics 모델에 비하여 5b 모델의 분석은 후반 90% 분석률에서 100%의 분석률까지의 걸리는 시간이 분석을 시작하여 중후반부까지 걸리는 시간보다 더 오랜 시간이 소요되는 문제가 있었다. 5b 모델 또한 5b_lyrics와 동일한 클라우드 컴퓨팅 자원 환경에서 다시 분석하였고 7초의 음원을 분석하는 데에 종전 1시간 40여 분에서 자원 투입에 따라 최대 3분 대의 분석에 필요한 소요 시간을 확보할 수 있었다.

5b_lyrics의 29분 대비 최대 3분 대의 시간 단축을 통하여 작 시퀀스의 한 부분에 시스템을 사용할 수 있을 정도의 사용 시간 확보가 되었다.

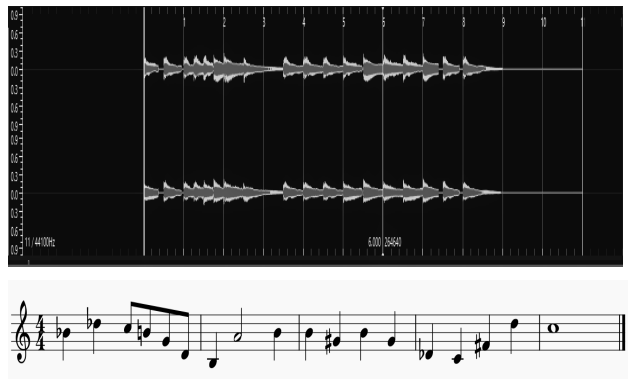

그림 11. 변환된 음원의 파형 및 음원을 악보화 하였을 때
Fig 11. The waveform and notation of recomposed song

그러나, 작품의 시작 혹은 끝에 배치되어야 시스템의 분석 도중 생기는 에러를 해결하기 비교적 쉽고 작품의 시퀀스에 매끄러운 진행에 영향을 덜 미친다는 점이 문제점 이었다.

도출된 결과는 실제 악기로 연주한 원래 음원에 비해 피치(pitch)가 모호하게 들렸다. 재생되는 음을 스펙트로그램을 통해 분석하였을 때 보여지는 주파수를 기반으로 근접 주파수 중 비슷한 피치 값으로 변환하여 악보화 시켰다(그림11). API에서의 속성 변경을 통해 밝은 느낌의 원래 음원을 단조를 연상케 하는 음원으로 변환시켰다.

Jukebox의 연산 방법상 다운샘플링(down sampling)후 업스케일(upscale)하는 방식으로 음원을 분석하기 때문에 스펙트로그램 상에서도 음 하나 하나마다 불필요한 노이즈가 생성되었으며, 특히 연주된 음원의 배음과는 확연히 다르게 거의 모든 대역에서 배음이 뚜렷하게 증가된 것을 볼 수 있었다(그림12).

다음으로 샘플 신경망 모델 중에서 마지막으로 제공하는 1b_lyrics 신경망 모델을 사용하였다. 1b_lyrics 모델은 5b_lyrics와 5b 모델이 완벽한 분석 후 음원의 멜로디나 화성을 재배치하는 것에 비해 적은 컴퓨터 자원에서 사용할 수 있도록 분석 알고리즘이 간소화된 모델이다. 5b_lyrics 대비 필요한 전용 비디오 메모리가 3.8GB밖에 안 될 정도로 적은 자원을 필요로 하며, 5b_lyrics 대비 분석되는 샘플의 수 또한 1/5밖에 되지 않는다.

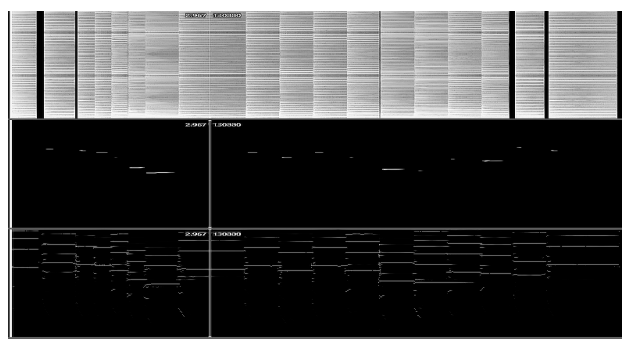

그림 12. 변환된 음원의 스펙트로그램
Fig. 12. The spectrogram of recomposed sample song

```
mpiexec -n {ngpus} python jukebox/train.py --hps=small_vqvae,small_prior,all_fp16,cpu_ema --name=small_prior ₩
--sample_length=100912 --bs=4 --audio_files_dir=C:₩max-sounds₩--labels=False --train --test --aug_shift --aug_blend ₩
--restore_vqvae=logs/small_vqvae/checkpoint_latest.pth.tar --prior --levels=2 --level=1 --weight_decay=0.07 --save_iters=700
```

그림 13. Jukebox API에서의 우선순위 속성 설정
Fig. 13. The attribute of sampling priority on the API

정확한 재배치를 기대할 수는 없으나 빠른 분석을 통해 실시간에 준한 결과를 기대할 수 있었다.

RTX 2080 그래픽 카드를 사용하여 1b_lyrics 모델로 앞서 사용하였던 7초 길이의 음원을 분석하였고 소요된 시간은 27분으로 매우 준수한 시간을 보여주었다.

RTX 2080로 일반 RAM과의 통합 메모리 설정 없이 가지고 있는 전용 비디오 메모리 8GB로도 분석이 가능하였다. 그러나 1b_lyrics 분석 모델의 작업 도중 프리즈 현상이 생기거나 알 수 없는 오류로 생성된 음원의 재생이 되지 않는 경우가 간헐적으로 발생하였다. 5b_lyrics 모델과 5b 모델에 비해 27분의 소요 시간은 빠른 처리 시간임에는 분명한 사실이나, 작품에서의 실시간 사용에서는 지장이 있는 시간인 만큼 앞서 사용하였던 클라우드 컴퓨팅 자원을 활용하여 소요 시간을 단축시키고자 하여 평균 43초, 최대 16초까지 단축시킬 수 있었다. 실시간 사운드 프로세싱에서 사용되는 단위인 밀리세컨(milliseconds; 1/1000 seconds)보다는 시간이 매우 길지만, 작품의 시퀀스에 있어 처음과 끝부분에서 활용하는 것보다는 적은 시간이 가지는 이점을 통해 다양한 부분에서 활용할 수 있었다.

또한 5b_lyrics 모델과 5b 모델은 기존에 존재하는 파라미터에서 작업 결과물의 시간 제한만 해주어도 다소 만족스러운 결과물을 얻을 수 있었으나, 1b_lyrics 모델에서는 모든 샘플을 분석하는 과정을 거치지 않기에 음원에서의 특정 시간대의 샘플이나, 악기의 특성 또는 사람 음성의 특성과 연관된 음역대의 샘플영역 분석에 우선순위(priority)를 주어야 한다. API 입력 시 별도의 속성(attribute)을 주어 어떤 음역대나 시간대 샘플에 분석 우선을 둘 것인지 결정할 수 있다(그림13).

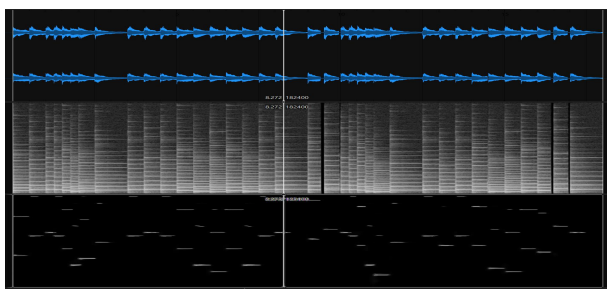

그림 14. 중고음역대의 배음과 저음역대의 배음 증가
Fig. 14. More harmonics on mid-range and high-range frequency area

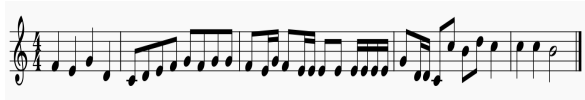

그림 15. 변환된 음원음원을 악보화 하였을 때
Fig. 15. The notation of recomposed song

1b_lyrics 모델을 이용한 분석 결과 악기에 우선 순위를 두었음에도 불구하고 피아노 악기의 음질이 매우 떨어졌으며, 5b 모델에 비해 모든 대역의 배음보다는 특정 대역의 배음이 증가됨을 볼 수 있었다(그림14). 또 몇몇 재분석 시 샘플 레이트가 절반으로 출력되어 본래 음원의 빠르기의 2배로 들리는 경우도 있었다.

재배치된 음원을 스펙트로그램을 통해 분석하였을 때 보여지는 주파수를 들리는 음과 비슷한 피치 값으로 변환하여 악보화 시켰다(그림15). 피아노 대역의 우선순위를 주고 리듬 분절에 –hop_fraction이라는 랜덤을 주어 흐트려뜨렸다. 같은 음이 반복되는 점이 아쉬우나 단순한 본래 음원에 비해 더 짧은 리듬을 가진 음원이 생성되었다.

샘플로 제시된 모델을 기반으로 작품에 적용하여 보았을 때 가장 만족스러운 모델은 5b 모델이었다. 예시 음원에 사람의 목소리가 포함되었을 경우 만족스러운 모델은 5b_lyrics가 되었을 것이지만, 단일 악기 혹은 단일 멜로디를 가진 음원의 경우 5b 모델이 좀 더 만족스러운 결과물을 만들어내었다. 그러나 5b_lyrics와 5b 모델은 분석에 소요되는 시간이 컴퓨터 자원에 따라서 기하급수적으로 증가되는 문제점이 있어 실시간으로 활용하는 것에 문제가 있었다. 1b_lyrics 모델은 빠른 시간에 적당한 결과물을 내어주는 장점이 있었지만 빠른 결과를 위해 음질을 상당부분 희생시켜 타 DAW 혹은 Max/MSP와 같은 사운드 프로세싱을 처리할 수 있는 프로그램을 사용하지 않았을 때 원 음원과 재배치된 음원 사이의 음질 위화감이 매우 크게 느껴진다는 문제가 있었다.

Ⅳ. 결 론

Jukebox는 음악적 품질, 일관성, 오디오 샘플 길이 및 아티스트, 장르 및 가사를 조절하는 능력을 한 단계 발전 시킨 AI 이지만, 인간이 만든 음악과는 상당한 차이가 있다. 예를 들어, 생성된 노래는 음악적 일관성을 보여주고, 전통적인 코드 패턴을 따르며 인상적인 솔로를 피처할 수 있지만, 반복되는 후렴, 대중에게 익숙한 캐논코드와 같은 친숙한 더 큰 구조는 들리지 않는다. 또한 효율적인 컴퓨팅 자원으로는 원하는 작품의 분석에 필요한 시간이 오래 걸린다는 단점 또한 존재한다. 하지만 다른 AI 프로그램에서는 할 수 없었던 최소한의 인간의 개입으로 우연성에 의한 실시간 알고리즘 작품의 제작이 가능하며 이를 통해 실시간 멀티미디어 작품의 창작 시 프로그램이라는 도구 안에 갇혀 한정적인 결과물이 아닌 폭 넓은 결과물을 도출시킬 수 있을 것이다.

참고문헌

[1] Introducing Amazon DeepComposer™ [Internet]. Available: http://aws.amazon.com/deepcomposer/

[2] Common Music GRACE (Graphic Real-time Algorithmic Composition Environment); Introduction [Internet]. Available: http://commonmusic.sourceforge.net/cm/res/doc/cm.html

[3] Vasquez, Sean, and Mike Lewis. “Melnet: A generative model for audio in the frequency domain.” arXiv preprint arXiv:1906.01083 (2019), p.1 .

[4] Understanding LSTM Networks [Internet]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[5] Van den Oord, Aaron, and Oriol Vinyals. “Neural discrete representation learning.” Advances in Neural Information Processing Systems. 2017.

[6] Oord, Aaron van den, et al. “Wavenet: A generative model for raw audio.” arXiv preprint arXiv:1609.03499 (2016)

[7] Mehri, Soroush, et al. “SampleRNN: An unconditional end-to-end neural audio generation model.” arXiv preprint arXiv:1612.07837 (2016)

[8] Vasquez, Sean, and Mike Lewis. “Melnet: A generative model for audio in the frequency domain.” arXiv preprint arXiv:1906.01083 (2019)

[9] Yamamoto, Ryuichi, Eunwoo Song, and Jae-Min Kim. “Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram.” ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020 p.3.

[10] Child, Rewon, et al. “Generating long sequences with sparse transformers.” arXiv preprint arXiv:1904.10509 (2019).

[11] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, Ilya Sutskever, “Jukebox: A Generative Model for Music ”, OpenAI, 2020

[12] A Playground for Invention [Internet] Available: https://cycling74.com/products/max

[13] M.J. Jeon, Interactive Multimedia Performance using Real-time Algorithmic composition (forcus on multimedia music <Angular Orbit>), MA, Dongguk University (Graduate School of Digital Media and Contents), Seoul, 2019

전명준(Myong-Jun Jeon)

2019년 : 동국대학교 영상대학원 (컴퓨터음악석사)
2020년 : 동국대학교 영상대학원 (컴퓨터음악박사과정)

2013년～2017년 : 국가평생교육진흥원 실용음악학 학사
2017년～2019년 : 동국대학교 영상대학원 멀티미디어학과 컴퓨터음악전공 석사
2019년～현재 : 동국대학교 영상대학원 멀티미디어학과 컴퓨터음악전공 박사과정
※관심분야 : 알고리즘 작곡, 실시간 인터랙티브 퍼포먼스, 컴퓨터음악

김 준(Jun Kim)

1989년 : 경희대학교(음악학사)
1994년 : Boston University(음악석사)
1999년 : Stanford University(음악박사)

2001년～현 재 : 동국대학교 영상대학원 멀티미디어학과 교수
※관심분야 : 멀티미디어컨텐츠, 컴퓨터음악, 인터랙티브 퍼포먼스