# Jukebox: A Generation Model for Music

소프트웨어학부
2018044566 박서연

# Introduction

Jukebox는 raw audio로부터 새로운 음악을 생성해내는 모델

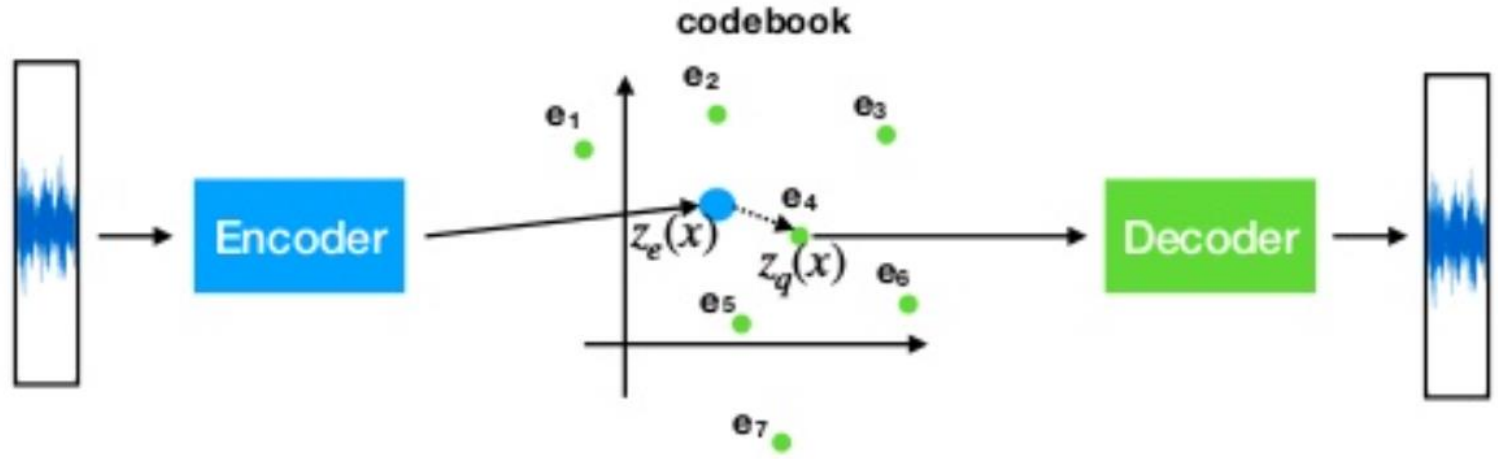
raw audio space는 매우 높은 차원이며 많은 양의 정보를 가지고 있음
-> raw audio를 가지고 모델링 할 경우, 높은 수준의 음악을 학습하는 것이 계산적으로 불가

이러한 어려움을 줄이기 위해 많은 양의 정보를 포함하되 중요하지 않은 정보는 버리는 방향으로 **오디오를 더 낮은 차원으로 인코딩하는 방법을 사용**

**계층 구조의 VQ-VAE를 사용하여 오디오를 discrete space로 압축**
**+ 음악에 대한 정보를 최대한으로 보존하기 위해 설계된 loss function을 사용하여 압축 수준을 높임**

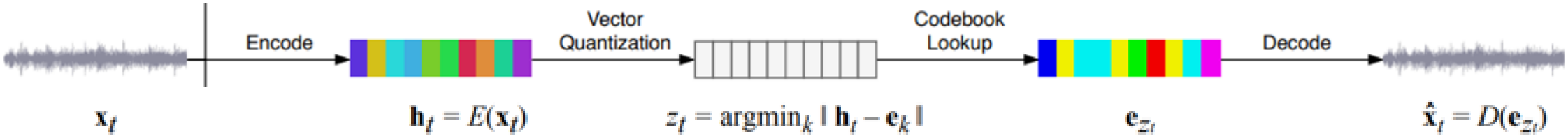
# Background

**VQ-VAE(Vector Quantization Variational Autoencoder)**
: latent space에 자주 등장하는 latent vector들을 미리 뽑아, 이들을 이용하여 데이터를 reconstruction하는 것




- Codebook: latent vector들의 set, 이를 학습하기 위해 loss function 사용

# Background: VQ-VAE




$$\mathcal{L} = \mathcal{L}_{\text{recons}} + \mathcal{L}_{\text{codebook}} + \beta\mathcal{L}_{\text{commit}}$$

$$\mathcal{L}_{\text{recons}} = \frac{1}{T}\sum_t \|\mathbf{x}_t - D(\mathbf{e}_{z_t})\|_2^2$$

(Lrecons : 원신호 x와 decoder를 거친 신호와의 에러를 줄이는 역할)

$$\mathcal{L}_{\text{codebook}} = \frac{1}{S}\sum_s \|\text{sg}\left[\mathbf{h}_s\right] - \mathbf{e}_{z_s}\|_2^2$$

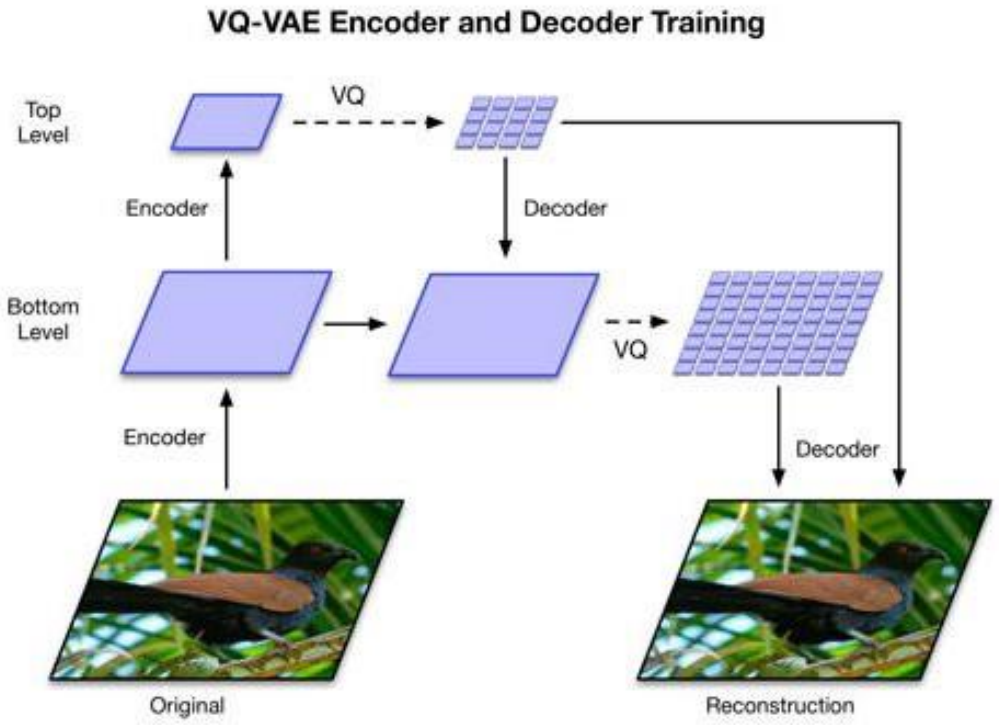
(encoder를 거친 신호와 codebook vector와의 거리를 줄이는 역할)

$$\mathcal{L}_{\text{commit}} = \frac{1}{S}\sum_s \|\mathbf{h}_s - \text{sg}\left[\mathbf{e}_{z_s}\right]\|_2^2$$

(dataset을 이용하여 codebook을 학습하는 과정에서 동일한 신호가 다른 codebook vector로 mapping되는 경우의 빈도를 줄이는 역할,
β= Lcommit의 기여도 조절)

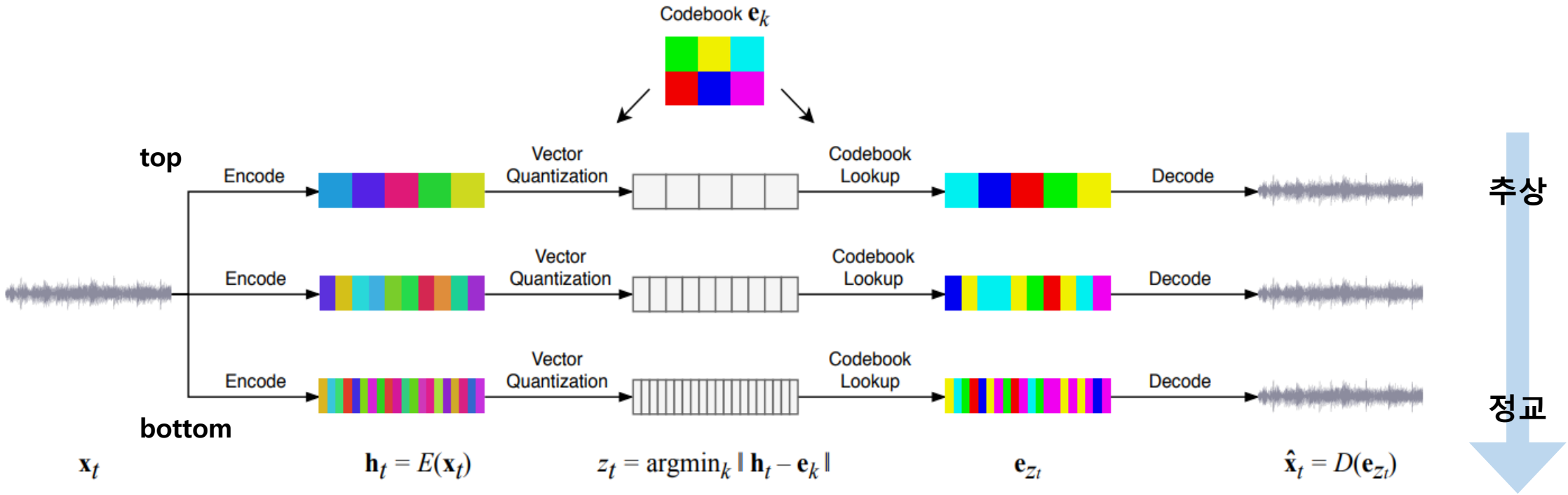
**=> VQ-VAE를 계층 구조로 확장**

# Music VQ-VAE




- single encoder와 decoder로 train
- multi-level의 인코딩 사용
- 이미지에 사용되었던 계층구조의 VQ-VAE를 가져와 audio에 적용

# Music VQ-VAE




- top, middle, bottom 3개의 레벨로 나누어 bottom으로 갈수록 더 잘게 쪼갬
- 각 오디오 세그먼트를 latent vector h로 인코딩
  -> 가장 가까운 코드북 latent vector에 맵핑
  -> decoder reconstruct 진행

# Music VQ-VAE

- 3가지 수정사항 발생

1. **Random restarts for embedding**
   - VQ-VAE에서 codebook collapse 존재
   - 이를 방지하기 위해 codebook vector의 평균 사용량이 임계값 이하로 떨어지면 현재 batch의 encoder output 중 하나로 randomly reset

2. **Separated Autoencoder**
   - 각 레벨에 저장되는 정보의 양을 최대화하기 위해, 다양한 hop의 길이를 가진 seperate autoencoder로 훈련
   - 각 레벨의 discrete code는 독립적인 input의 인코딩으로 간주

# Music VQ-VAE

**3. Spectral Loss**

- simple-level reconstruction loss를 사용하면 낮은 주파수만 reconstruct
- Mid~high 주파수를 잡아내기 위해 spectral loss 사용

$$\mathcal{L}_{\text{spec}} = \left\| |\text{STFT}(\mathbf{x})| - |\text{STFT}(\hat{\mathbf{x}})| \right\|_2$$

- 특정 STFT paramater만 선택함으로써 모델이 오버피팅되는 현상을 막기 위해, 여러 paramater을 가지고 계산된 spectral loss의 합 사용

# Music Priors and Upsamplers

$$p(\mathbf{z}) = p(\mathbf{z}^{\text{top}}, \mathbf{z}^{\text{middle}}, \mathbf{z}^{\text{bottom}})$$

$$= p(\mathbf{z}^{\text{top}})\, p(\mathbf{z}^{\text{middle}} | \mathbf{z}^{\text{top}})\, p(\mathbf{z}^{\text{bottom}} | \mathbf{z}^{\text{middle}}, \mathbf{z}^{\text{top}})$$

upsampler

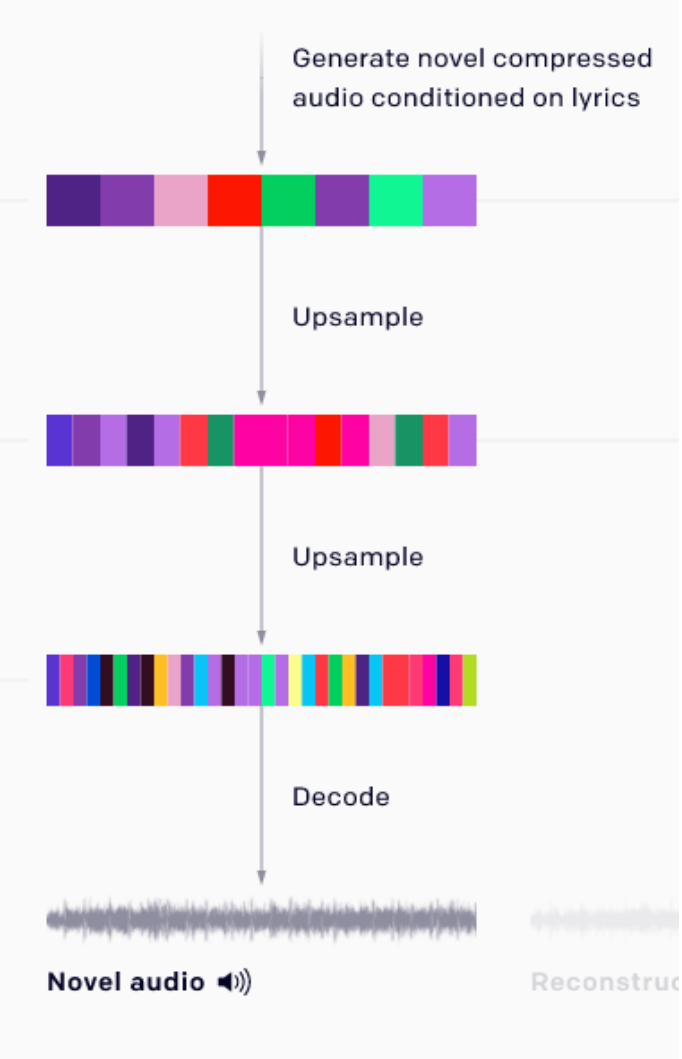
top level prior

input

produce

- VQ-VAE 훈련 후, 샘플을 생성하기 위해 압축된 공간에 대해 prior p(z)를 계산

# Music Priors and Upsamplers




- scalable transformer 사용
- upsample을 하기 위해 상위 레벨에서 사용한 conditioning info를 transformer에게 제공
- 학습 중 추가적인 conditioning info를 제공함으로써 생성모델을 더 controllable하게 만들 수 있음

# Music Priors and Upsamplers

- 3가지 conditioning 제공

1. **Artist, Genre and Timing Conditioning**
   - 아티스트와 장르 제공: 모델이 어느 특정 스타일에서 더 좋은 퀄리티 도달할 수 있음. 또한 생성 시 선택한 스타일에 따라 모델 조절 가능
   - timing signal(곡의 총 길이, 특정 샘플의 시작시간 등)을 train 시에 각 segment에 추가 -> 노래를 부르는 부분 또는 악기가 시작되는 부분과 끝과 같은 정보를 통해 전반적인 구조 학습 가능

# Music Priors and Upsamplers

## 2. Lyrics Conditioning

- 앞의 conditioned 모델은 인식 가능한 영어 단어 생성 불가
- 가사를 생성하기 위해 각 오디오 세그먼트에 해당하는 가사를 conditioning으로 제공
- **Lyrics-to-siging(LTS) task**
  : conditioning signal에 가사만 포함, 코러스와 같이 반복되는 섹션, 리드보컬 등의 오디오간 구별이 없기 때문에 정교하지 않음
- **Providing lyrics for chunks of audio**
  : 노래의 특정 세그먼트에 해당하는 텍스트를 분석해 오디오 길이 만큼으로 가사를 늘려서 사용하려고 시도 했으나, 힙합과 같은 빠른 가사에서는 오류가 발생. 이를 해결하기 위해 보컬을 추출하여 텍스트 배열로 가사에 대한 단어 수준 분석 획득.
- **Encoder-Decoder model**
  : 가사들의 특징을 조건으로 사용하기 위해 encoder-decoder사용. encoder는 decoder에 의해 처리되는 가사 들로부터 특징 생산. decoder는 top level music token을 생성.

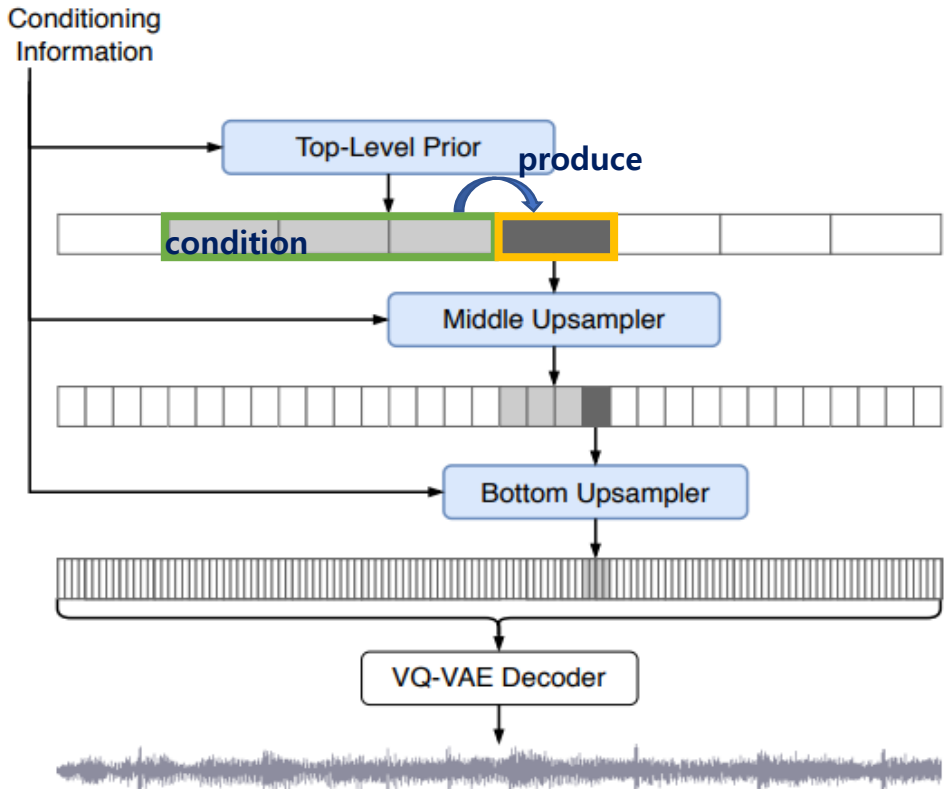
# Music Priors and Upsamplers

## 3. Decoder Pretraining

- lyrics conditional model 훈련에 요구된 계산을 줄이기 위해 사용
- 초기에 모델은 pretrained decoder과 동일하게 동작하지만
  인코더 state와 paramater에 변화가 감지되면, encoder 사용법을 학습할 수 있게 허용

# Sampling

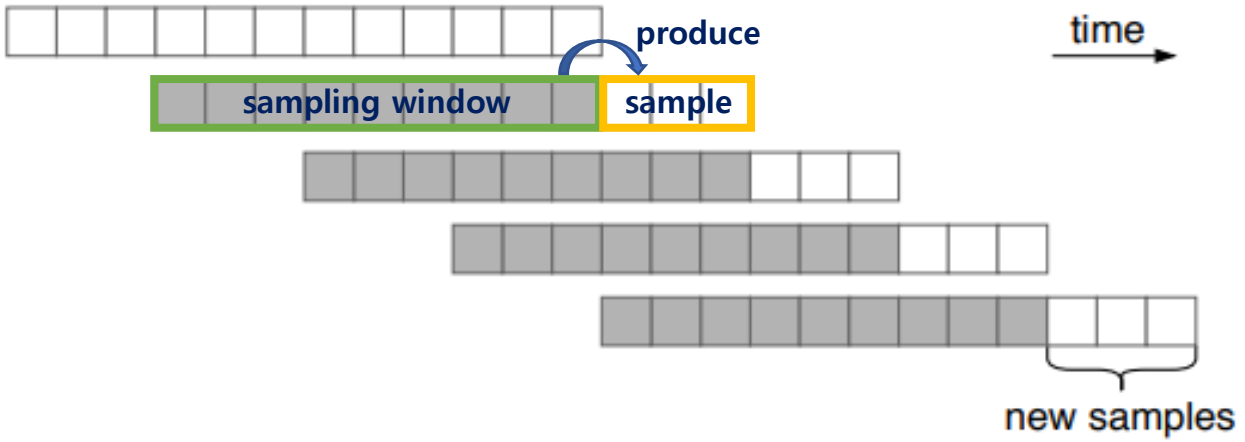
VQ-VAE, upsampler, top level prior 학습 후 sampling 수행




## 1. Ancestral Sampling

- top level code를 한번에 한 token 씩 생성
- 첫번째 token 생성 후, 이전에 생성된 모든 토큰들을 모델의 input으로 보냄
- top level code에서 mid level의 conditioning info 생성, 이 과정을 bottom까지 반복

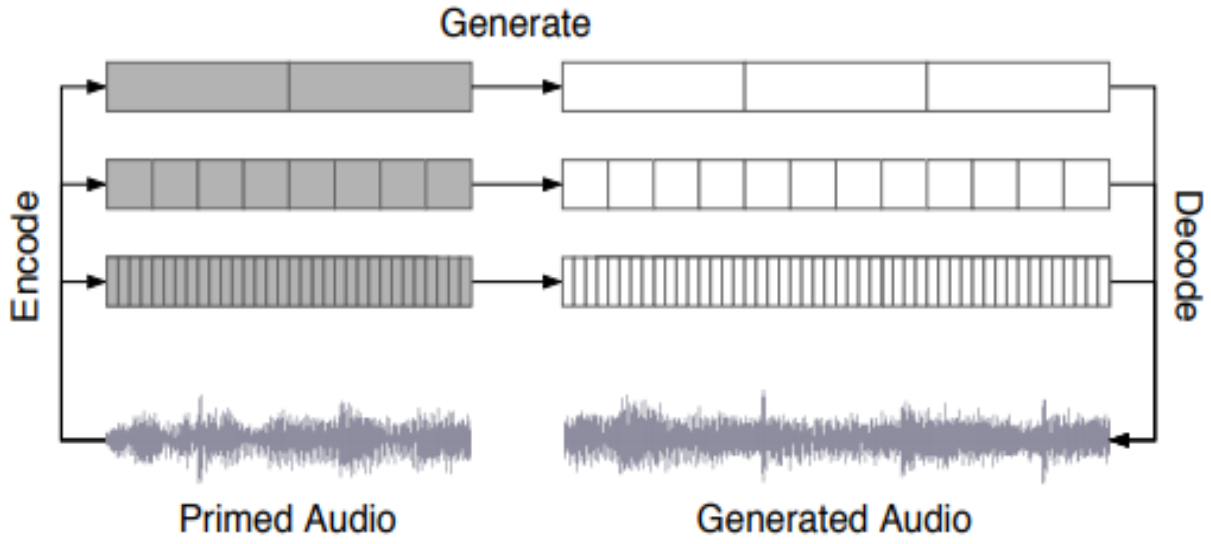
# Sampling




## 2. Windowed Sampling

- segment를 context 길이보다 길게 샘플링하기 위해 사용
- sampling window를 사용하여 샘플 생성

# Sampling




## 3. Primed Sampling

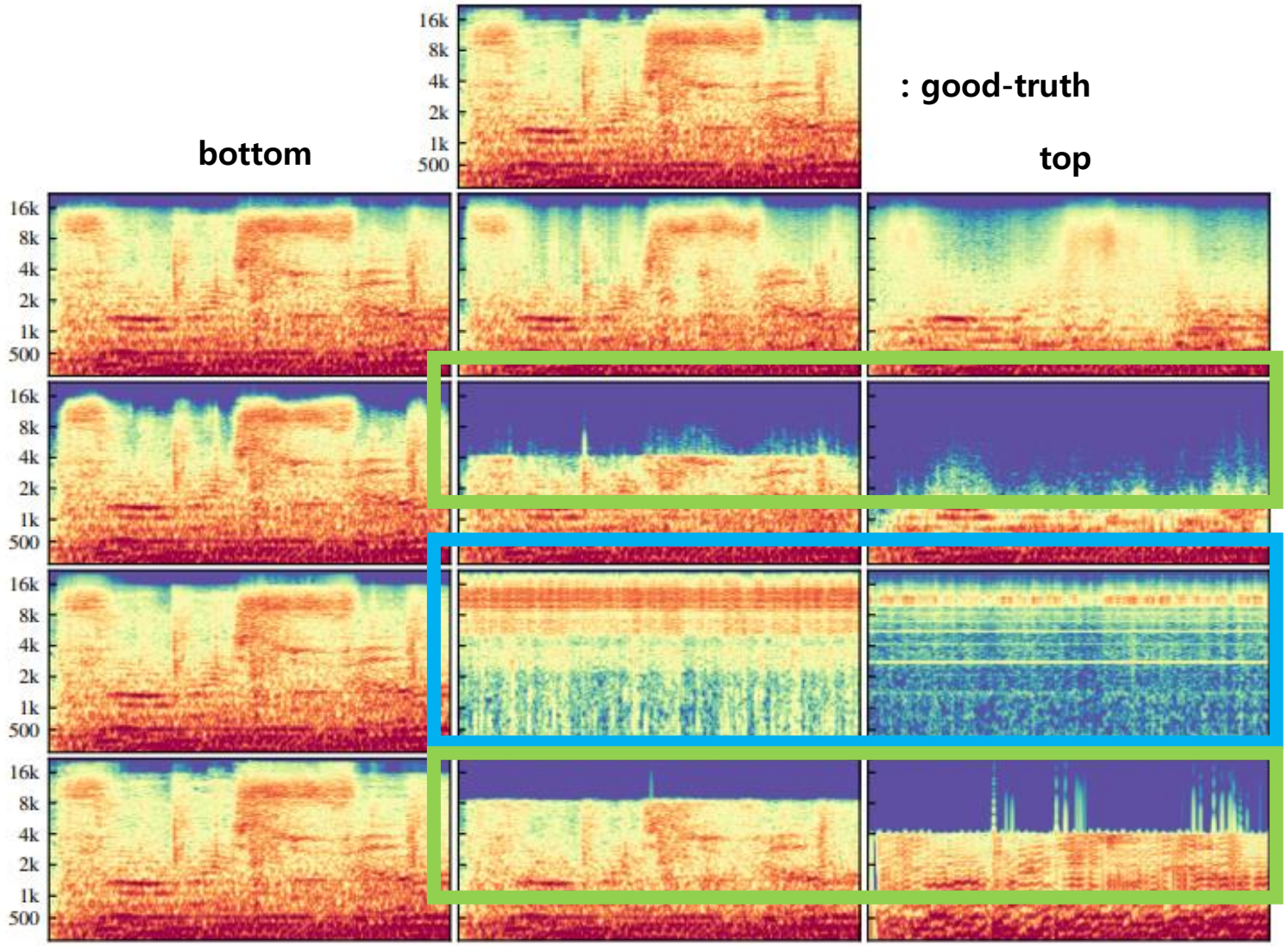
- 전체 token 샘플링 대신 실제 노래의 세그먼트에 대응되는 오디오에 대해서만 샘플링 진행
- Ancestral Sampling 과정에서 initial token으로 사용

# Experiments

- 120만개의 노래로 이루어진 데이터 셋으로 훈련 진행
- 데이터 셋의 곡에는 가사와 아티스트, 앨범, 발매년도, 장르, 키워드 등의 메타 데이터가 크롤링되어 연결되어 있음
- 32 bit, 44.1kHz raw audio로 훈련, 무작위로 다운믹스하여 모노 오디오를 만들어 데이터 증강 수행
- 8, 32, 128의 hop 길이를 가지도록 오디오를 쪼갬
  각 레벨의 codebook 사이즈는 2048

# Experiments

각기 다른 조건에서 수행한 reconstruction




3가지 수정 사항을 거친 계층구조의 VQ-VAE 사용

Spectral Loss 적용X

수정하지 않은 계층구조의 VQ-VAE 사용

Opus codec 사용

# Conclusion

- Multiple minutes의 곡을 만들어 낼 수 있고, 목소리로 인식되는 노래 생성
- 압축

  : 원곡을 8배, 32배, 128배 3가지 비율로 인코딩하여 3종류의 학습데이터 구축
- 학습

  : 3개의 학습 모델이 top level로 부터 곡의 전반적인 구조를, bottom level로부터 음색, 분위기 등 세부적인 내용을 분업하여 학습
- 생성

  : 3개의 모델이 각자 학습한 내용을 활용하여 음악을 생성(decoding)하고, 이를 조합하여 최종 음악 완성

# Thank you