Jukebox: A Generative Model for Music

Prafulla Dhariwal et al., 2020

吉永 墾

September 30, 2020

概要

- 内容：音楽生成
- 特徴：
  1. raw audio domain で歌を伴う音楽を生成できる
  2. 数分の長さの楽曲を生成することができる
  3. 生成時に楽曲スタイルを条件指定できる

- 実際に生成された楽曲 `https://jukebox.openai.com/`

先行研究

- 生成モデル
  - 文章生成
  - 音声生成
  - 画像生成

  近年急速に発展

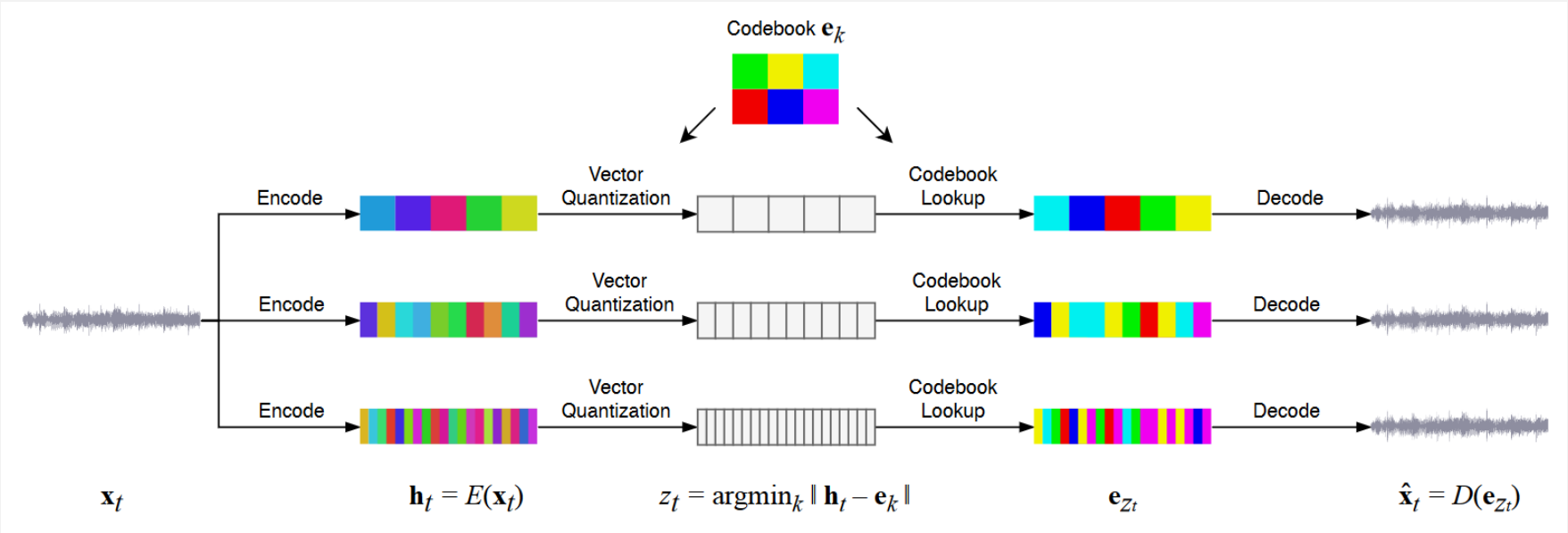
- 音楽生成モデル
  - symbolic なアプローチ
    - 低次元空間上なのでモデリングが簡単
    - 生成できる音楽に制約がかかる（例：楽器の音が固定）
  - non-symbolic なアプローチ
    - 音楽を audio として直接生成
    - raw audio は高次元で情報量が多い
    - 長期依存性 (long-range dependency) により音楽の高度な意味 (semantics) の学習が難しい

提案手法

重要度の低い情報を捨て，音楽の大部分の情報を保持するように圧縮

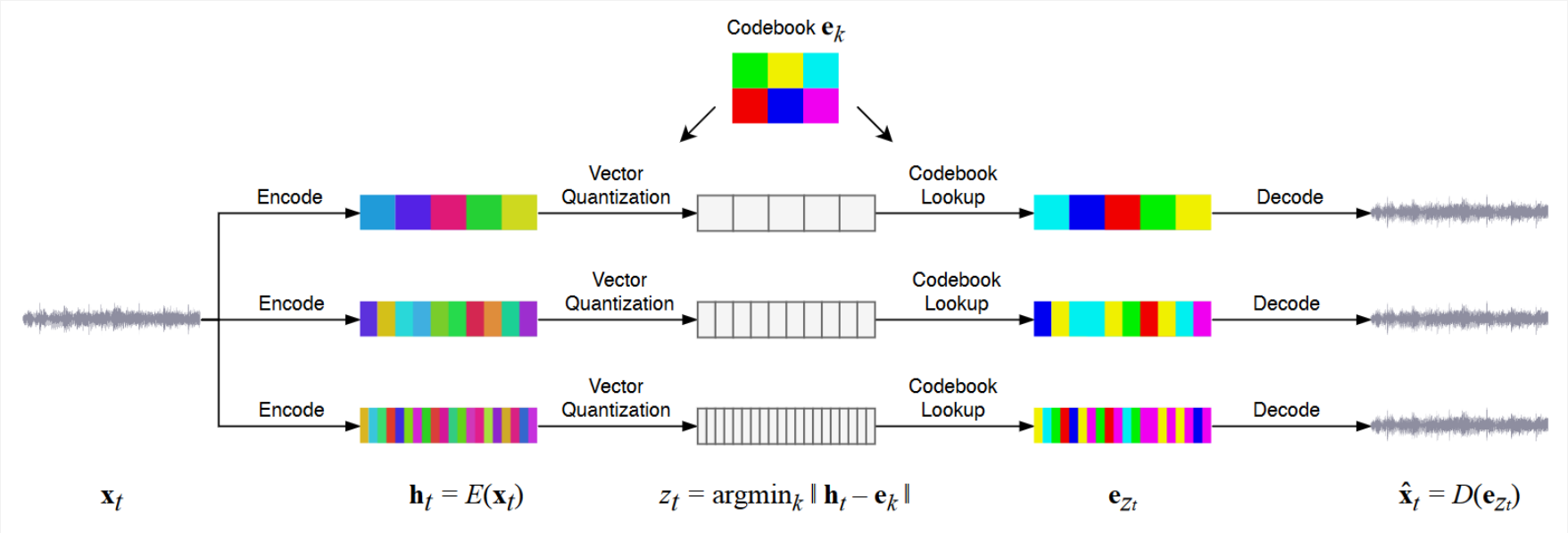
- Hierarchical VQ-VAE の構造を用いて audio を離散空間へ圧縮
- prior モデルによって離散空間上で code を生成
- 生成された code を VQ-VAE(Decoder) によって再構成

Music VQ-VAE



- hierarchical VQ-VAE を使用（[1][2] で提案されたものを修正）
- Encoder/Decoder は複数の CNN で構成
- 学習の対象：Encoder, Decoder, Codebook

Music VQ-VAE

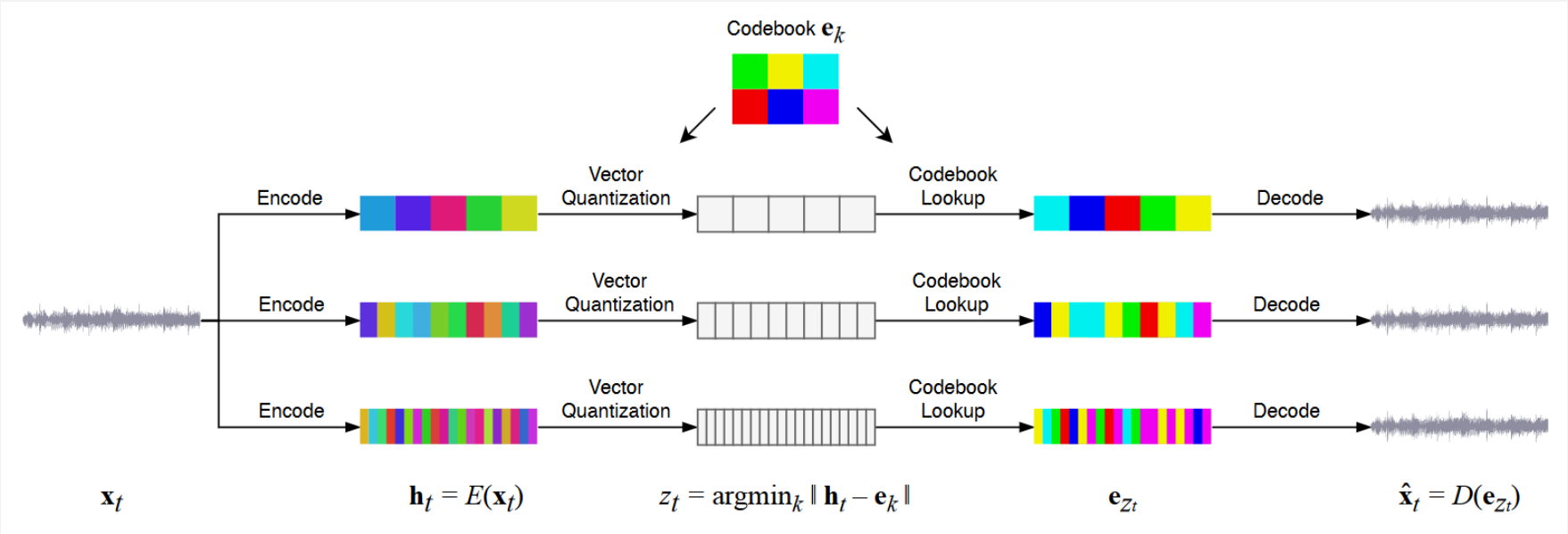


Encode の流れ

1. 入力 $\boldsymbol{x}_t$ を Encode して潜在ベクトル $\boldsymbol{h}_t = E(\boldsymbol{x}_t)$ へ
2. $\boldsymbol{h}_t$ をベクトル量子化して $z_t = \text{argmin}_k \|\boldsymbol{h}_t - \boldsymbol{e}_k\|$ へ

Codebook が持つ要素 $\boldsymbol{e}$ のうち $\boldsymbol{h}$ に一番近いもの $(\boldsymbol{e}_{z_t})$ を見つける
ここで現れる z が楽曲の離散表現（z は code と呼ばれる）

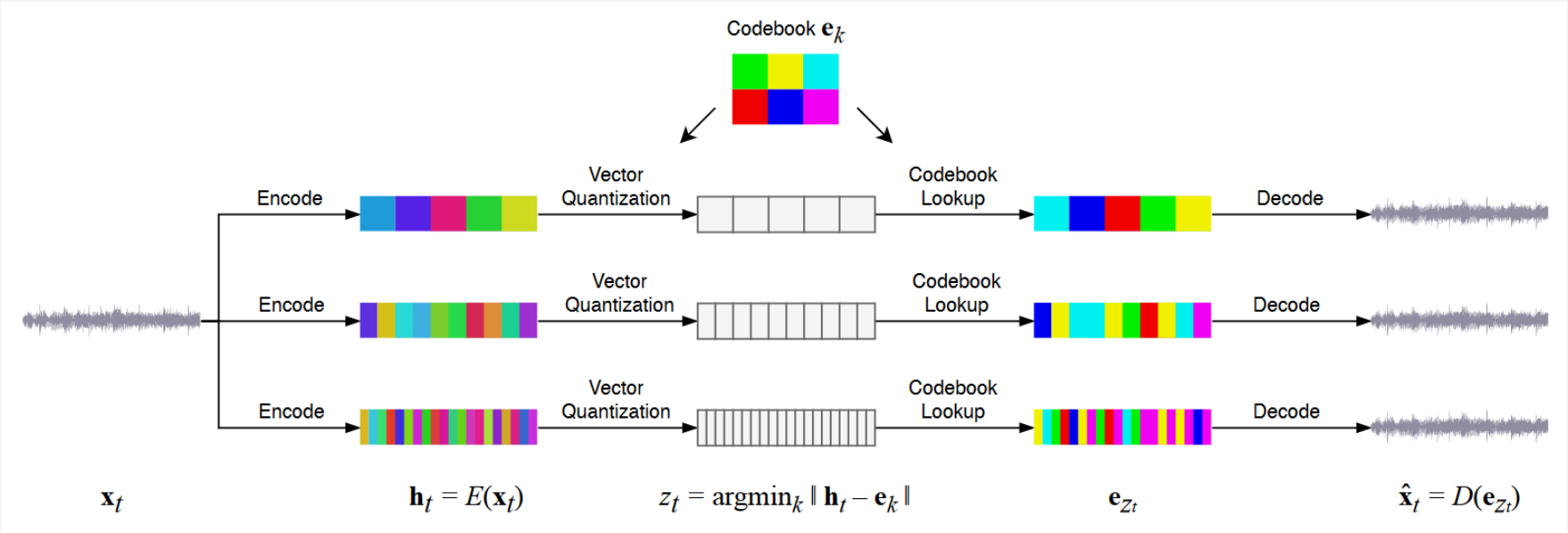
Music VQ-VAE



Decode の流れ

1. codebook を参照して code z_t をベクトル $\boldsymbol{e}_{z_t}$ へ
2. $\boldsymbol{e}_{z_t}$ を Decode して入力を復元 $\hat{\boldsymbol{x}}_t = D(\boldsymbol{e}_{z_t})$

Music VQ-VAE



- Top/Middle/Bottom で圧縮率を変える
- 上にあがるにつれて抽象度は高くなる
- 抽象度によってより local/global な情報を捉える

Music VQ-VAE

Objective

$$\mathcal{L} = \mathcal{L}_{\text{recons}} + \mathcal{L}_{\text{codebook}} + \beta \mathcal{L}_{\text{commit}} \tag{1}$$

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

$$\mathcal{L}_{\text{codebook}} = \frac{1}{S} \sum_{s} \|\text{sg}\left[\boldsymbol{h}_s\right] - \boldsymbol{e}_{z_t}\|_2^2 \tag{3}$$

$$\mathcal{L}_{\text{commit}} = \frac{1}{S} \sum_{s} \|\boldsymbol{h}_s - \text{sg}\left[\boldsymbol{e}_{z_s}\right]\|_2^2 \tag{4}$$

（記号）sg (stop-gradient operation)：逆伝播のとき勾配を 0 にする

各項の役割

(2) 入力 $\boldsymbol{x}$ と出力 $\hat{\boldsymbol{x}} = D(\boldsymbol{e}_z)$ を近づける

(3) codebook が持つ要素 $\boldsymbol{e}_z$ と潜在ベクトル $\boldsymbol{h}$ を近づける

(4) 潜在ベクトル $\boldsymbol{h}$ が動きすぎるのを防ぐ ($\beta = 0.02$ は重み)

Music VQ-VAE

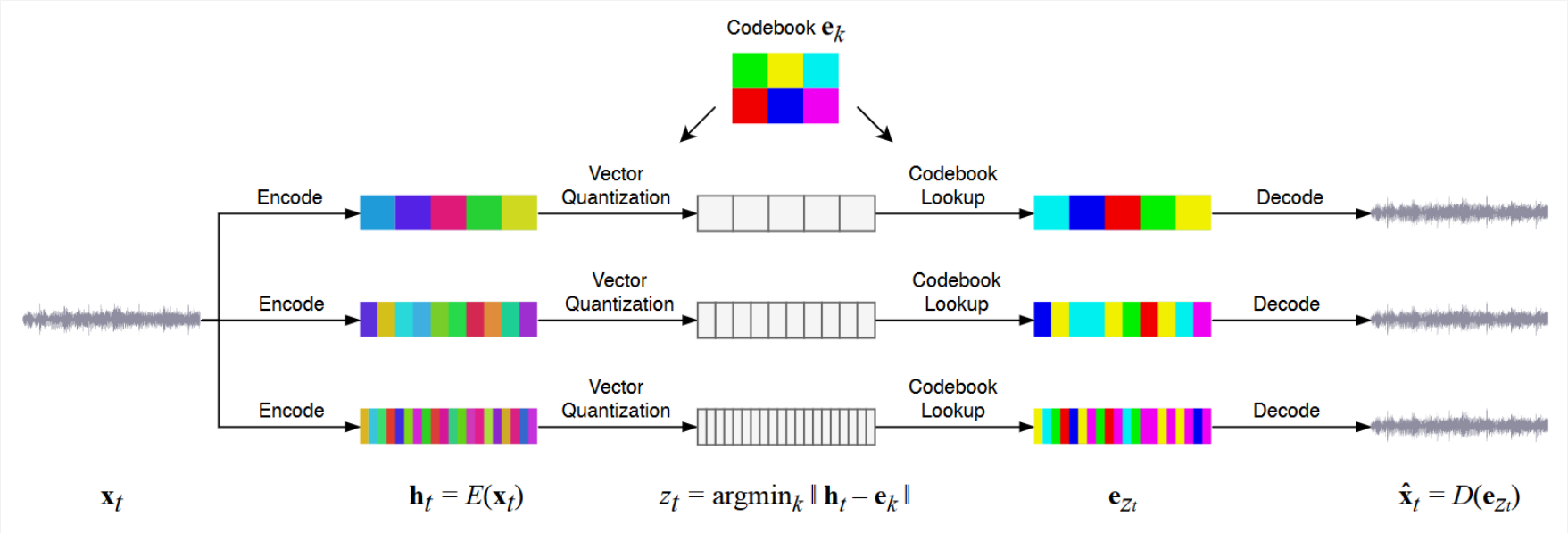
codebook の更新

Exponential Moving Average ([2] より)

- $\{z_{i,1}, \ldots, z_{i,n_i}\}$: $\boldsymbol{e}_i$ に最も近い n_i 個の Encoder の出力から成る集合
- $\gamma \in [0, 1]$: decay parameter ($\gamma = 0.99$)

$$N_i^{(t)} := N_i^{(t-1)} * \gamma + n_i^{(t)}(1 - \gamma)$$
$$m_i^{(t)} := m_i^{(t-1)} * \gamma + \sum_j z_{i,j}^{(t)}(1 - \gamma)$$
$$e_i^{(t)} := \frac{m_i^{(t)}}{N_i^{(t)}}$$

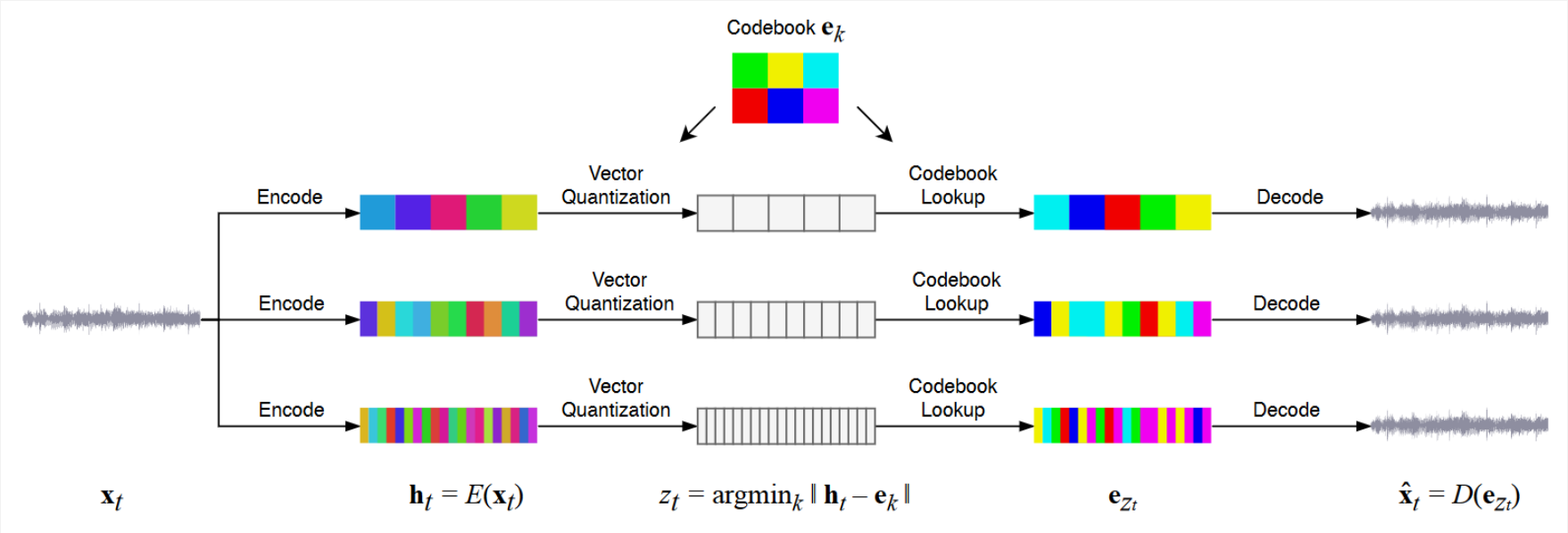
Music VQ-VAE



(1) Random restarts for embeddings

- 問題点：codebook collapse
  Codebook 内の一部のベクトルしか使われない
- 解決策：random restart
  codebook 内のベクトルの平均使用量が閾値以下になると，そのベクトルを random に選んだ Encoder の出力の一つに reset

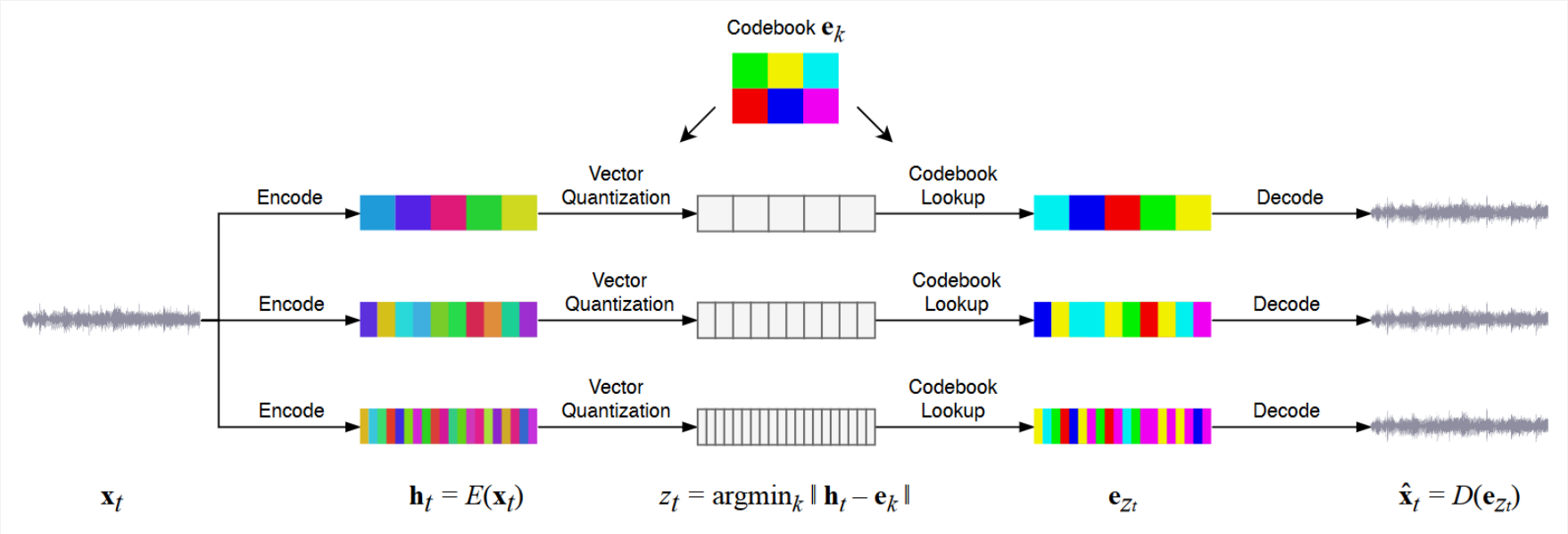
Music VQ-VAE



(2) Separated Autoencoders

- 問題点：a complete collapse
  top level がほとんど使われない
- 解決策：各レベルの Encoder を別々に学習

Music VQ-VAE



(3) Spectral loss

- 問題点：モデルが低周波数のみを学習
- 解決策：Spactral loss の導入

$$\mathcal{L}_{\text{spec}} = \| \, |\text{STFT}(\boldsymbol{x})| - |\text{STFT}(\hat{\boldsymbol{x}})| \, \|_2$$

Music Prior and Upsamplers

(VQ-VAE の学習後)

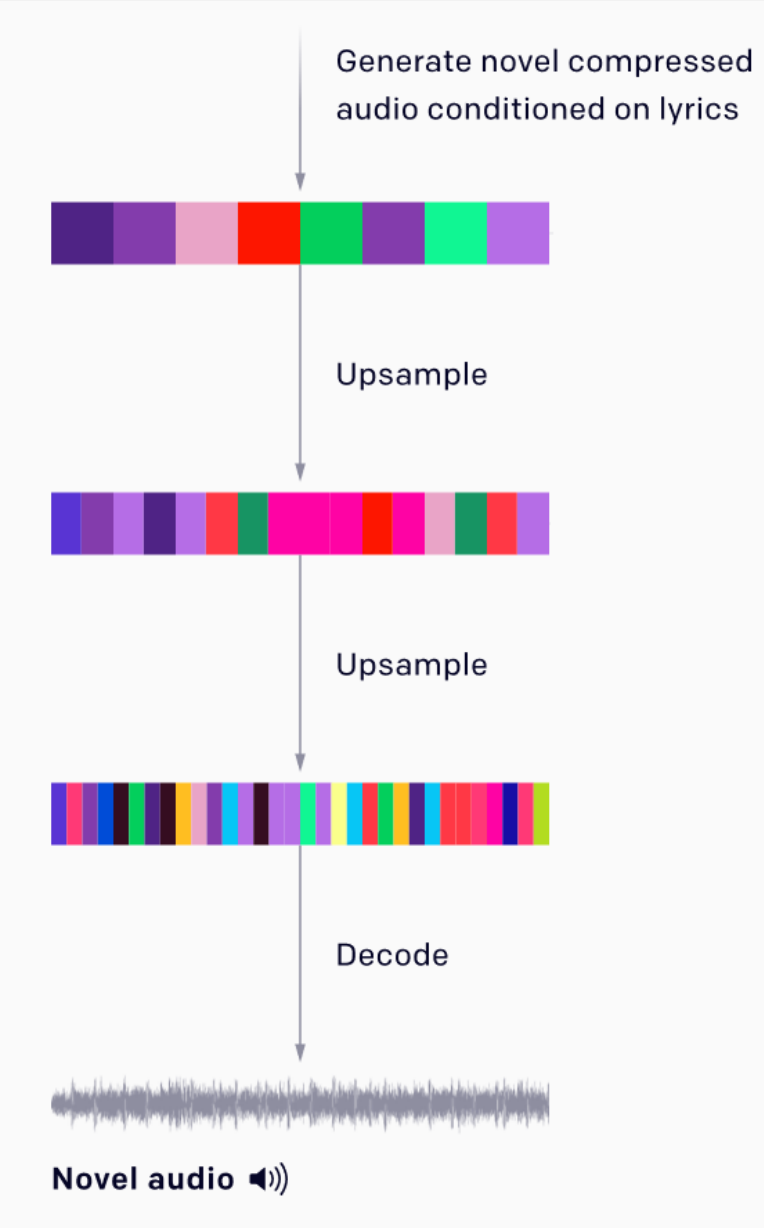
prior $p(\boldsymbol{z})$ をモデル化

$$p(\boldsymbol{z}) = p(\boldsymbol{z}^{\text{top}}, \boldsymbol{z}^{\text{middle}}, \boldsymbol{z}^{\text{bottom}}) \tag{5}$$

$$= p(\boldsymbol{z}^{\text{top}}) p(\boldsymbol{z}^{\text{middle}} | \boldsymbol{z}^{\text{top}}) p(\boldsymbol{z}^{\text{bottom}} | \boldsymbol{z}^{\text{middle}}, \boldsymbol{z}^{\text{top}}) \tag{6}$$

- $p(\boldsymbol{z}^{\text{top}})$: Top-Level Prior
  - 条件より $\boldsymbol{z}^{\text{top}}$ を生成
- $p(\boldsymbol{z}^{\text{middle}} | \boldsymbol{z}^{\text{top}})$: Middle Upsampler
  - 条件と $\boldsymbol{z}^{\text{top}}$ より $\boldsymbol{z}^{\text{middle}}$ を生成
- $p(\boldsymbol{z}^{\text{bottom}} | \boldsymbol{z}^{\text{middle}}, \boldsymbol{z}^{\text{top}})$: Bottom Upsampler
  - 条件と $\boldsymbol{z}^{\text{middle}}$ より $\boldsymbol{z}^{\text{bottom}}$ を生成

Music Prior and Upsamplers



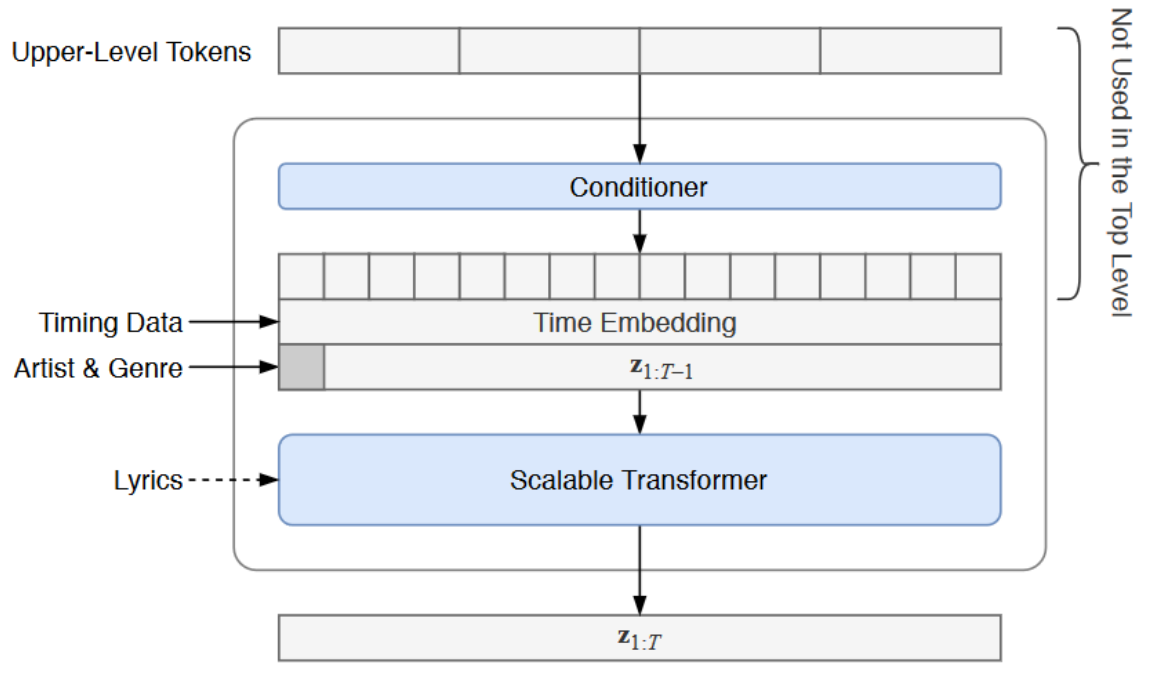
Sample の流れ

1. Lyrics 条件をもとにして Top-Level Prior が z^{top} を生成
2. 1 で生成された z^{top} と条件より z^{middle} へ upsample
3. 2 で生成された z^{middle} と条件より z^{bottom} へ upsample
4. 3 で生成された z^{bottom} を VQ-VAE の Encoder に渡す

Top-Level Prior / Upsamplers は Transformer を基本とした構造

画像：https://openai.com/blog/jukebox/

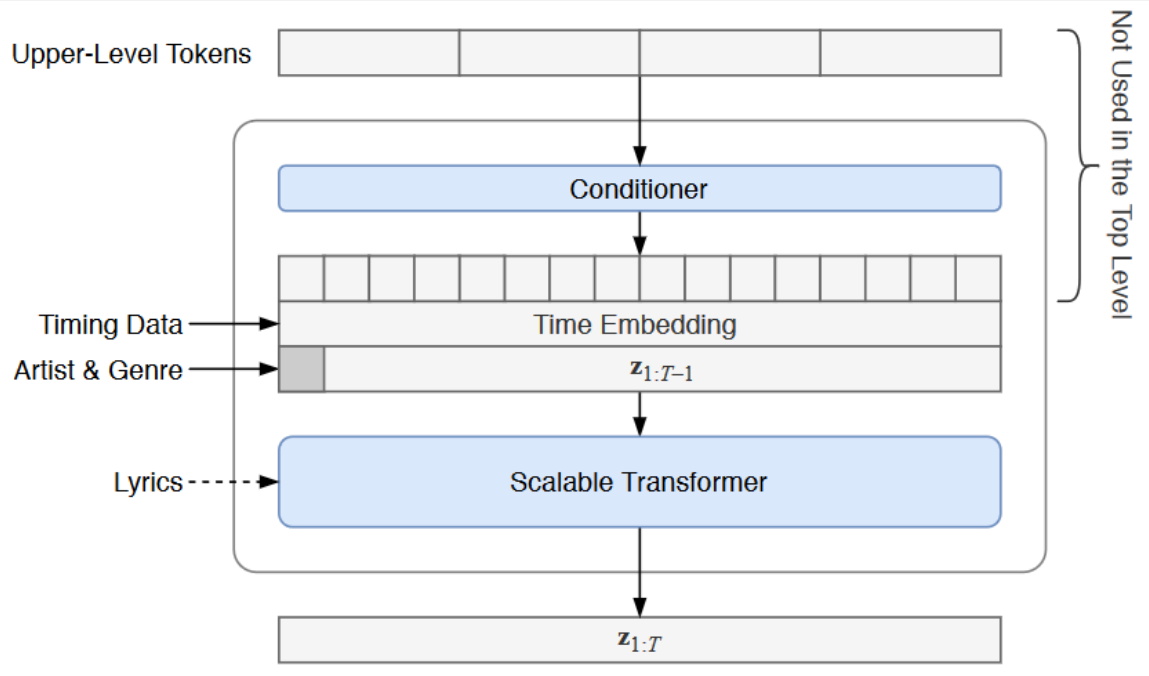
Music Prior and Upsamplers



Top-Level Prior の構造

- Scalable Transformer(Transformer の簡易版) を使用
- Artist, Genre, Timing Conditioning を Embedding として与える
- Lyrics Conditioning を与える
- 既に生成したコード ($z_{1:T-1}$) より次のコード (z_T) を生成

Music Prior and Upsamplers



Upsampler の構造

- Scalable Transformer(Transformer の簡易版) を使用
- Artist, Genre, Timing Conditioning を Embedding として与える
- 上位レベルの token を Conditioner で変換，条件として与える
- 既に生成したコード ($z_{1:T-1}$) より次のコード (z_T) を生成

Music Prior and Upsamplers

Top-Level Prior / Upsampler の学習

- 最尤推定 (?)
- We use an autoregressive Sparse Transformer trained with maximum-likelihood estimation over this compressed space,and also train autoregressive upsamplers to recreate the lost information at each level of compression.

- Conditioner は CNN で構成 (学習方法 ?)

生成楽曲

生成楽曲：`https://jukebox.openai.com/`

manually に評価

- 生成された楽曲は一貫性があり音楽的
- 長期的な構造（コーラスやメロディの繰り返し）は持たない
- 歌詞：自然，メロディ：人間の作曲よりは面白くない
- モデルは幅広い歌唱スタイルや演奏スタイルを学習
- 面白く多様性に富んでいるが、全体的には原曲に比べてまだまだ音楽性に改善の余地がある

Future Work

- 音楽的構造
  - local な音やハーモニーが常に意味を成す
  - ニュアンスのある適当な楽器の強弱
  - 録音バランス，ノイズ
  - 繰り返されるコーラスやメロディの応答などの長期の音楽的構造
- 言語・スタイルの多様性
  - このモデルは歌詞が英語の曲で学習
  - 他の言語やアーティストを含める
  - 創造性や発展は現存のスタイルの普通ではないな融合によって起こる
- 音楽制作ツールとしての応用 [重要]
  - mood/dynamic/楽器の演奏タイミングでの条件付け
  - 楽曲生成にかかる時間：1 分の楽曲生成に 9 時間程度
  - top-level token を生成した段階で decode，人間が概要を聞いて気に入ったものを生成

参考文献

[1] Ali Razavi, Aaron van den Oord, and Oriol Vinyals.
Generating Diverse High-Fidelity Images with VQ-VAE-2, 2019.

[2] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu.
Neural Discrete Representation Learning, 2017.

Training Details

- Music VQ-VAE
  - audio を 8x, 32x, 128x に圧縮
  - codebook のサイズ：各レベル 2048
  - パラメータ数：2 million
  - 44.1kHz，9 秒の audio clip：256V100 で 3 日
- Prior / Upsampler
  - パラメータ数：5 billion(Top-Level Prior)，1 billion(Upsamplers)
  - VQ-VAE の token 8192 個（audio 換算で 24，6，1.5 秒）：128V100s で 4 週間，128V100s で 2 週間
  - Lyrics conditioning 用の Encoder 追加後：512V100s で 2 週間

Music VQ-VAE

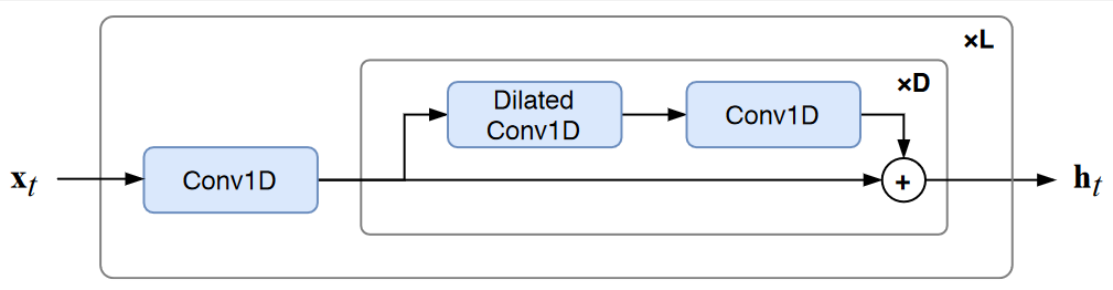


Figure: Encoder

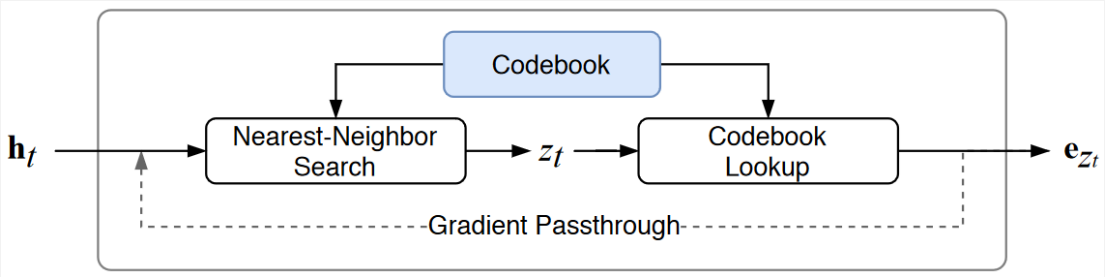


Figure: Bottleneck

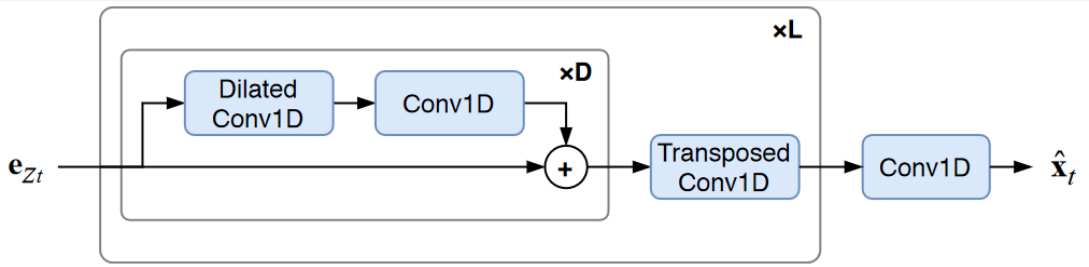


Figure: Decoder

Music Prior and Upsamplers

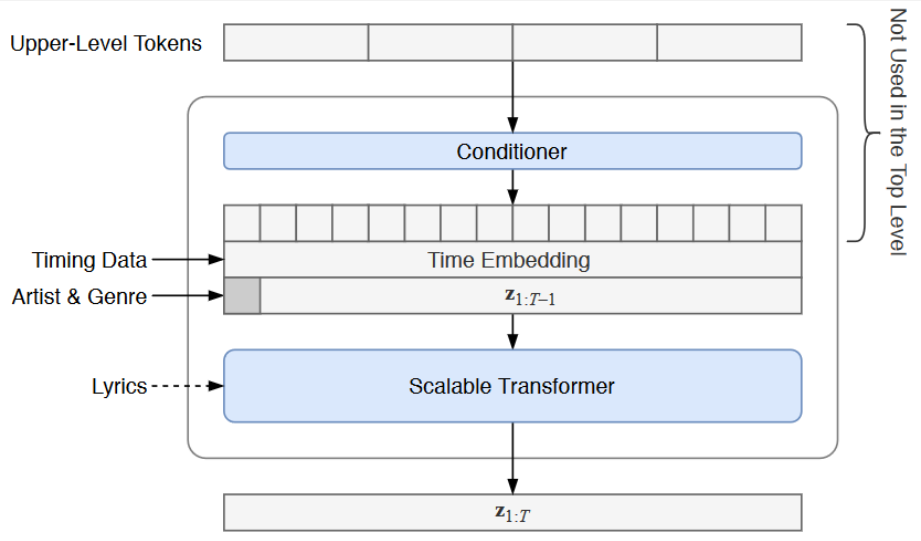


Figure: Top-Level Prior / Upsamplers

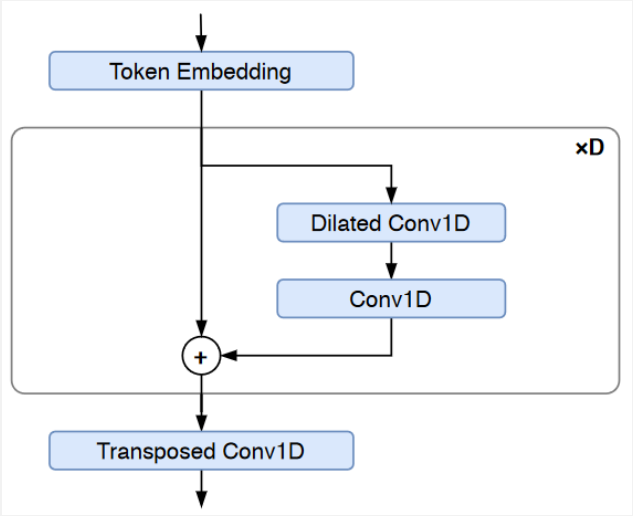


Figure: Conditioner

Music Prior and Upsamplers

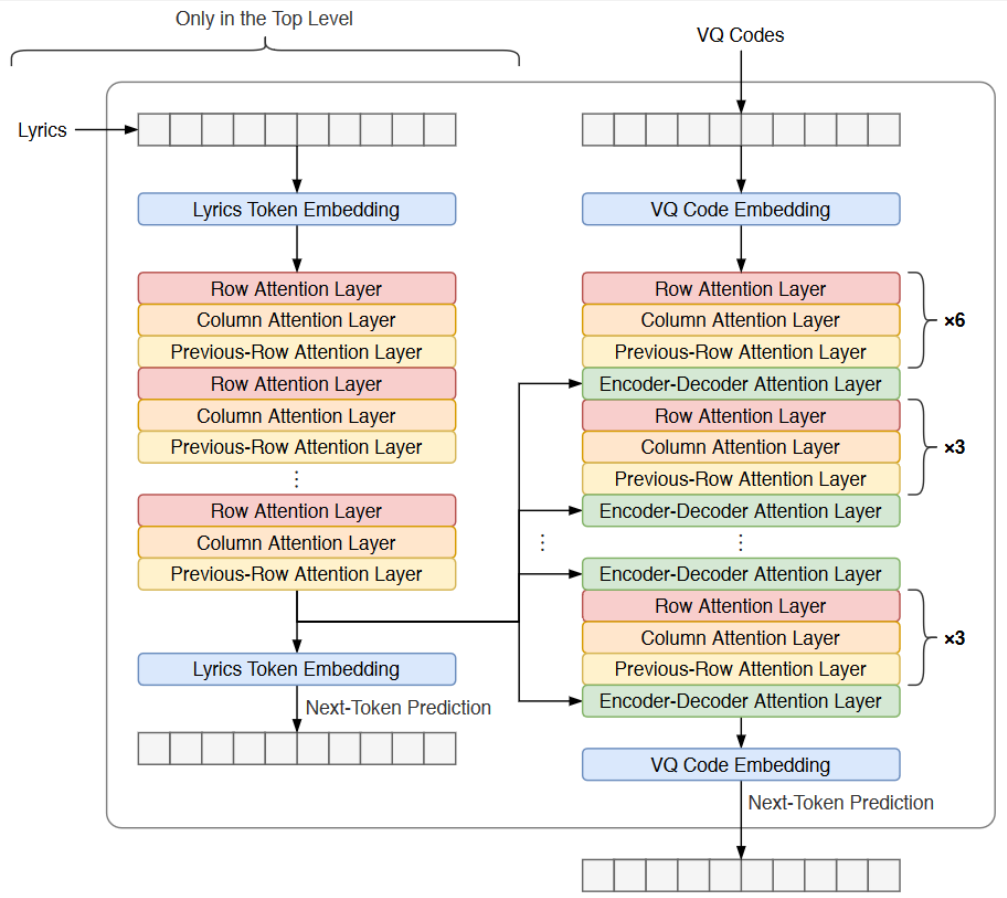


Figure: Scalable Transformer

Scalable Transformer の構造

(Top-Level)

- encoder-decoder 形式
- decoder 部は Top-Level Prior を再利用
- encoder 部に model surgery を使用し decoder 部と合わせる
- encoder は Lyrics の表現を decoder 部に渡す

(Upsamplers)

- decoder 部のみ

VQ-VAE Ablations

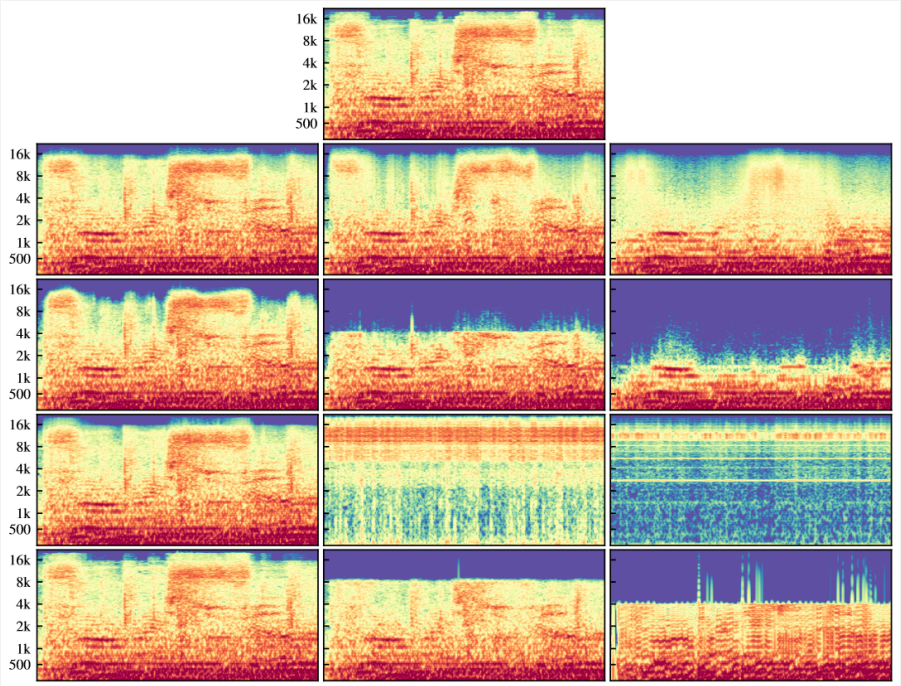


Figure: VQ-VAE 再構成比較

軸: x: 時間 (time), y: 周波数 (frequency)

列: 左から順に bottom, middle, top-level の再構成

行: 1: ground truth, 2: 提案手法, 3: 提案手法 (spectral loss なし),
4: VQ-VAE(Separate なし), 5: baseline 手法 (Opus codec)