2023 November 3

Deep Learning for Music Analysis and Generation

Text-to-music Generation

(text → audio)

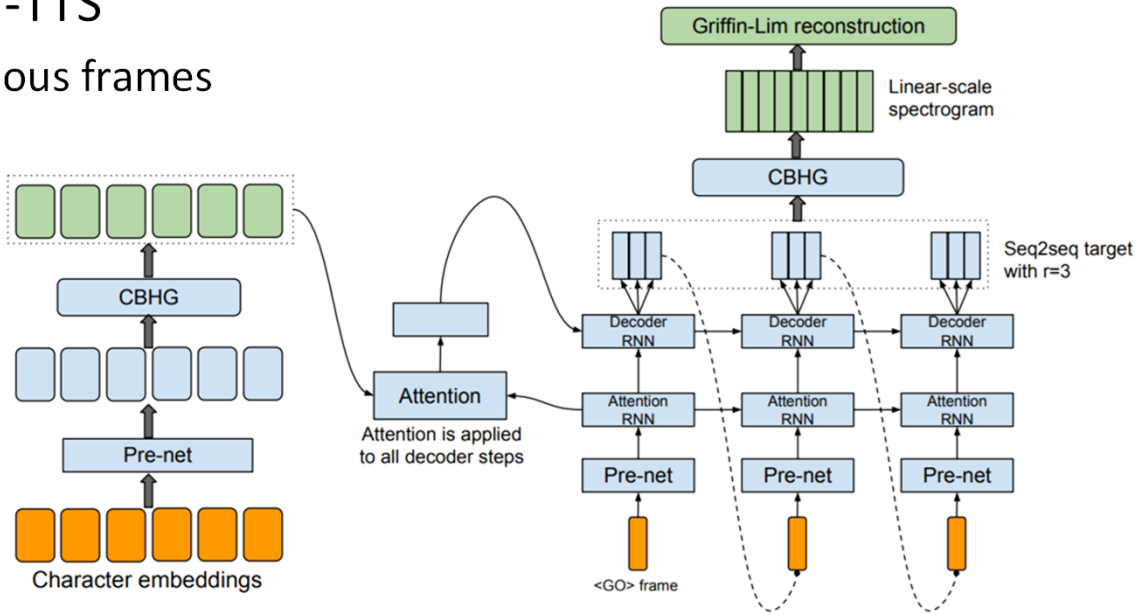
Yi-Hsuan Yang Ph.D.
yhyangtw@ntu.edu.tw

Outline

- **Image/audio tokenization and generation via VQVAE**
- Audio codec models
- Text-to-music: linking text and musical audio
  - Transformer-based approach
  - Diffusion-based approach

Building an LM for Audio

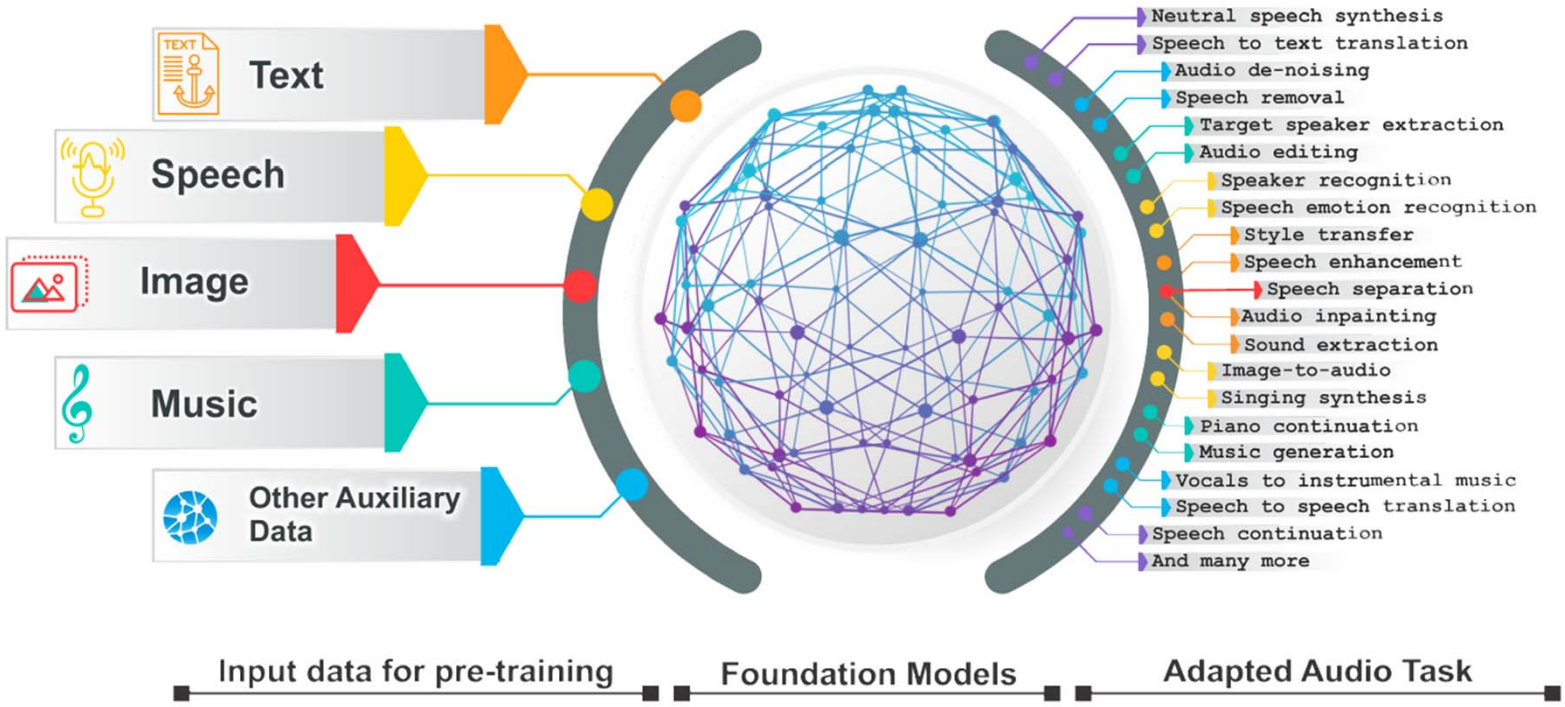
- MIDI events are by nature **discrete**; easy to build an LM for MIDI
- Audio files (waveforms or spectrograms) are **continuous** so it's trickier
- We can still build an autoregressive model for audio
  - Example: TacoTron or Transformer-TTS
    - Predict the next frame given the previous frames



(Figure from Wang et al, "Tacotron: Towards end-to-end speech synthesis," INTERSPEECH 2017)

Building an LM for Audio

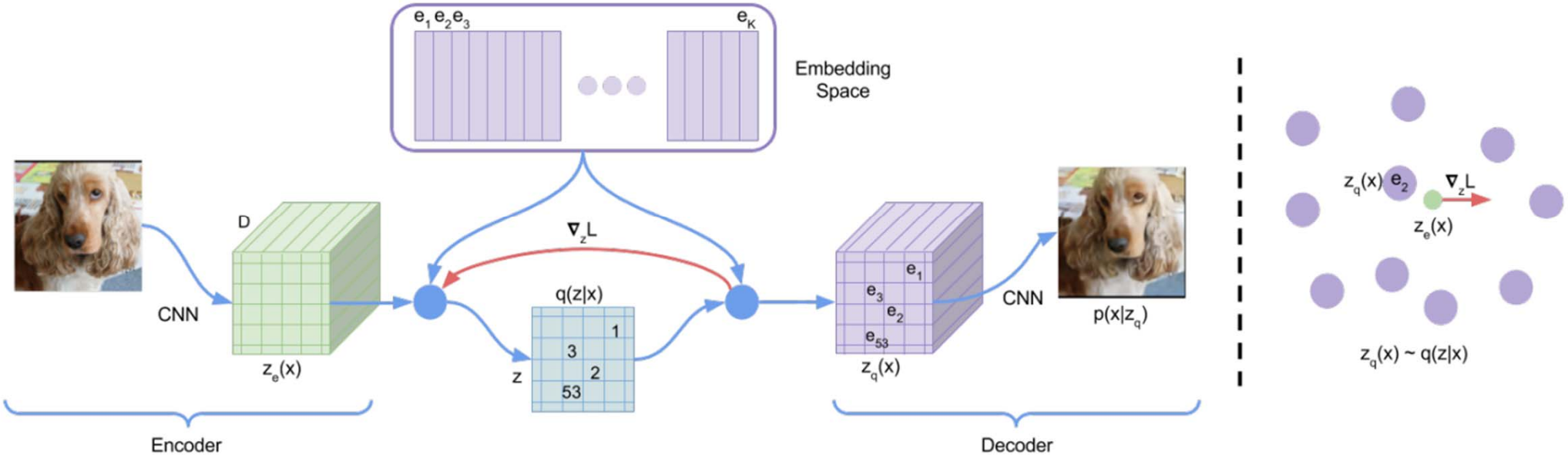
- But, easier to build a multimodal LM if *everything* is **discrete**



(Figure from Latif et al, "Sparks of large audio models: A survey and outlook," arXiv 2023)

Image Tokenization by VQVAE

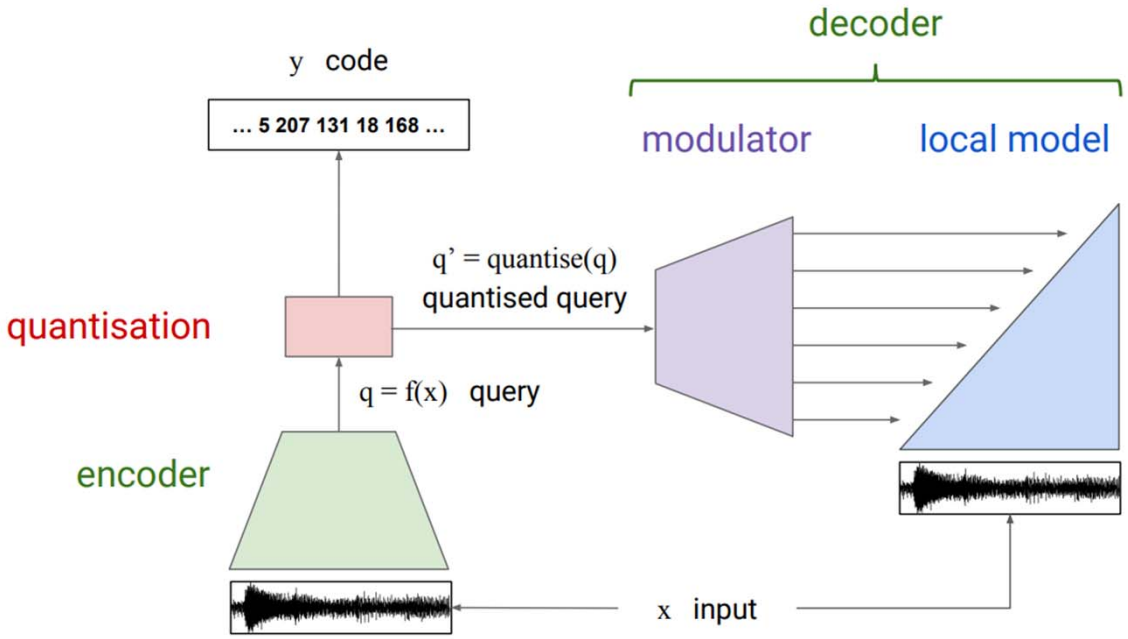
- We can obtain **discrete tokens of continuous data** by doing **vector quantization** at the output of an image **encoder** (Oord et al., 2017)



Ref: Oord et al, "Neural discrete representation learning, NeurIPS 2017

Audio Tokenization by VQVAE

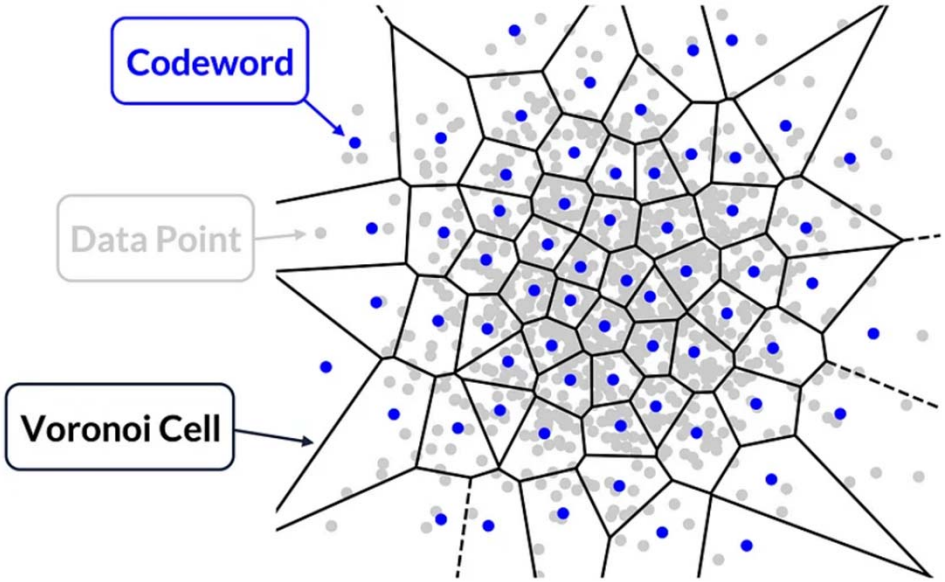
- Similarly, doing **vector quantization** at the output of an audio **encoder** (Dieleman et al., 2018)



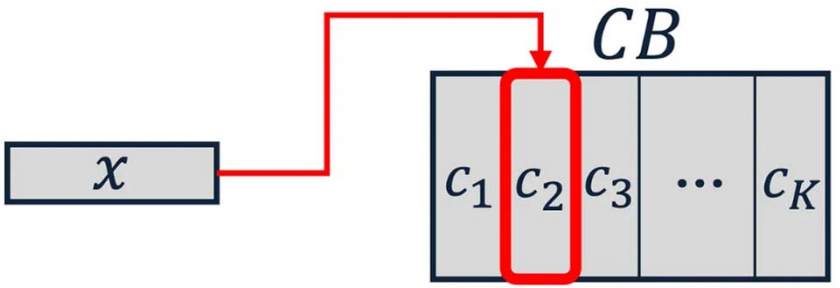
Ref: Dieleman et al, "The challenge of realistic music generation: modelling raw audio at scale," NeurIPS 2018

Vector Quantization

- Form a codebook by **clustering** in an embedding space from an encoder



Vector Quantization Operation (image by author)

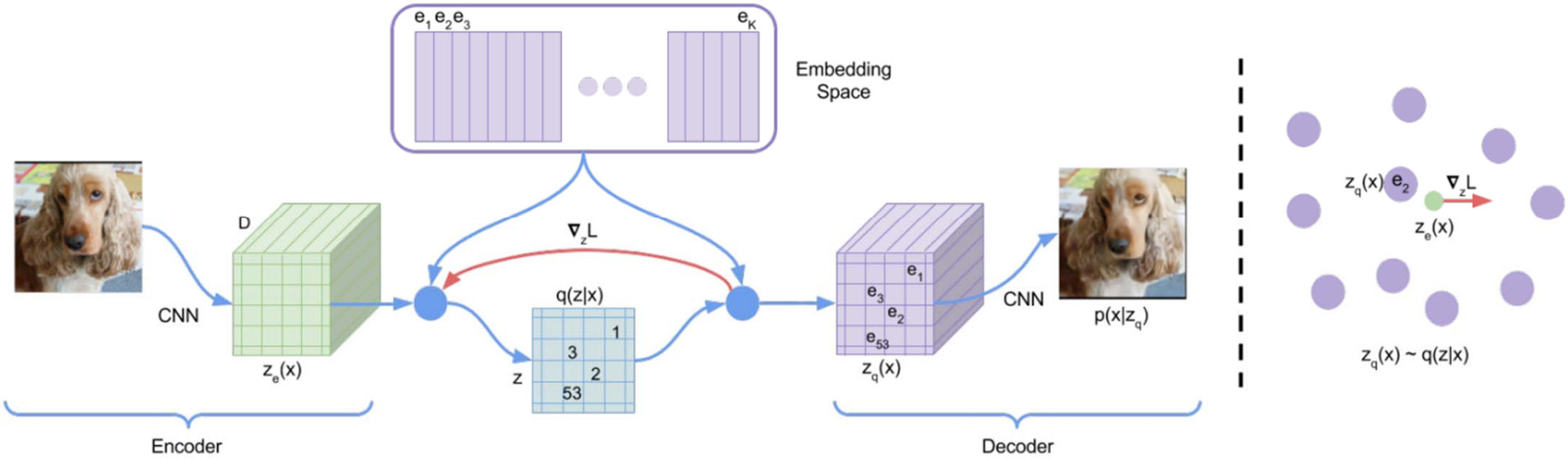


$$x_{quantized} = c_{i^*} \; ; \; i^* = \arg\min_i \|x - c_i\|^2 \; ; \; i \in \{1, ..., K\}$$

$$in\ this\ case : x_{quantized} = c_2$$

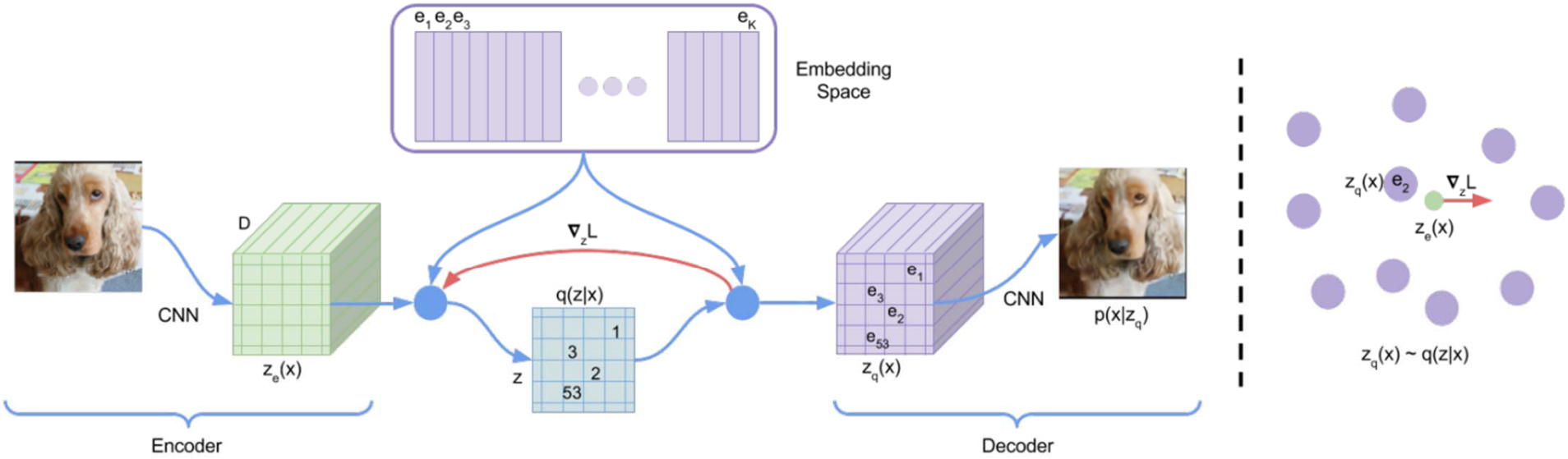
(Figure from: https://towardsdatascience.com/optimizing-vector-quantization-methods-by-machine-learning-algorithms-77c436d0749d)

VQVAE



- Encoder output: $z_e(\mathbf{x})$
- VQ: find the codeword $\mathbf{e}$ from a learned codebook $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_K\}$ that is closest to $z_e(\mathbf{x})$ to represent $z_e(\mathbf{x})$
- Decoder input: $z_q(\mathbf{x}) \triangleq \mathbf{e}$

VQVAE



$$L = \log p(x|z_q(x)) + \|\mathrm{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \mathrm{sg}[e]\|_2^2,$$

Reconstruction loss

Optimizes the encoder and decoder

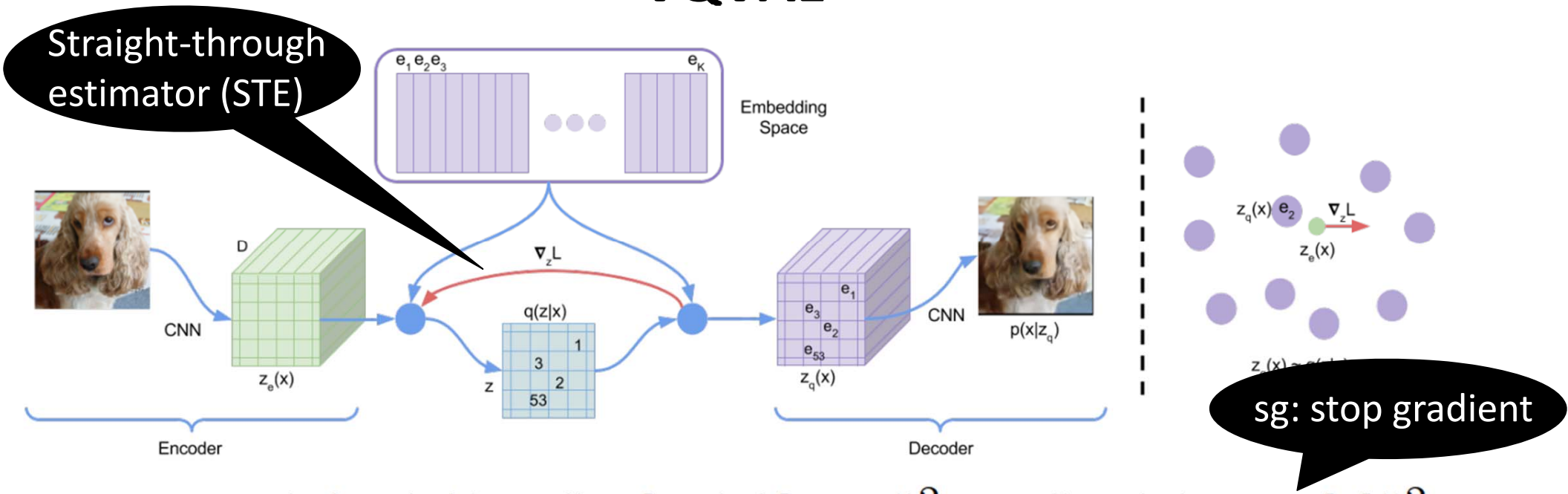
Codebook loss

Optimizes the codebook (move the codewords towards the encoder outputs)

commitment loss

Optimizes the encoder (move the encoder outputs towards the codewords)

VQVAE



$$L = \log p(x|z_q(x)) + \|\mathrm{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \mathrm{sg}[e]\|_2^2,$$

Reconstruction loss

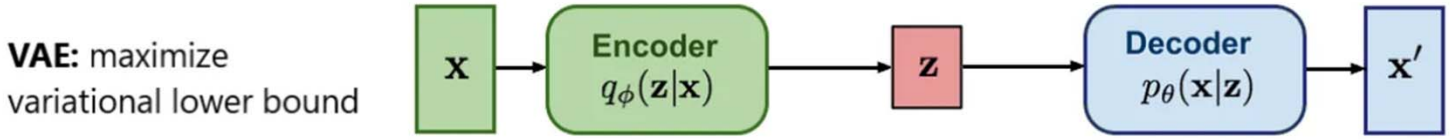
Codebook loss

commitment loss

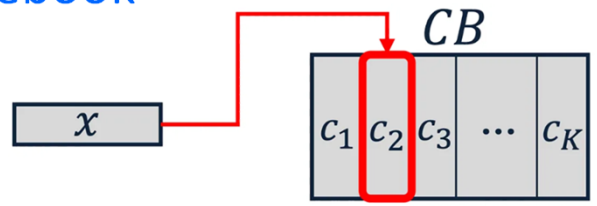
Decoder needs to recover the input from a quantized latent

Commitment loss helps improves convergence because the encoder output would “commit” to some codewords instead of changing the codewords too frequently

VQVAE vs VAE



- **VQVAE produces discrete codes** (indices for a fixed number of latents)
  - **Encoder output**: an arbitrary continuous latent $q_\phi(\mathbf{z}|\mathbf{x})$
  - **Decoder input**: a *continuous* latent *selected* from the codebook
  - **Codebook size** K → only *K* possible decoder input
  - **The latent** $\mathbf{z}$ → one-hot index vector of dimension K



- **VAE produces continuous latent**
  - **Encoder output**: mean and STD
  - **Decoder input**: sampled continuous latent $\mathbf{z}$ from the mean and STD

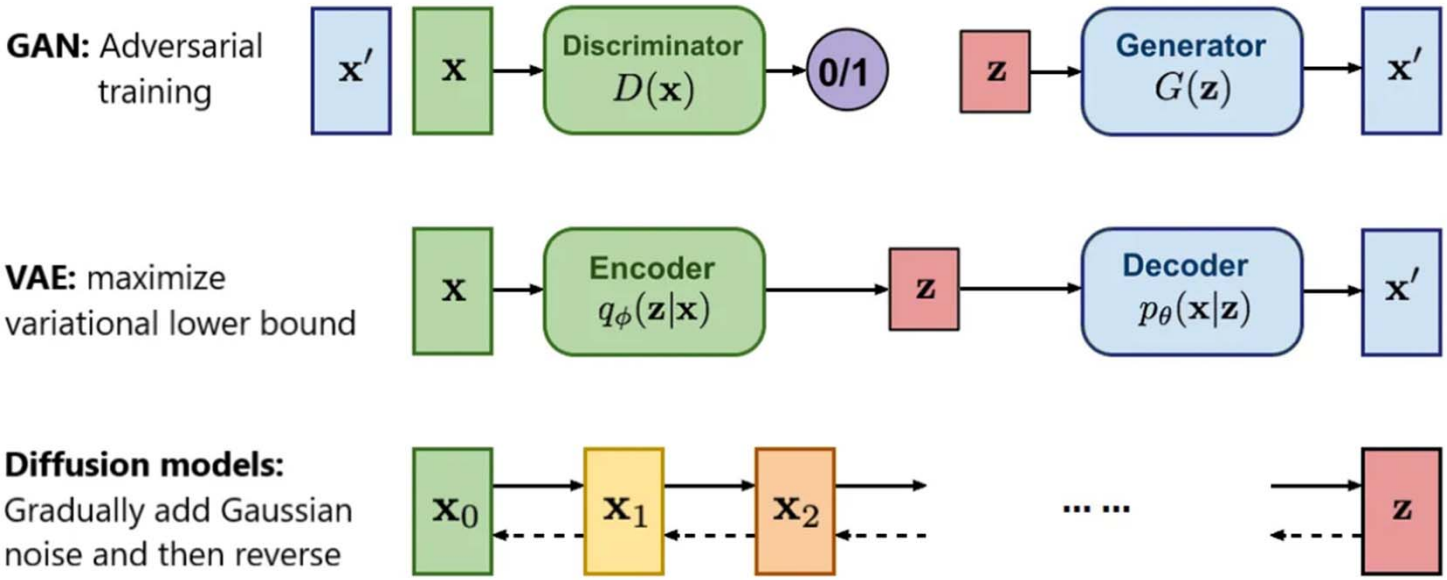
$$x_{quantized} = c_{i^*} \; ; \; i^* = \arg\min_i \|x - c_i\|^2 \; ; \; i \in \{1, ..., K\}$$

$$in\ this\ case : x_{quantized} = c_2$$

(Figure from: https://pub.towardsai.net/diffusion-models-vs-gans-vs-vaes-comparison-of-deep-generative-models-67ab93e0d9ae)

VQVAE vs VAE

- Both have an "**information bottleneck**"
- VQVAE produces **discrete codes** (indices for a fixed number of latents), while VAE produces **continuous latent**

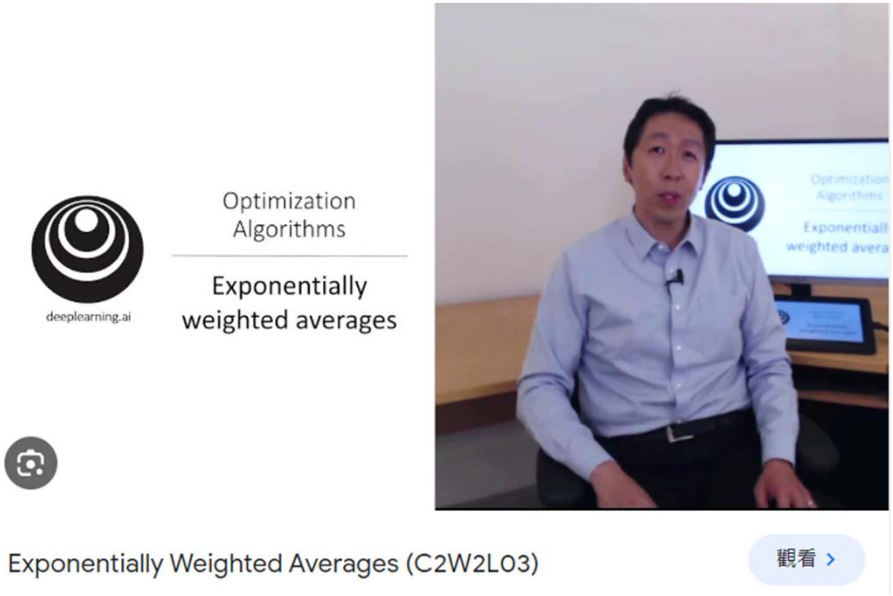


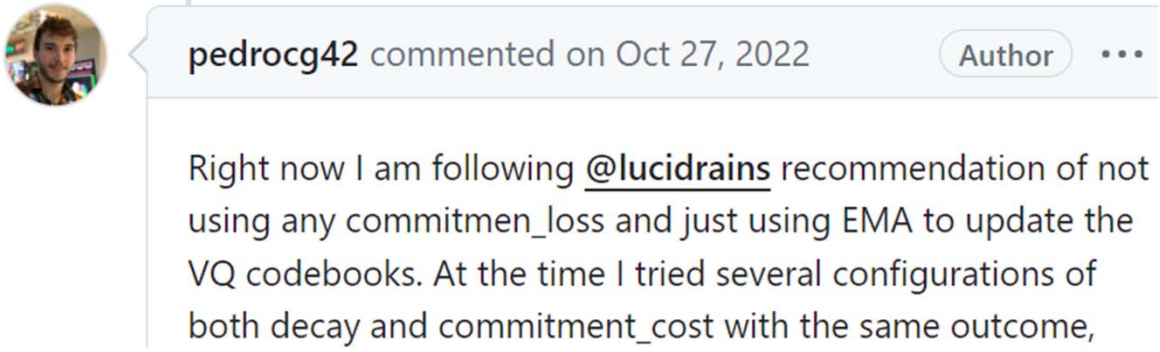
(Figure from: https://pub.towardsai.net/diffusion-models-vs-gans-vs-vaes-comparison-of-deep-generative-models-67ab93e0d9ae)

VQVAE Training Tips: Exponential Moving Average (EMA)

$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \text{sg}[e]\|_2^2,$$

Reconstruction loss | Codebook loss (EMA update) | commitment loss



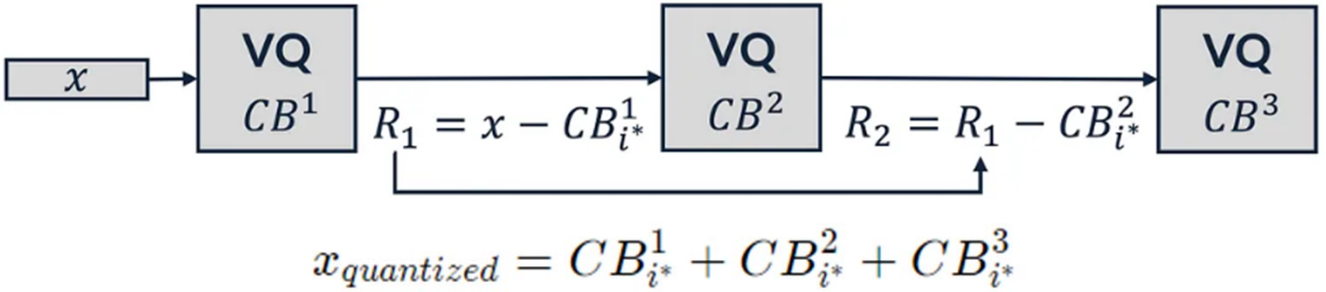


Some people said that we don't need the commitment loss if we use EMA
(https://github.com/lucidrains/vector-quantize-pytorch/issues/27)

Ref: Oord et al, "Neural discrete representation learning, NeurIPS 2017

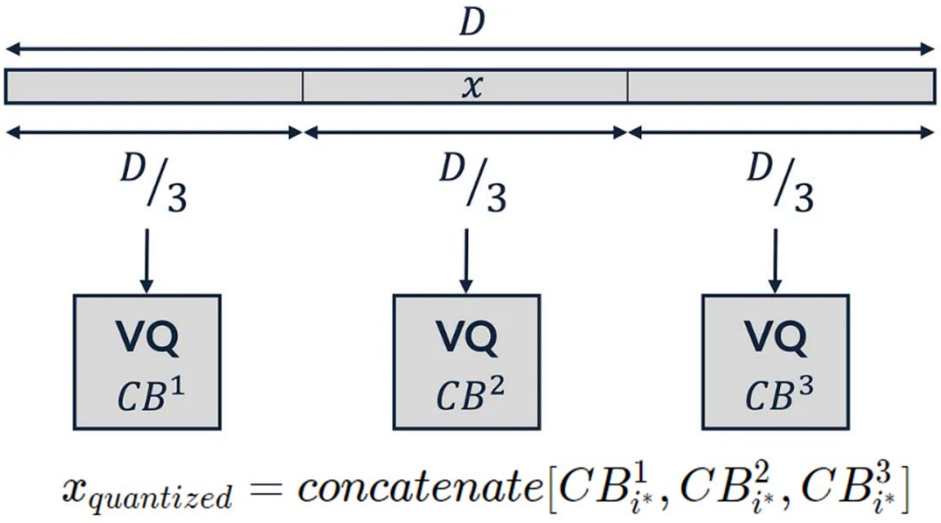
Variants of VQ

- Residual VQ (RVQ)



- Product VQ (PVQ)

  (a.k.a., decomposed VQ; DVQ)



- More… (e.g., Additive VQ)

(Figure from: https://towardsdatascience.com/optimizing-vector-quantization-methods-by-machine-learning-algorithms-77c436d0749d)

VQVQE Library

https://github.com/lucidrains/vector-quantize-pytorch

vector-quantize-pytorch

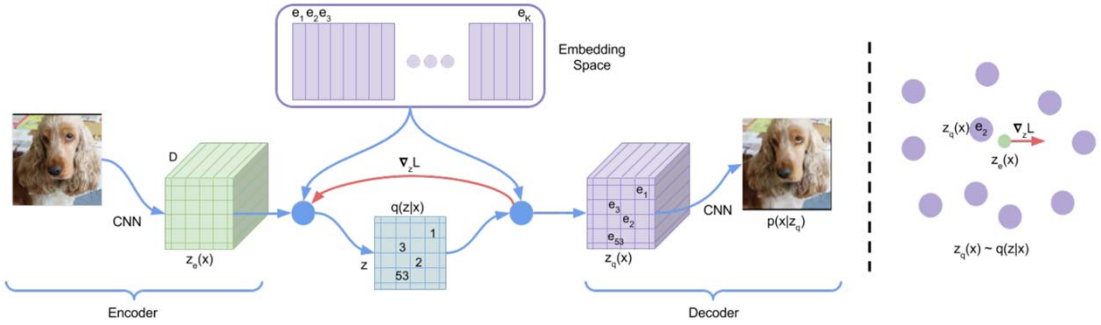
- Initialization
- **Lower codebook dimension**
- Cosine similarity
- **Expiring stale codes**
- Orthogonal regularization loss
- Multi-headed VQ
- Random Projection Quantizer
- Finite Scalar Quantization
- Lookup Free Quantization

VQVAE Training Tips: Increasing Codebook Usage

https://github.com/lucidrains/vector-quantize-pytorch

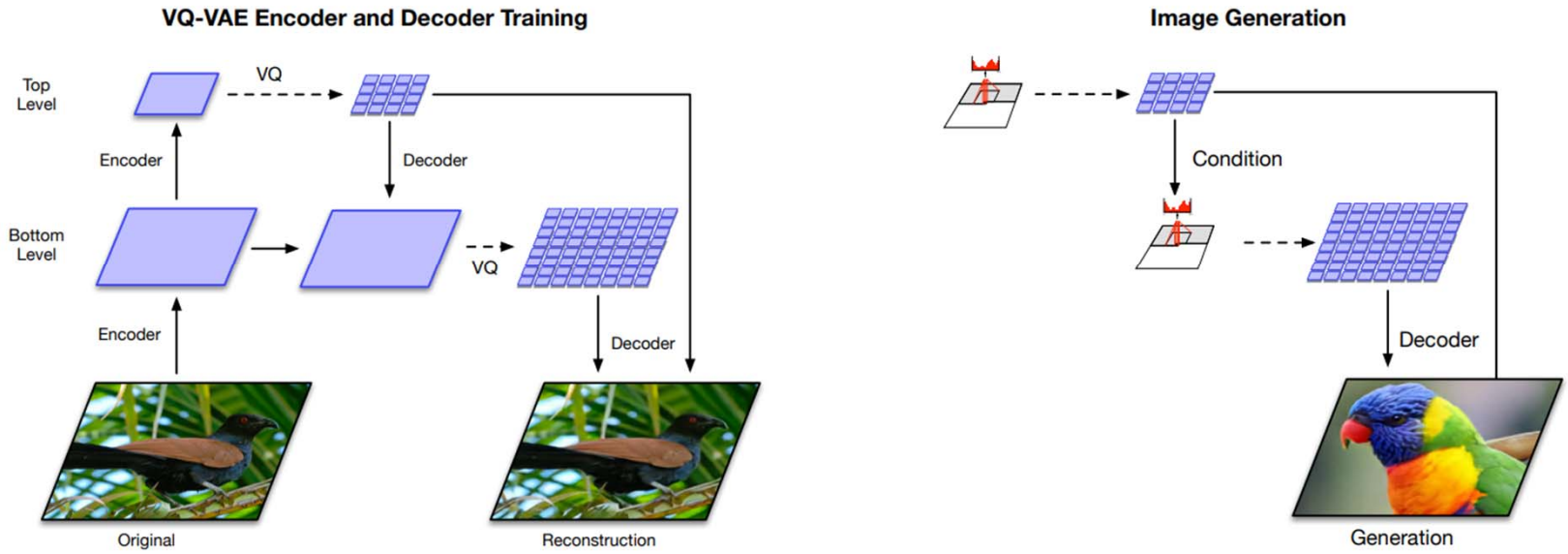
vector-quantize-pytorch

- Techniques to combat “dead” codebook entries (*index collapse*), which is a common problem when using vector quantizers
- **Lower codebook dimension**
  - Do VQ in a lower-dim space
  - The encoder values are projected down before being projected back to high dim after quantization



VQVAE 2: Hierarchical VQ

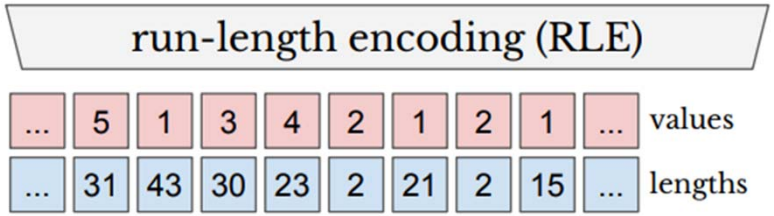
- The input to the model is a 256×256 image that is compressed to quantized latent maps of size 64×64 and 32×32 for the bottom and top levels, respectively. The decoder reconstructs the image from the two latent maps.



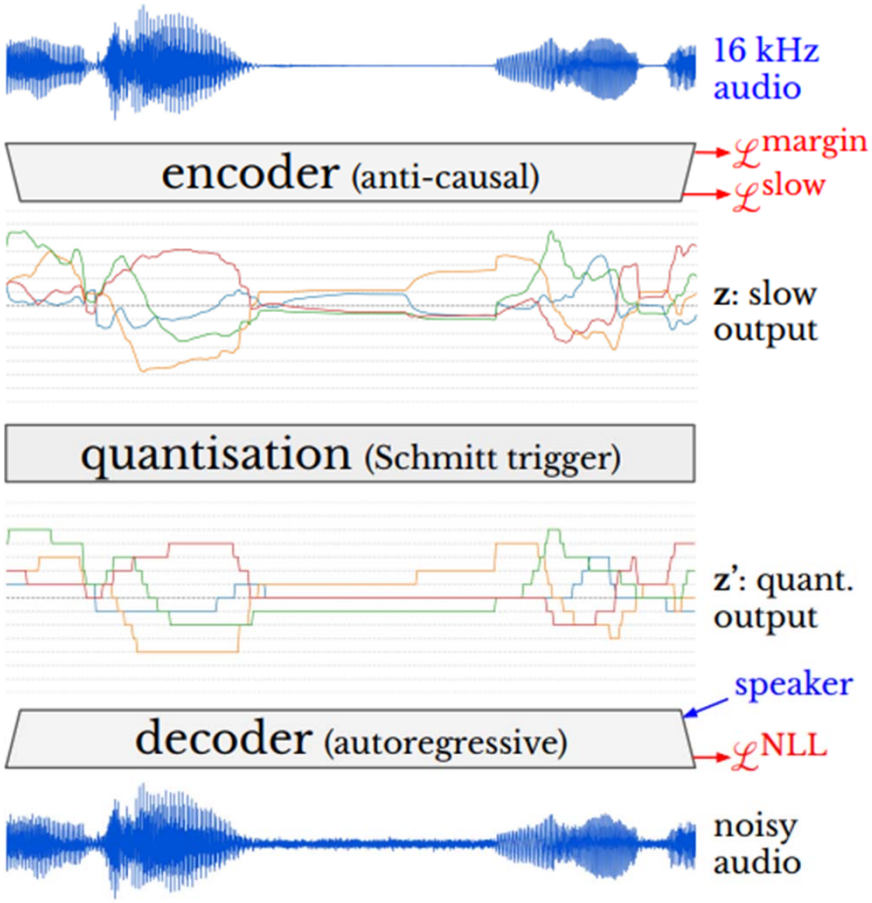
Ref: Razavi et al, "Generating diverse high-fidelity images with VQ-VAE-2," NeurIPS 2019

Variable-rate Discrete Representation Learning

- **Run length coding** via **s**low autoencoders (SlowAEs)
  - l repeated instances of the same value v will be represented as a (v, l)- tuple



- Given the RLE codes, train an RLE Transformer



Ref: Dieleman et al, "Variable-rate discrete representation learning," arXiv 2021

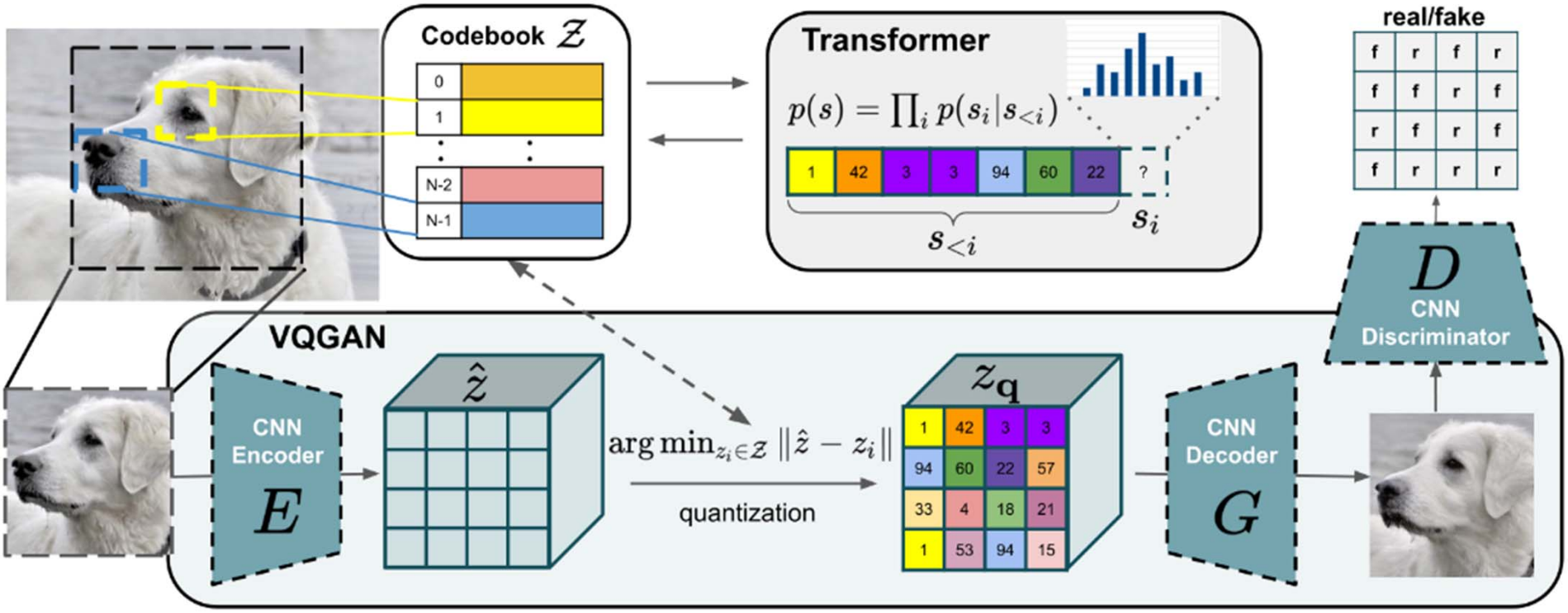
Summary

- We can use **VQVAE** to convert continuous data to discrete tokens
- What we get from VQVAE: **encoder**, **decoder**, and **codebook**

- We can then train a Transformer **LM** over the codeword sequences
  - Image generation: VQGAN
  - Music generation: Jukebox, KaraSinger, JukeDrummer, SingSong

- The VQVAE and the Transformer LM are trained separately

VQVAE-based Image Generation: VQGAN

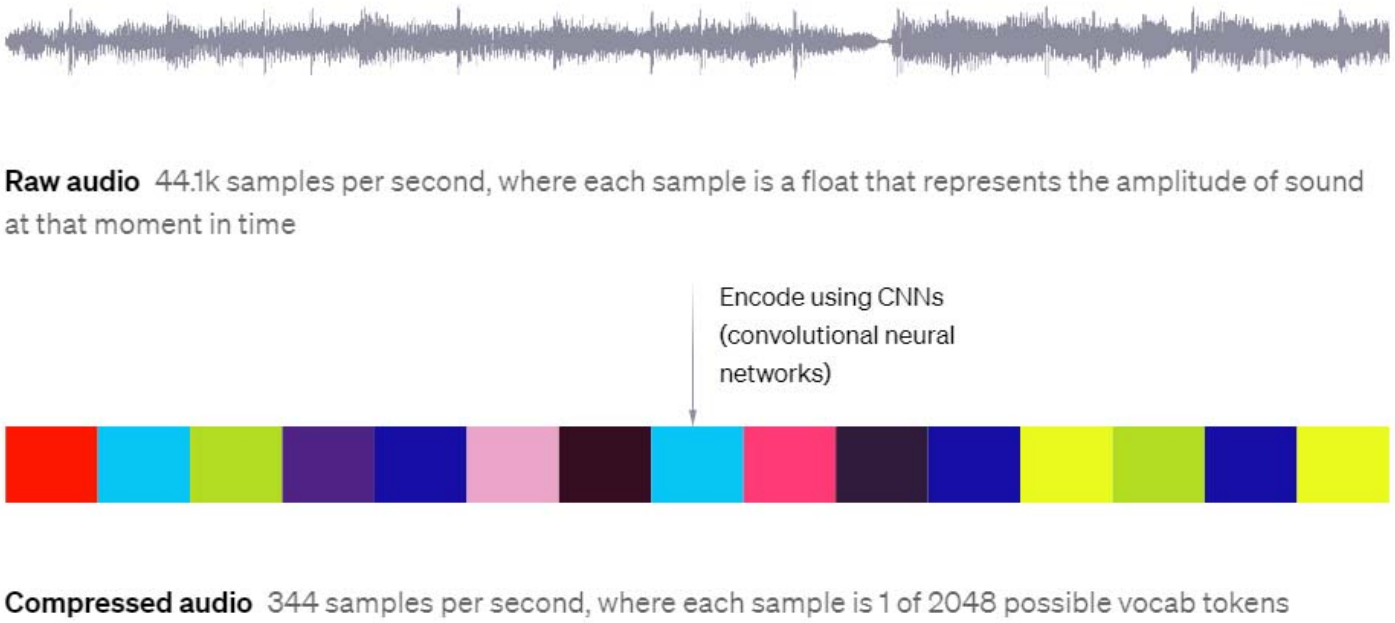


Ref: Esser et al, "Taming Transformers for high-resolution image synthesis," CVPR 2021
Ref: Yu et al, "Vector-quantized Image Modeling with Improved VQGAN," ICLR 2022

VQVAE-based Music Generation: JukeBox

https://openai.com/research/jukebox

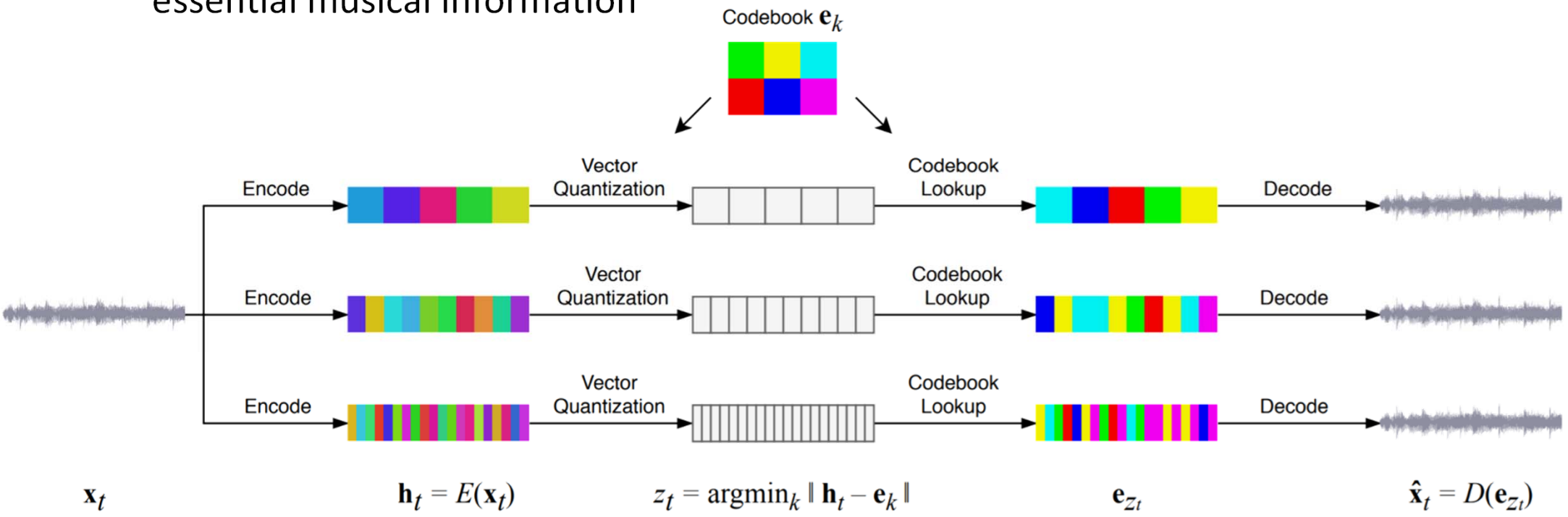
- Use **vector quantization** (VQ) to model waveforms as "**acoustic tokens**"



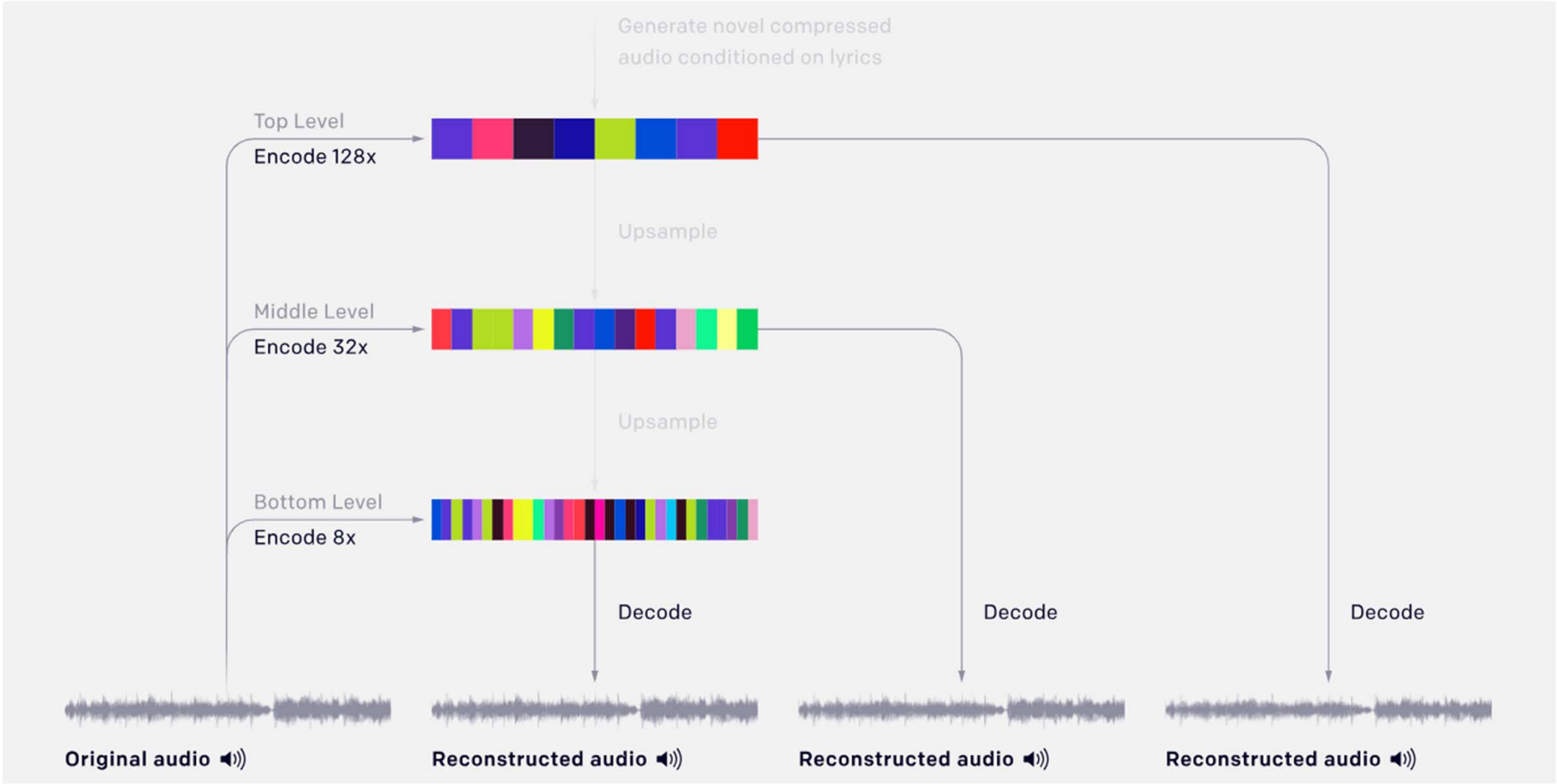
Ref: Dhariwal et al, "Jukebox: A generative model for music," arXiv 2020

VQVAE-based Music Generation: JukeBox

- Three layers of VQ for fidelity (like VQVAE 2): **top**, **middle**, **bottom**
  - Each VQ-VAE level independently encodes the input
  - **Bottom** produces the highest quality reconstruction, while **top** retains only the essential musical information

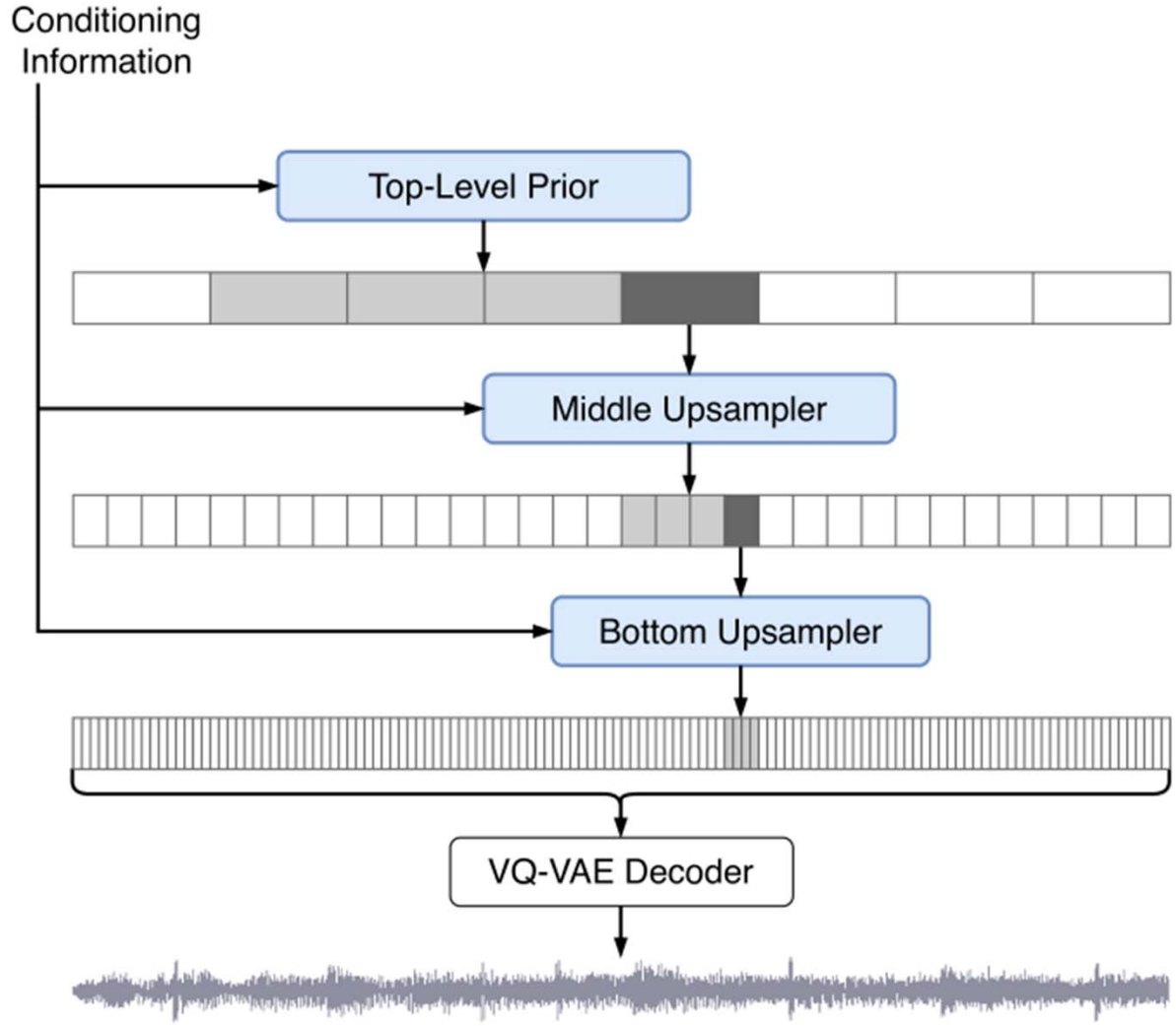


VQVAE-based Music Generation: JukeBox

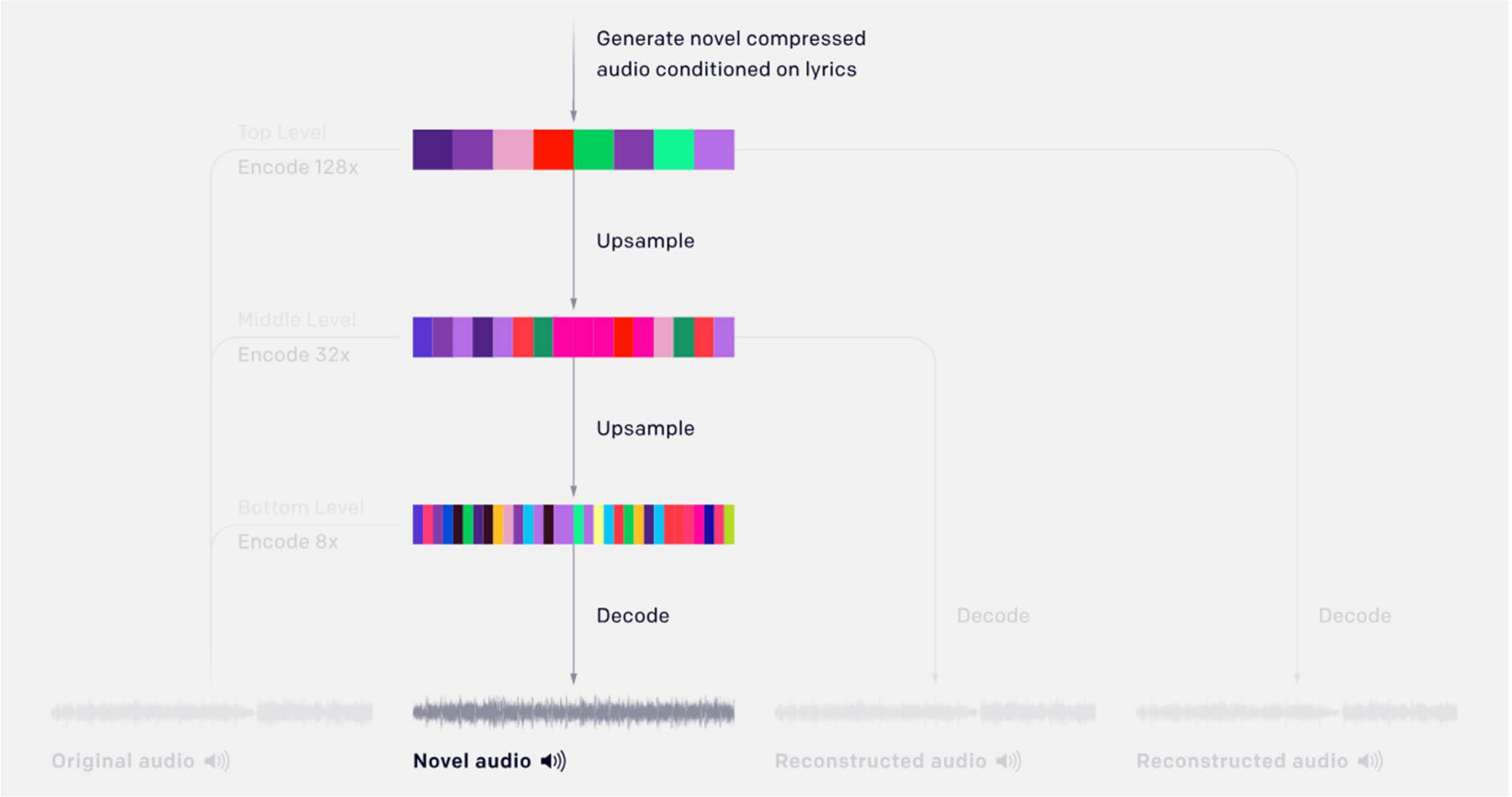


VQVAE-based Music Generation: JukeBox

- Take **lyrics** as condition
  - Given lyrics, generate **top-level** tokens
  - Given lyrics+top, generate **mid-level** tokens
  - Given lyrics+mid, generate **bottom-level** tokens
- Generate music waveforms that contain both **vocal** and **instrumental background** (mixed together)

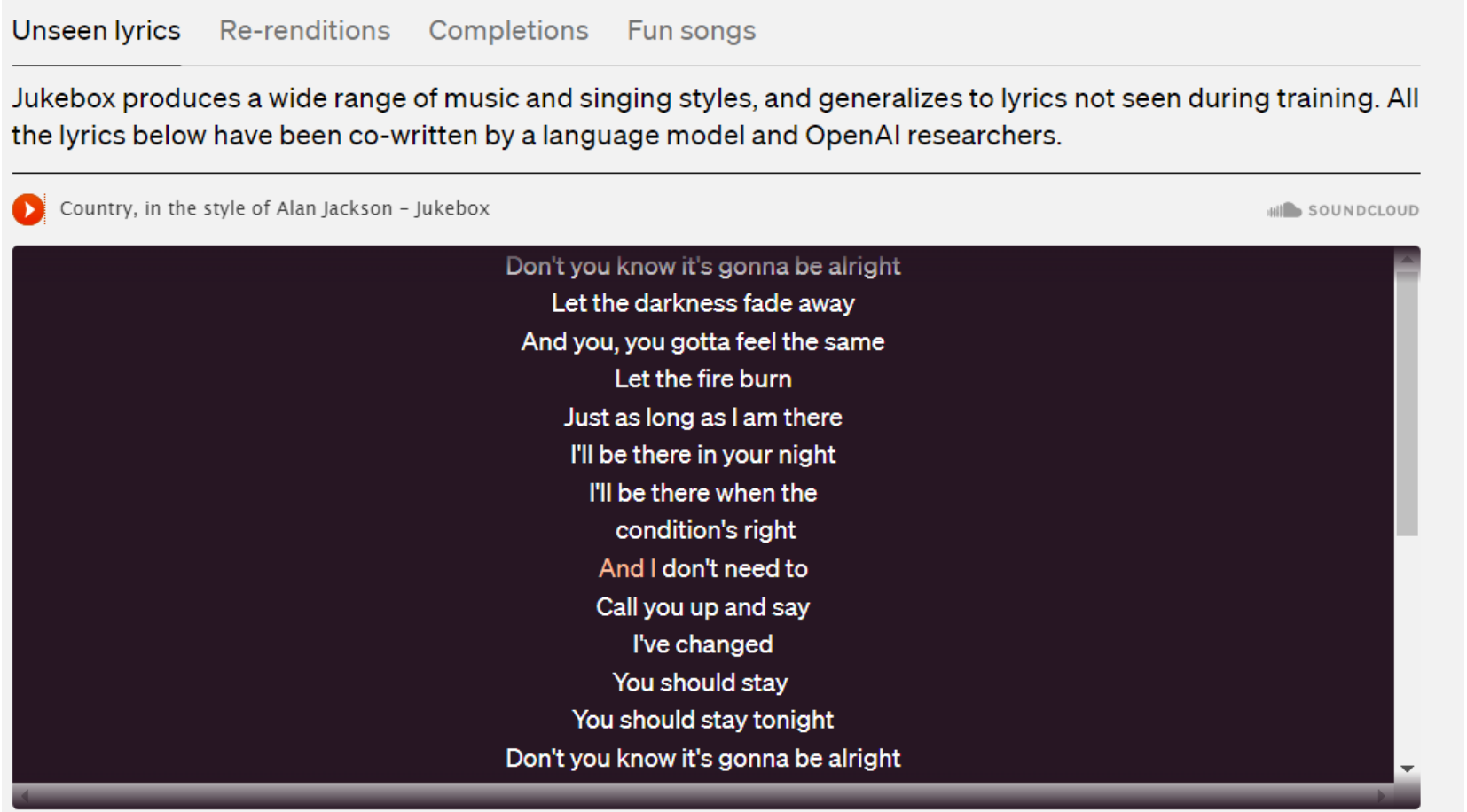


VQVAE-based Music Generation: JukeBox



VQVAE-based Music Generation: JukeBox

https://openai.com/research/jukebox



Issues of OpenAI's Jukebox

- **A game of computes**
  - 1M songs as training data & 512 V100s & total training time >6 weeks (i.e., around >60 years of single V100 time)

trained on 128 V100s for 2 weeks, and the top-level prior has 5 billion parameters and is trained on 512 V100s for 4 weeks. We use Adam with learning rate 0.00015 and weight decay of 0.002. For lyrics conditioning, we reuse the prior and add a small encoder, after which we train the model on 512 V100s for 2 weeks. The detailed hyperparameters for our models and training are provided in Appendix B.3.

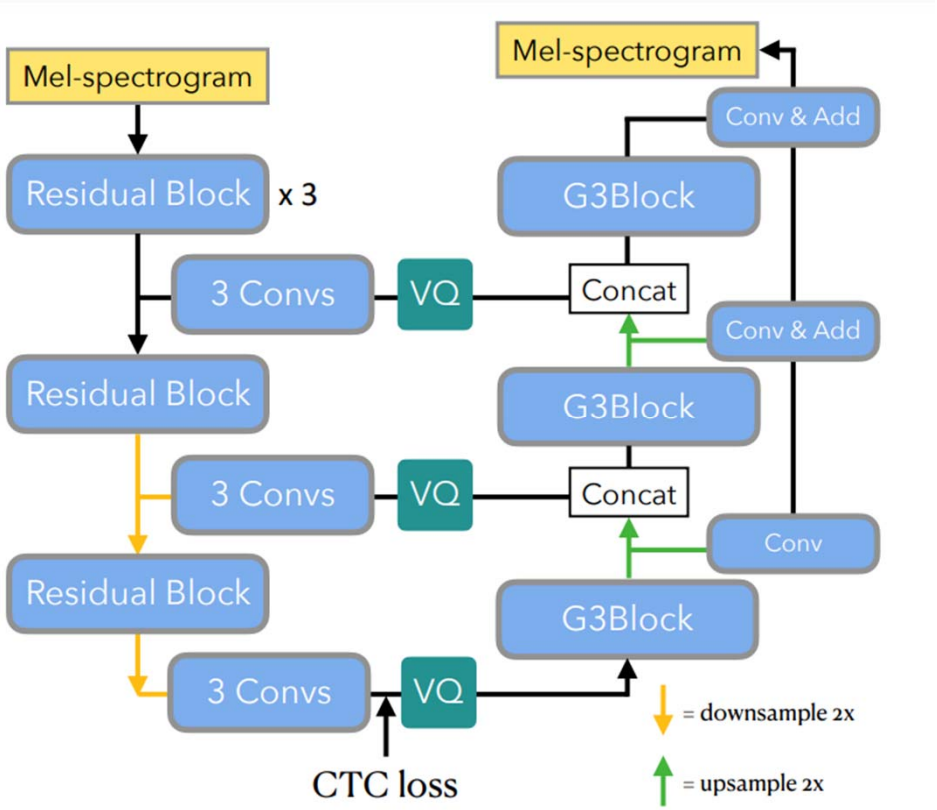
The current model takes around an hour to generate 1 minute of top level tokens. The upsampling process is very slow, as it proceeds sequentially through the sample. Currently it takes around 8 hours to upsample one minute of top level tokens. We can create a human-in-the-loop co-composition process at the top level only, using the VQ-VAE decoders

- **Low controllability**
  - The only control signal is the lyrics

KaraSinger: VQVAE on Mel-spectrograms

https://jerrygood0703.github.io/KaraSinger/

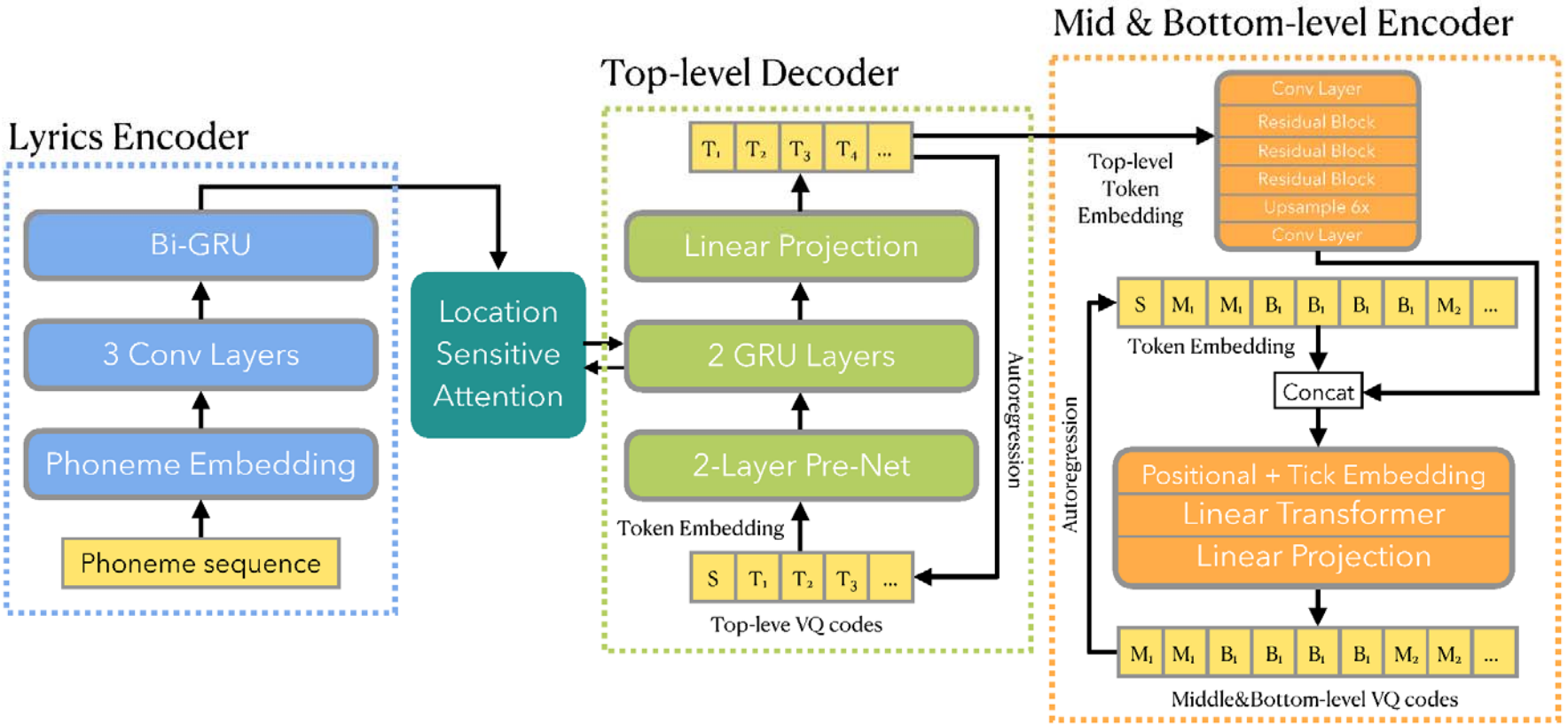
- Multi-scale hierarchical VQ-VAE with **top, mid, bottom** layers
- Encoder: five residual blocks
- Decoder: G3blocks (GRU + dilated convolution + group normalization)
- MelGAN vocoder
- 4 days training on a single NVIDIA RTX 2080Ti, 1:1 RTF for inference



Ref: Liao et al, "KaraSinger: Score-free singing voice synthesis with VQ-VAE using Mel-spectrograms," ICASSP 2022

KaraSinger: VQVAE on Mel-spectrograms

- GRU for top codes (T), Transformer for **joint prediction** of mid (M) & bottom (B)
- Token organization: T,M,M,B,B,B,B, T,M,M,B,B,B,B, …



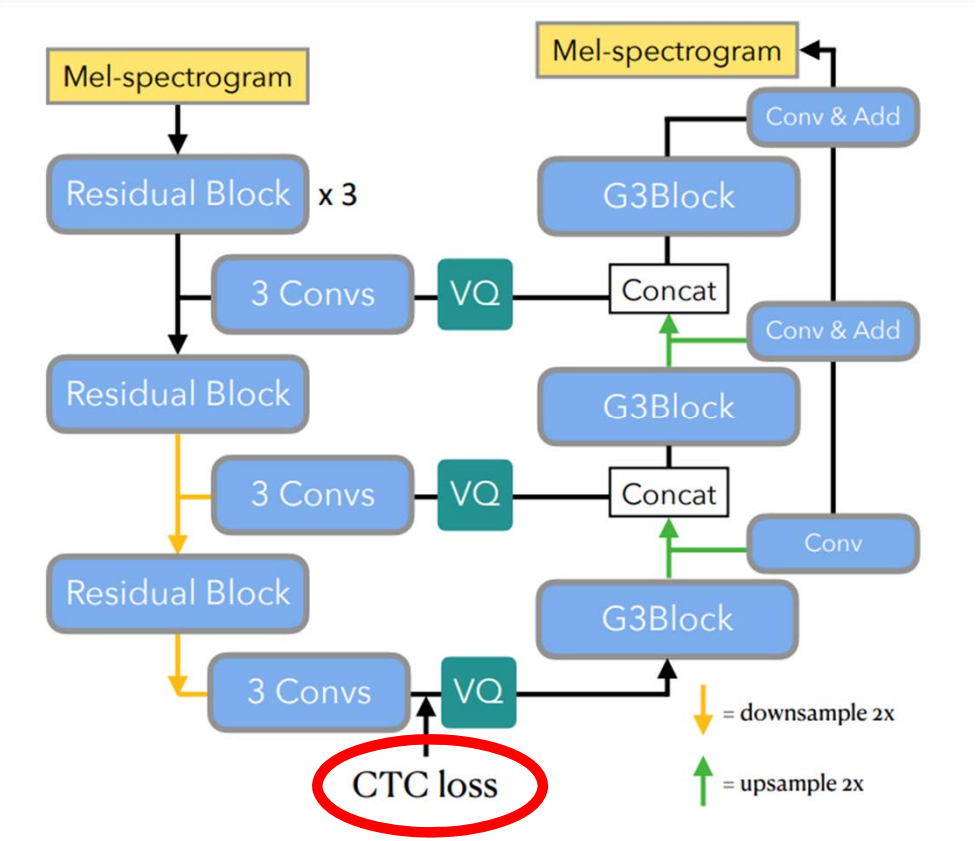
KaraSinger: VQVAE on Mel-spectrograms

https://jerrygood0703.github.io/KaraSinger/

- Connectionist temporal classification (**CTC**) loss on the top-level code to carry phoneme-related information
  - Without it the seq2seq model struggles to learn meaningful alignments in cross-attention to attend to the input lyrics

Model	Intelligibility	Musicality	Overall
noCTC	1.30 ± 0.14	2.00 ± 0.16	1.64 ± 0.14
3-level	3.30 ± 0.18	3.43 ± 0.16	3.19 ± 0.17
Proposed	4.23 ± 0.14	3.85 ± 0.14	3.67 ± 0.15

*3-level: no joint prediction of M and B

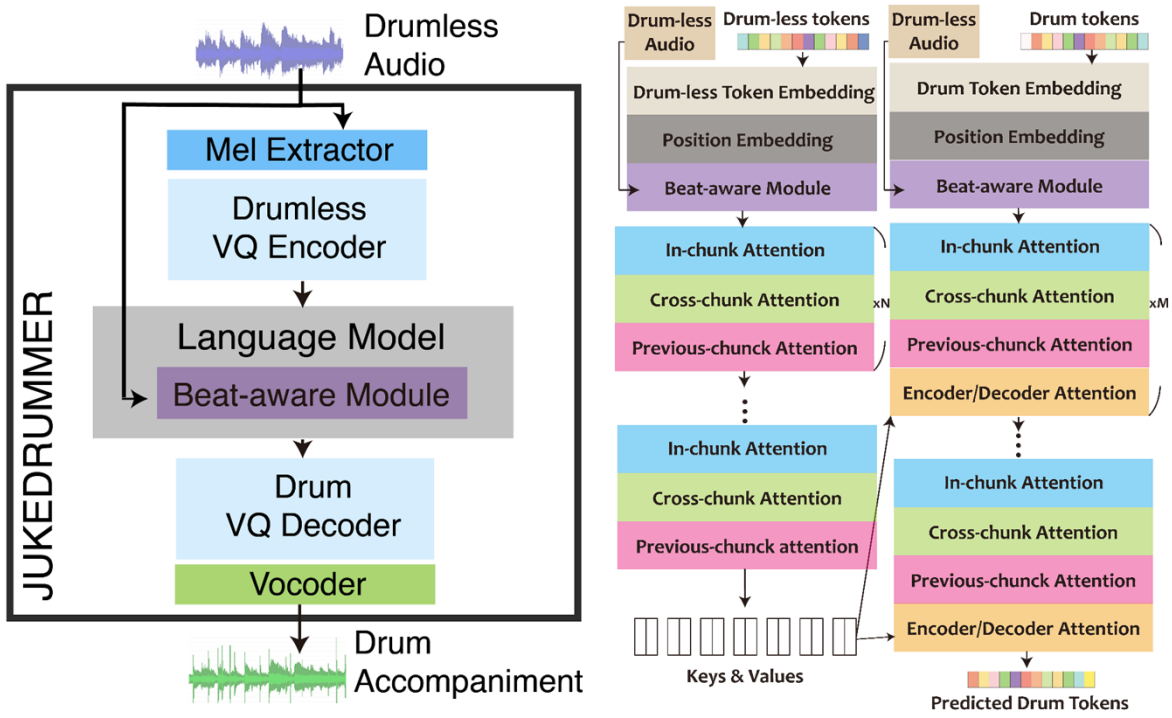


Ref: Liao et al, "KaraSinger: Score-free singing voice synthesis with VQ-VAE using Mel-spectrograms," ICASSP 2022

JukeDrummer: VQVAE on Mel-spectrograms

https://legoodmanner.github.io/jukedrummer-demo/

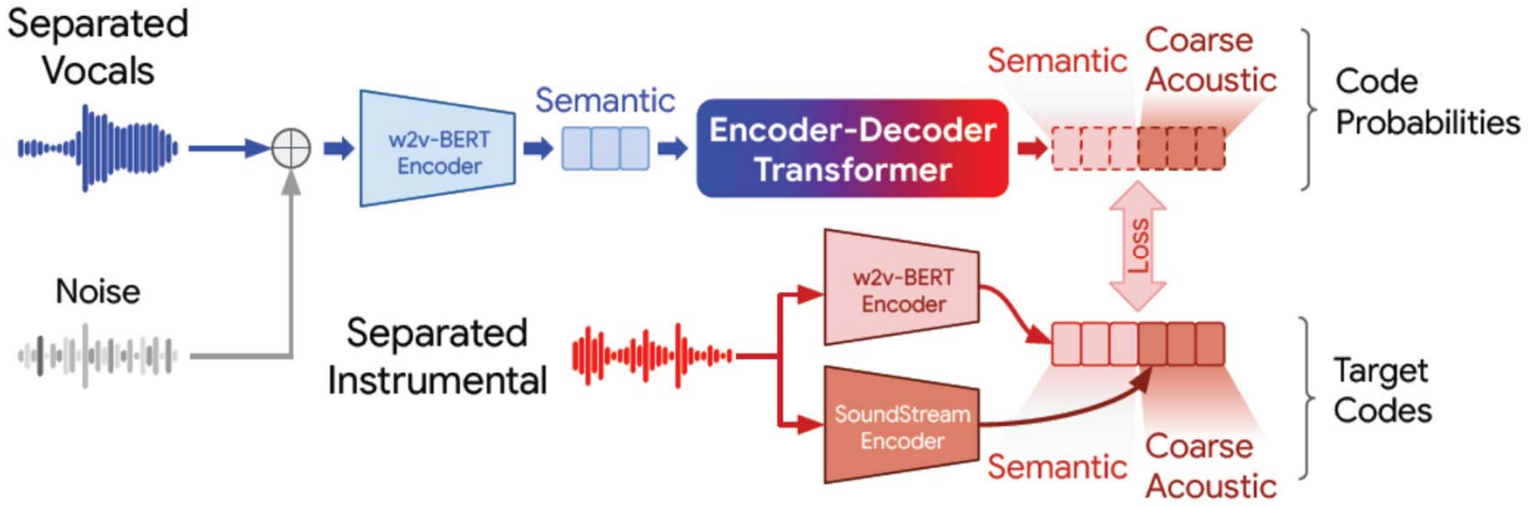
- **Given no-drum music, generate drum accompaniment**
  - Apply blind source separation to get pairs of drum/no-drum
  - Seq2seq: **no-drum tokens** as the source, **drum** tokens as the target to be generated
- Using Mel-spectrogram
  - Train on **1** GTX 1080-Ti for **2** days



Ref: Wu et al, "JukeDrummer: Conditional beat-aware drum accompaniment generation in the audio domain using Transformer VQ-VAE," ISMIR 2022

SingSong: VQVAE on Waveforms

https://storage.googleapis.com/sing-song/index.html



- **Given vocal, generate multi-instrument accompaniment**
  - Apply "blind source separation" to get pairs of vocal/backing tracks
  - Seq2seq: **vocal tokens** as the source, generate the backing tokens
- Use **raw waveforms** for VQ-VAE; bypass Mel-vocoders

Ref: Donahue et al, "SingSong: Generating musical accompaniments from singing," arXiv 2023

BTW, the Pursuit of "Audio Words" is Actually Not New…

Scale	Formulation	Notation	Dimension
Frame level (alphabet; e.g., 46 ms) (elementary unit)	$\hat{\alpha}_{\mathbf{s,t}} = \arg\min_{\alpha_{s,t}} \|\mathbf{x}_{s,t} - \mathbf{D}_1\alpha_{s,t}\|_2^2 + \lambda f(\alpha_{s,t})$	$\mathbf{D}_1$: first-layer codebook of audio-alphabets $\mathbf{x}_{s,t}$: frame-level acoustic feature $\alpha_{s,t}$: frame-level encoding over audio-alphabets	$\mathbf{D}_1 \in \mathbb{R}^{m\times k_1}$ $\mathbf{x}_{s,t} \in \mathbb{R}^m$ $\alpha_{s,t} \in \mathbb{R}^{k_1}$
Segment level (word; e.g., 2.5 sec) (combination of alphabets)	$\hat{\beta}_{\mathbf{s}} = \arg\min_{\beta_s} \|\sum_t \alpha_{s,t} - \mathbf{D}_2\beta_s\|_2^2 + \lambda f(\beta_s)$	$\mathbf{D}_2$: second-layer codebook of audio-words level $\overline{\alpha}_s = \sum_t \alpha_{s,t}$: pooled frame-level encoding β_s: segment-level encoding over audio-words	$\mathbf{D}_2 \in \mathbb{R}^{k_1\times k_2}$ $\overline{\alpha}_s \in \mathbb{R}^{k_1}$ $\beta_s \in \mathbb{R}^{k_2}$
Clip level (document; e.g., 30 sec)	$\gamma_1 = \sum_s \overline{\alpha}_s = \sum_{s,t} \alpha_{s,t}$ $\gamma_2 = \sum_s \beta_s$	γ_1: bag-of-audio alphabet (BoAA) γ_2: bag-of-audio word (BoAW)	$\gamma_1 \in \mathbb{R}^{k_1}$ $\gamma_2 \in \mathbb{R}^{k_2}$

- Codebook construction methods
  - Exemplar-based method
  - k-means
  - Online dictionary learning
- Encoding methods
  - l_1-regularized sparse coding
  - Top- Vector Quantization
  - Triangle k-means

TABLE III
VARIANTS OF TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY

Abbr.	Formulation	Abbr.	Formulation
tf_b	$f_{d,t}$	idf_b	$\ln \|\mathcal{D}\|/F_t$
tf_{l1}	$\ln(1+f_{d,t})$	idf_p	$\ln (\|\mathcal{D}\| - F_t)/F_t$
tf_{l2}	$1+\log_2 f_{d,t}$	idf_{e1}	$(\max_{t'\in\mathcal{T}} n_{t'}) - n_t$
tf_{o1}	$\frac{f_{d,t}}{f_{d,t}+\kappa\|d\|/\|\Delta d\|}$	idf_{e2}	$1 - \frac{n_t}{\ln\|\mathcal{D}\|}$
tf_{o2}	$\frac{(\kappa+1)f_{d,t}}{f_{d,t}+\kappa\left((1-b)+b\cdot\frac{\|d\|}{\|\Delta d\|}\right)}$	idf_o	$\ln\left(\frac{\|\mathcal{D}\|-F_t+0.5}{F_t+0.5}\right)$

$f_{d,t}$ — the number of occurrences of term t in document d
F_t — the number of documents in $\mathcal{D}$ that contain t
$|d|$ — cardinality (length) of d; $|\Delta d|$ — average document length in $\mathcal{D}$
n_t — entropy of term t in $\mathcal{D}$; $n_t = -\sum_d (f_{d,t}/F_t)\ln(f_{d,t}/F_t)$
$\mathcal{T}$ — the universe of terms; $\mathcal{D}$ — the universe of documents

Ref 1: Yeh et al, "AWtoolbox: Characterizing audio information using audio words," MM 2014

Ref 2: Su et al, "A systematic evaluation of the bag-of-frames representation for music information retrieval," TMM 2014

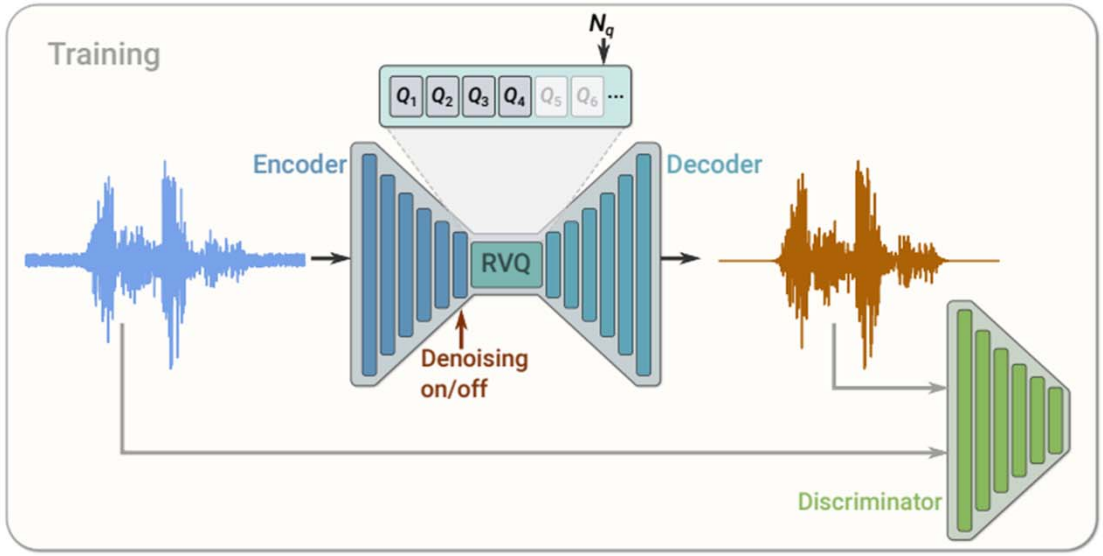
Outline

- Image/audio tokenization and generation via VQVAE
- **Audio codec models**
- Text-to-music: linking text and musical audio
  - Transformer-based approach
  - Diffusion-based approach

Audio Codec

- **Use raw waveforms for VQ-VAE**

- Initially developed for audio compression
  - Bitrate vs fidelity
  - Computational complexity vs coding efficiency



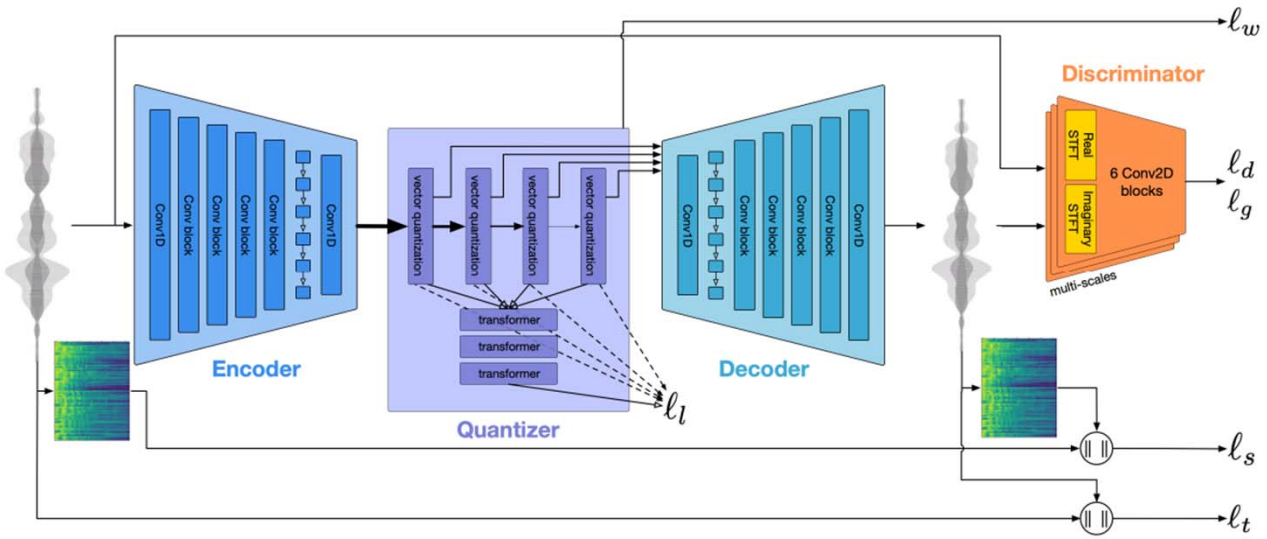
- Use vector quantization (VQ) for quantization
  - Produce **discrete tokens** that can be used for building a Transformer LM
  - **Not the focus of these papers**

Ref: Zeghidour et al, "SoundStream: An end-to-end neural audio codec," IEEE TASLP 2021

Audio Codec

- Many new models
  - **SoundStream**: An end-to-end neural audio codec (arXiv:2107.03312)
  - **EnCodec**: High fidelity neural audio compression (arXiv:2210.13438)
  - **HiFi-Codec**: Group-residual vector quantization for high fidelity audio codec (arXiv:2305.02765)
  - High-fidelity audio compression with **improved RVQGAN** (arXiv:2306.06546)

System diagram of FB's Encodec model (proposed by the Demucs team)



SoundStream

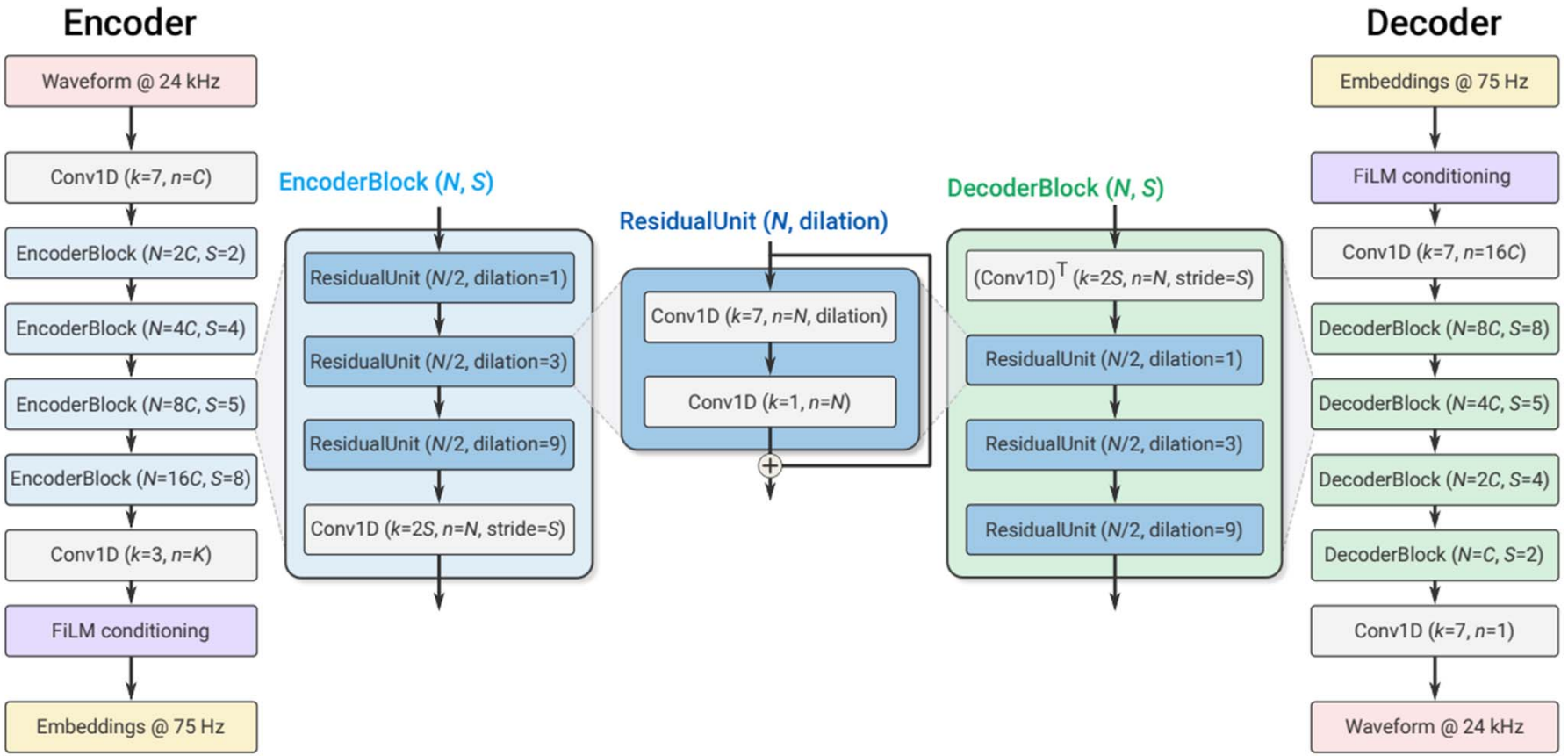


Fig. 3: Encoder and decoder model architecture.

Ref: Zeghidour et al, "SoundStream: An end-to-end neural audio codec," IEEE TASLP 2021

SoundStream

- Use both a waveform-domain discriminator
- And an STFT-based discriminator, which receives as input the complex-valued STFT of the input waveform, expressed in terms of real and imaginary parts

- Both discriminators are fully convolutional

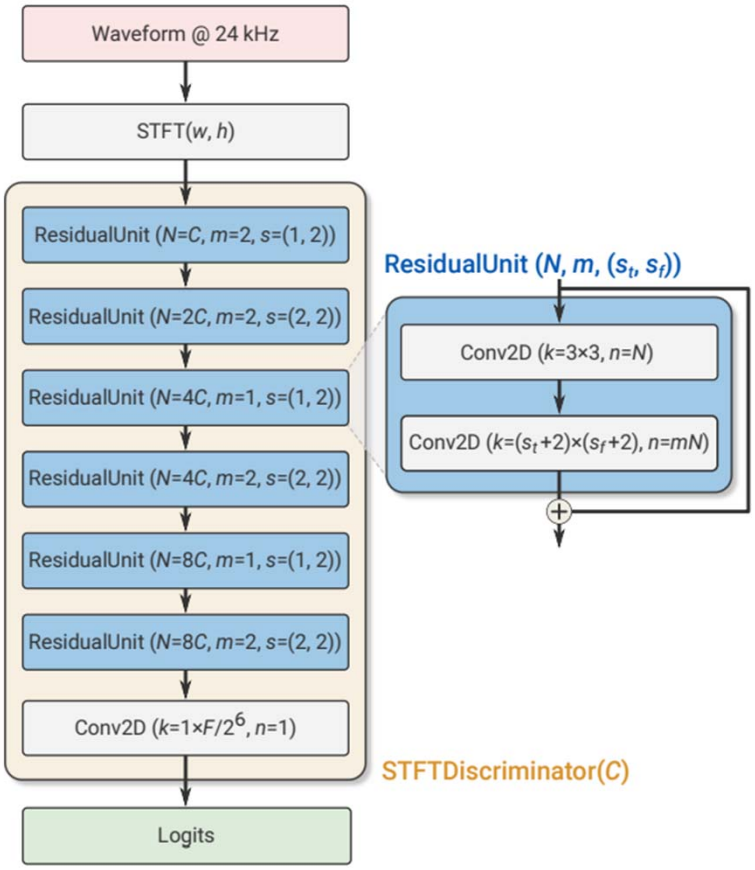


Fig. 4: STFT-based discriminator architecture.

Ref: Zeghidour et al, "SoundStream: An end-to-end neural audio codec," IEEE TASLP 2021

SoundStream

- Use 8 layers of RVQ
- Codebook size set to 1024 for each quantizer

TABLE II: Trade-off between residual vector quantizer depth and codebook size at 6 kbps.

Number of quantizers N_q	8	16	80
Codebook size N	1024	32	2
ViSQOL	4.01 $\pm$ 0.03	3.98 $\pm$ 0.03	3.92 $\pm$ 0.03

Ref: Zeghidour et al, “SoundStream: An end-to-end neural audio codec,” IEEE TASLP 2021

EnCodec

https://github.com/facebookresearch/encodec

- A causal model operating at 24 kHz on monophonic audio trained on a variety of audio data.
- A non-causal model operating at 48 kHz on stereophonic audio trained on music-only data.

The 24 kHz model can compress to 1.5, 3, 6, 12 or 24 kbps, while the 48 kHz model support 3, 6, 12 and 24 kbps. We also provide a pre-trained language model for each of the models, that can further compress the representation by up to 40% without any further loss of quality.

Compression

```
encodec [-b TARGET_BANDWIDTH] [-f] [--hq] [--lm] INPUT_FILE [OUTPUT_FILE]
```

Decompression

```
encodec [-f] [-r] ENCODEC_FILE [OUTPUT_WAV_FILE]
```

Ref: Défossez et al, “High fidelity neural audio compression,” TMLR 2023

EnCodec

Extracting discrete representations

```
from encodec import EncodecModel
from encodec.utils import convert_audio
```

```
# Instantiate a pretrained EnCodec model
model = EncodecModel.encodec_model_24khz()
# The number of codebooks used will be determined bythe bandwidth selected.
# E.g. for a bandwidth of 6kbps, `n_q = 8` codebooks are used.
# Supported bandwidths are 1.5kbps (n_q = 2), 3 kbps (n_q = 4), 6 kbps (n_q = 8) and 12 kbps (n_q =16) and 24
# For the 48 kHz model, only 3, 6, 12, and 24 kbps are supported. The number
# of codebooks for each is half that of the 24 kHz model as the frame rate is twice as much.
model.set_target_bandwidth(6.0)

# Load and pre-process the audio waveform
wav, sr = torchaudio.load("<PATH_TO_AUDIO_FILE>")
wav = convert_audio(wav, sr, model.sample_rate, model.channels)
wav = wav.unsqueeze(0)

# Extract discrete codes from EnCodec
with torch.no_grad():
    encoded_frames = model.encode(wav)
codes = torch.cat([encoded[0] for encoded in encoded_frames], dim=-1)  # [B, n_q, T]
```

HiFi-Codec

- Shows that using 4 codebooks is enough

Method	Sample rate (K hz)	Down-sample times	Number of codebooks	PESQ ↑	STOI ↑
Encodec (Facebook)	24	320	8	3.01	0.94
Encodec (Facebook)	24	320	12	3.21	0.95
Encodec (ours)	24	240	8	3.62	0.94
Encodec (ours)	24	32	2	3.08	0.91
Encodec (ours)	16	320	8	3.04	0.93
SoundStream (ours)	16	320	12	3.26	0.95
HiFi-Codec	24	240	4	3.63	0.95
HiFi-Codec	24	240	8	**3.92**	**0.95**
HiFi-Codec	24	320	4	3.64	0.95
HiFi-Codec	16	320	4	3.22	0.94

Ref: Yang et al, “HiFi-Codec: Group-residual vector quantization for high fidelity audio codec,” arXiv 2023

Improved RVQGAN

Outperform EnCodec in both objective and subjective studies

Ablation on	Decoder dim.	Activation	Multi-period	Single-scale	# of STFT bands	Multi-scale mel.	Latent dim	Quant. method	Quant. dropout	Bitrate (kbps)	Balanced samp.	Mel distance ↓	STFT distance ↓	ViSQOL ↑	SI-SDR ↑	Bitrate efficiency ↑
	1536	snake	✓	✗	5	✓	8	Proj.	1.0	8	✓	1.09	1.82	3.96	9.12	99%
Architecture	512	snake	✓	✗	5	✓	8	Proj.	1.0	8	✓	1.11	1.83	3.91	8.72	99%
	1024	snake	✓	✗	5	✓	8	Proj.	1.0	8	✓	1.07	1.82	3.96	9.07	99%
	1536	relu	✓	✗	5	✓	8	Proj.	1.0	8	✓	1.17	1.81	3.83	6.92	99%
Discriminator	1536	snake	✗	✗	✗	✓	8	Proj.	1.0	8	✓	1.13	1.92	4.12	1.07	62%
	1536	snake	✓	✗	1	✓	8	Proj.	1.0	8	✓	1.07	1.80	3.98	9.07	99%
	1536	snake	✗	✗	5	✓	8	Proj.	1.0	8	✓	1.07	1.81	3.97	9.04	99%
	1536	snake	✗	✓	5	✓	8	Proj.	1.0	8	✓	1.08	1.82	3.95	8.51	99%
Reconstruction loss	1536	snake	✓	✗	5	✗	8	Proj.	1.0	8	✓	1.10	1.87	4.01	7.68	99%
Latent dim	1536	snake	✓	✗	5	✓	2	Proj.	1.0	8	✓	1.44	2.08	3.65	2.22	84%
	1536	snake	✓	✗	5	✓	4	Proj.	1.0	8	✓	1.20	1.89	3.86	7.15	97%
	1536	snake	✓	✗	5	✓	32	Proj.	1.0	8	✓	1.10	1.84	3.95	9.05	98%
	1536	snake	✓	✗	5	✓	256	Proj.	1.0	8	✓	1.31	1.97	3.79	5.09	59%
Quantization setup	1536	snake	✓	✗	5	✓	8	EMA	1.0	8	✓	1.11	1.84	3.94	8.33	97%
	1536	snake	✓	✗	5	✓	8	Proj.	0.0	8	✓	0.98	1.70	4.09	10.14	99%
	1536	snake	✓	✗	5	✓	8	Proj.	0.25	8	✓	0.99	1.69	4.04	10.00	99%
	1536	snake	✓	✗	5	✓	8	Proj.	0.5	8	✓	1.01	1.75	4.03	9.74	99%
	1536	snake	✓	✗	5	✓	8	Proj.	1.0	24	✓	0.73	1.62	4.16	13.83	99%
Data	1536	snake	✓	✗	5	✓	8	Proj.	1.0	8	✗	1.09	1.94	3.89	8.89	99%

Ref: Kumar et al, "High-fidelity audio compression with improved RVQGAN," NeurIPS 2023

Improved RVQGAN

https://github.com/descriptinc/descript-audio-codec

```
python3 -m dac download --model_type 44khz # downloads the 44kHz variant
python3 -m dac download --model_type 24khz # downloads the 24kHz variant
python3 -m dac download --model_type 16khz # downloads the 16kHz variant
```

```
# Load audio signal file
signal = AudioSignal('input.wav')

# Encode audio signal as one long file
# (may run out of GPU memory on long files)
signal.to(model.device)

x = model.preprocess(signal.audio_data, signal.sample_rate)
z, codes, latents, _, _ = model.encode(x)
```

Ref: Défossez et al, “High fidelity neural audio compression,” TMLR 2023

Outline

- Image/audio tokenization and generation via VQVAE
- Audio codec models
- **Text-to-music: linking text and musical audio**
  - Transformer-based approach
  - Diffusion-based approach

Rapid Progress in “Text-to-Music”

- Given arbitrary text input, generate background music (BGM)

- **Google**’s MusicLM model (2023.01): https://google-research.github.io/seanet/musiclm/examples/
- **ByteDance**’s MeLoDy model (2023.05): https://Efficient-MeLoDy.github.io/
- **Facebook**’s MusicGen model (2023.06): https://github.com/facebookresearch/audiocraft

- Also great progress in “**text-to-sound**” (mostly using CLAP) but omitted here

TABLE 8: Large Music Models, TTM: Text-to-Music, VIM: Vocals to instrumental music, MTL: Melody to Lyric, MTM: Music to music, TSM: Text-to-symbolic music, PTM: Programming to Music

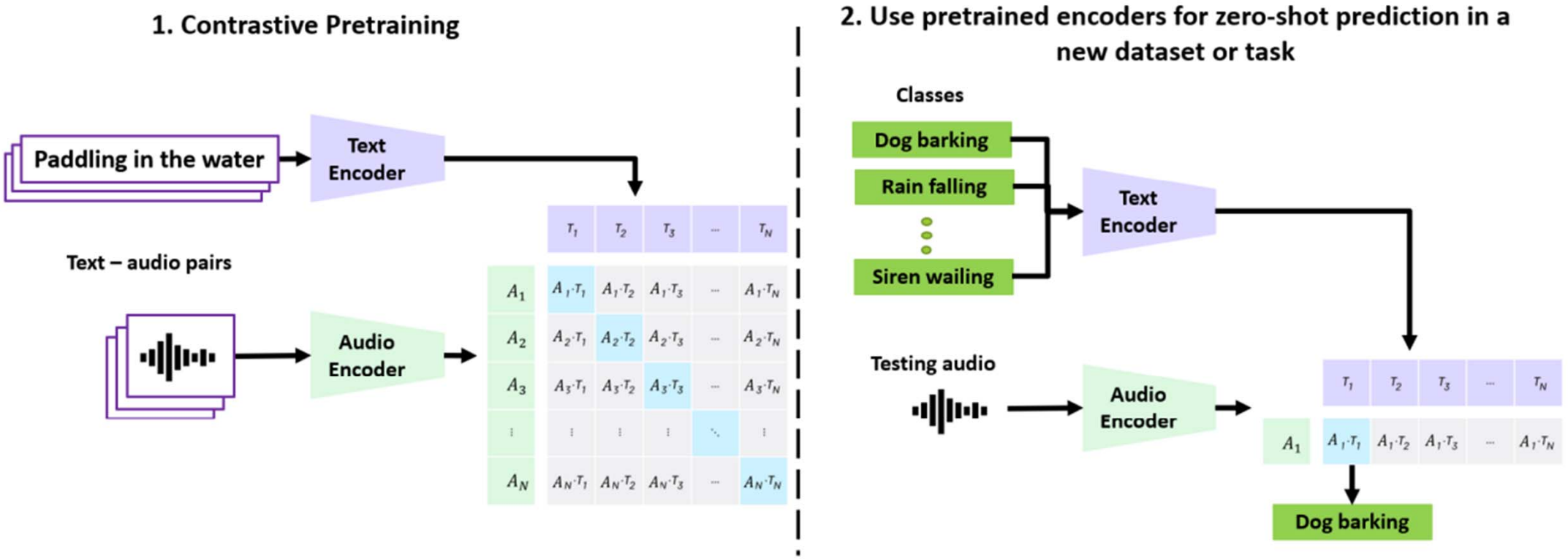
Model	Data	Tasks	Limitations	Code
MusciLDM [221]	Audiostock	TTM	The model is trained on a sample rate of 16 kHz while usually, music holds 44.1 kHz. Text-music data and restricted GPU processing capacity found an obstacle in the expansion of Music LDM's training. Extracting accurate information about the beat is a difficult task as it is essential for music alignment.	✓
TANGO [218]	AudioCaps	TTM	Cannot always perform when trained on a smaller dataset	✗
WavJourney [127]	AudioCaps	TTM	Inflexible to expand the functions. The process of remixing and deteriorating may push the synthetic audio away from the real. Model is time complex when generating the complex audio.	✓
SingSong [229]	1 million audio samples	VIM	The generated instrumentals often exhibit a disparity, with harmonic elements being notably weaker (both in volume and coherence) when compared to their percussive counterparts.	✓
LOAF-M2L [230]	Music Genaration	MTL	_ _	✗
MeLoDy [232]	6.4 Million Samples based on MusicCaps	TTM MTM	Training data mostly contain non-vocal music only Training on LM and DPD on 10-second audio chunks can affect the long generation	✓
MuseCoco [233]	Million MIDI Dataset EMPOIA MetaMidi POP909 Symphony Emotion-gen	TSM	Model primarily focuses on producing symbolic music based on textual descriptions, with little consideration on long sequence modelling. The attribute set discussed in this work only represents a subset of all available music attributes.	✓
LaunchpadGPT [236]	music-frame pairs dataset	PTM	Although LaunchpadGPT partially captures colour similarities, it lacks the ability to effectively learn more structured patterns.	✓
Jukebox [241]	f 1.2 million songs	MTM	- -	✓

(Figure from Latif et al, "Sparks of large audio models: A survey and outlook," arXiv 2023)

Elements of a Text-to-Music Model

- **1. Audio encoder/decoder**
  - So that we can build an LM over the (discrete or continuous) decoder input
- **2. Music LM**
  - Can do unconditional generation
- **3. Text encoder** or **text-music joint embedding space**
  - Provides conditions for the music LM to realize text-to-music generation

Microsoft's CLAP: Text-Audio Joint Embedding Learning

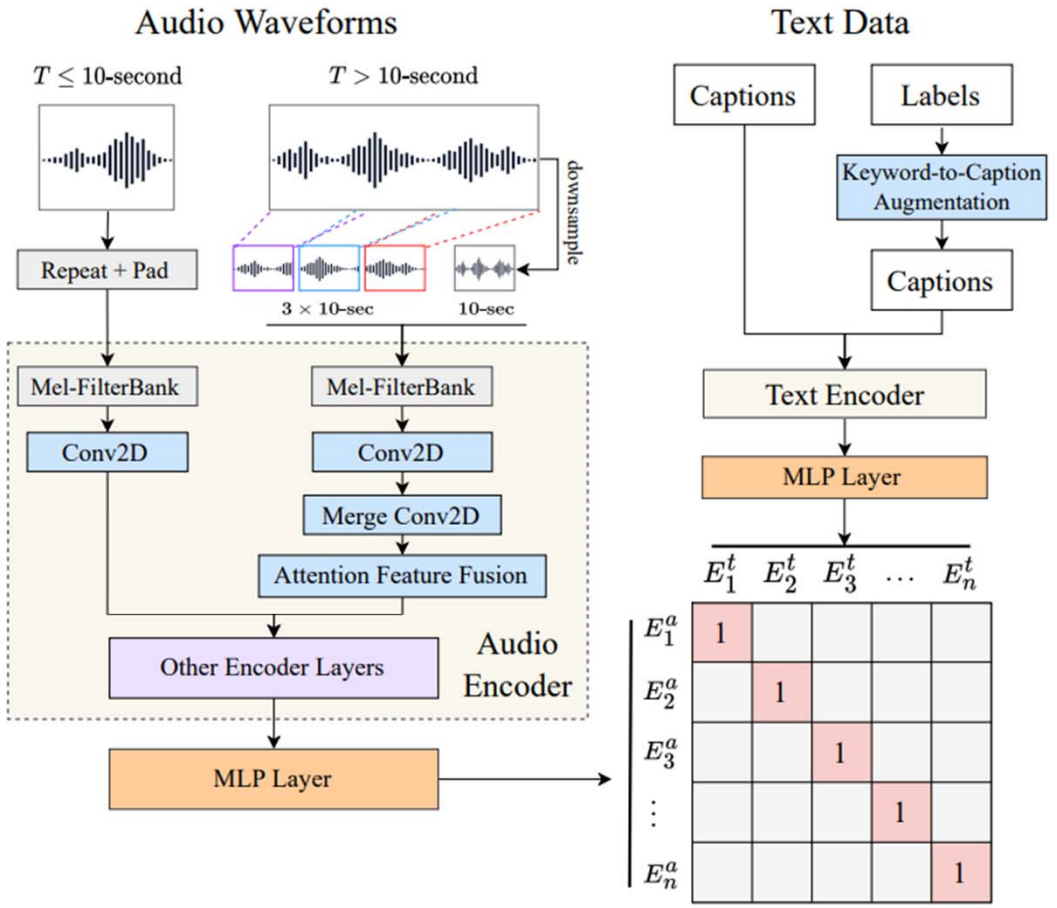


Ref: Elizalde et al, "CLAP: Learning audio concepts from natural language supervision," ICASSP 2023

MILA+UCSD+LAION's CLAP: Text-Audio Joint Embedding Learning

Dataset	Pairs	Audio Durations (hrs)
Clotho [15]	5,929	37.00
SoundDescs [16]	32,979	1060.40
AudioCaps [17]	52,904	144.94
LAION-Audio-630K	633,526	4325.39

Model	AudioCaps (mAP@10)	
	A→T	T→A
PANN+CLIP Trans.	4.7	11.7
PANN+BERT	34.3	44.3
PANN+RoBERTa	37.5	45.3
HTSAT+CLIP Trans.	2.4	6.0
HTSAT+BERT	43.7	49.2
HTSAT+RoBERTa	**45.7**	**51.3**



Ref: Wu et al, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," ICASSP 2023

CLAP

https://github.com/microsoft/CLAP

CLAP weights are downloaded automatically (choose between versions *2022*, *2023*, and *clapcap*), but are also available at: Zenodo or HuggingFace

clapcap is the audio captioning model that uses the 2023 encoders.

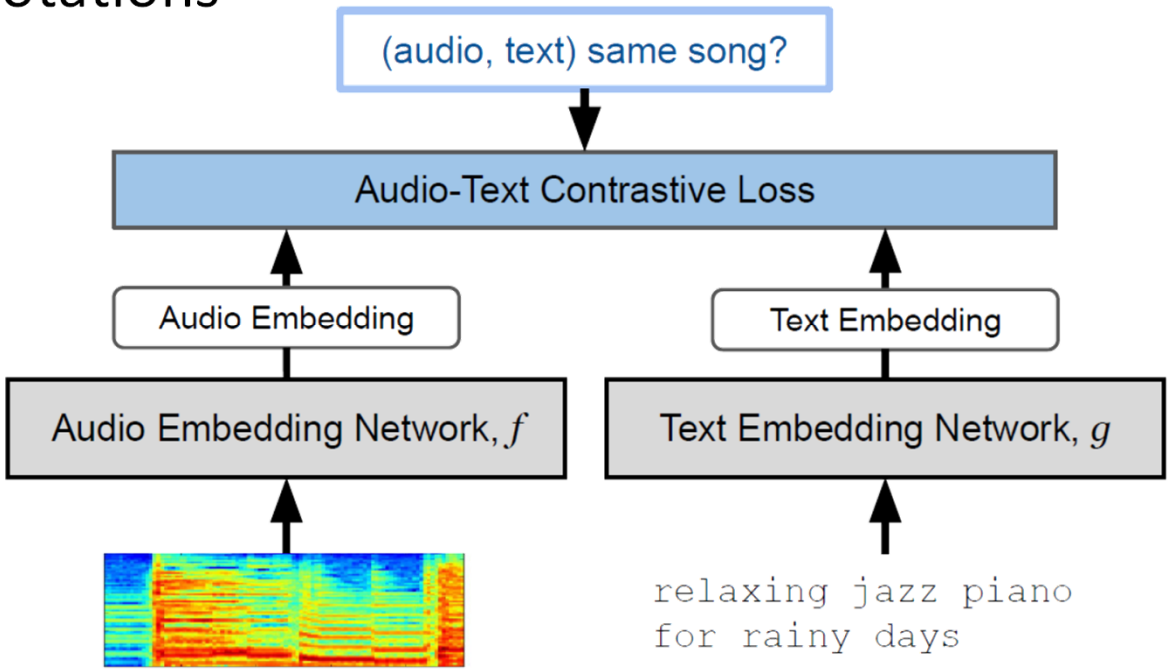
https://github.com/LAION-AI/CLAP

- For general audio less than 10-sec: 630k-audioset-best.pt or 630k-best.pt
- For general audio with variable-length: 630k-audioset-fusion-best.pt or 630k-fusion-best.pt
- For music: music_audioset_epoch_15_esc_90.14.pt
- For music and speech: music_speech_epoch_15_esc_89.25.pt
- For speech, music and general audio: music_speech_audioset_epoch_15_esc_89.98.pt

MuLan: Text-Music Joint Embedding Learning

- MuLan takes the form of a two-tower, joint audio-text embedding model trained using **44 million music recordings** (370K hours) and weakly-associated, free-form text annotations
  - **Audio encoder**: Resnet-50, or audio spectrogram Transformer
  - **Text encoder**: BERT

- However, not open source…



Ref: Huang et al, "MuLan: A joint embedding of music audio and natural language," ISMIR 2022

MuLan: Text-Music Joint Embedding Learning

- MuLan takes the form of a two-tower, joint audio-text embedding model trained using **44 million music recordings** (370K hours) and weakly-associated, free-form text annotations
  - Data cleansing
    - Fine-tune a pre-trained BERT with a binary classification task on a small curated set of 700 sentences, which are manually labeled to be music-descriptive or not
    - However, found that training is surprisingly robust to annotation noise, achieving similar performance using *unfiltered* training text

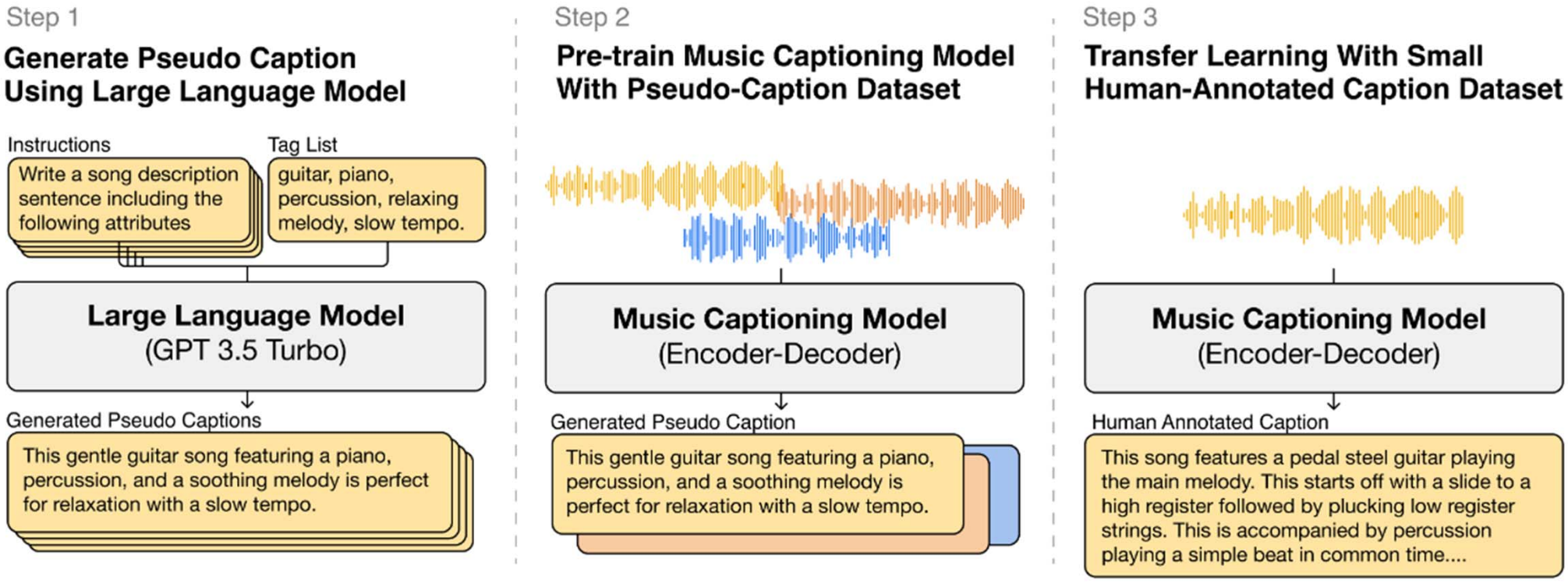
Table 1. Text annotation examples.

Type	Examples
Short-form (SF)	tags like genre, mood, instrument, artist name, song title, album name
Long-form (LF)	'Hip-hop features rap with an electronic backing.' 'The melody is so nostalgic and unforgettable.'
Playlist (PL)	'Feel-good mandopop indie', 'Latin workout' 'Salsa for broken hearts', 'Piano for study'

Ref: Huang et al, "MuLan: A joint embedding of music audio and natural language," ISMIR 2022

LP-MusicCaps: LLM-Based Pseudo Music Captioning

https://github.com/seungheondoh/lp-music-caps



Ref: Doh et al, "LP-MusicCaps: LLM-based pseudo music captioning," ISMIR 2023

LP-MusicCaps: LLM-Based Pseudo Music Captioning

Dataset	# item	Duration (h)	C/A	Avg. Token
General Audio Domain				
AudioCaps [30]	51k	144.9	1	9.0±N/A
LAION-Audio [22]	630k	4325.4	1-2	N/A
WavCaps [18]	403k	7568.9	1	7.8±N/A
Music Domain				
MusicCaps [12]	6k	15.3	1	48.9±17.3
MuLaMCap* [19]	393k	1091.0	12	N/A
LP-MusicCaps-MC	6k	15.3	4	44.9±21.3
LP-MusicCaps-MTT	22k	180.3	4	24.8±13.6
LP-MusicCaps-MSD	514k	4283.1	4	37.3±26.8

Table 3. Comparison of audio-caption pair datasets. C/A stands for the number of caption per audio. *Although we include MuLaMCap in the table for comparison, it is not publicly accessible.

Ref: Doh et al, “LP-MusicCaps: LLM-based pseudo music captioning,” ISMIR 2023

More Datasets for Text-to-Audio in General

Dataset	Hours	Type	Source
Clotho	152	Caption	Drossos et al.
AudioCaps	109	Caption	Kim et al.
MACS	100	Caption	Martín-Morató & Mesaros
WavText5Ks	25	Caption	Deshmukh et al. (2022)
BBC sound effects	481	Caption	https://sound-effects.bbcrewind.co.uk/
Audiostock	43	Caption	https://audiostock.net/se
Filter AudioSet	2084	Label	Gemmeke et al.
ESC-50	3	Label	Piczak
FSD50K	108	Label	https://annotator.freesound.org/fsd/
Sonniss Game Effects	20	Label	https://sonniss.com/gameaudiogdc/
WeSoundEffects	11	Label	https://wesoundeffects.com/
Epidemic Sound	220	Label	https://www.epidemicsound.com/
UrbanSound8K	8	Label	Salamon et al.
LibriTTS	300	Language-free	Zen et al. (2019)
LP-MusicCaps-MSD	4283	Caption	Doh et al. (2023)
LP-MusicCaps-MC	15	Caption	Doh et al. (2023)

Ref: “HarmonyLM: Advancing unified large-scale language modeling for audio and music generation,” ICLR 2024 submission

MusicLM

https://google-research.github.io/seanet/musiclm/examples/

- Given a **text description of music**, generate **waveforms** directly
  - "Piano for study" | "Hip-hop song with violin solo" | "Latin workout" | "Feel-good mandopop indie" | "Salsa for broken hearts"

- **Story mode**

Text prompts

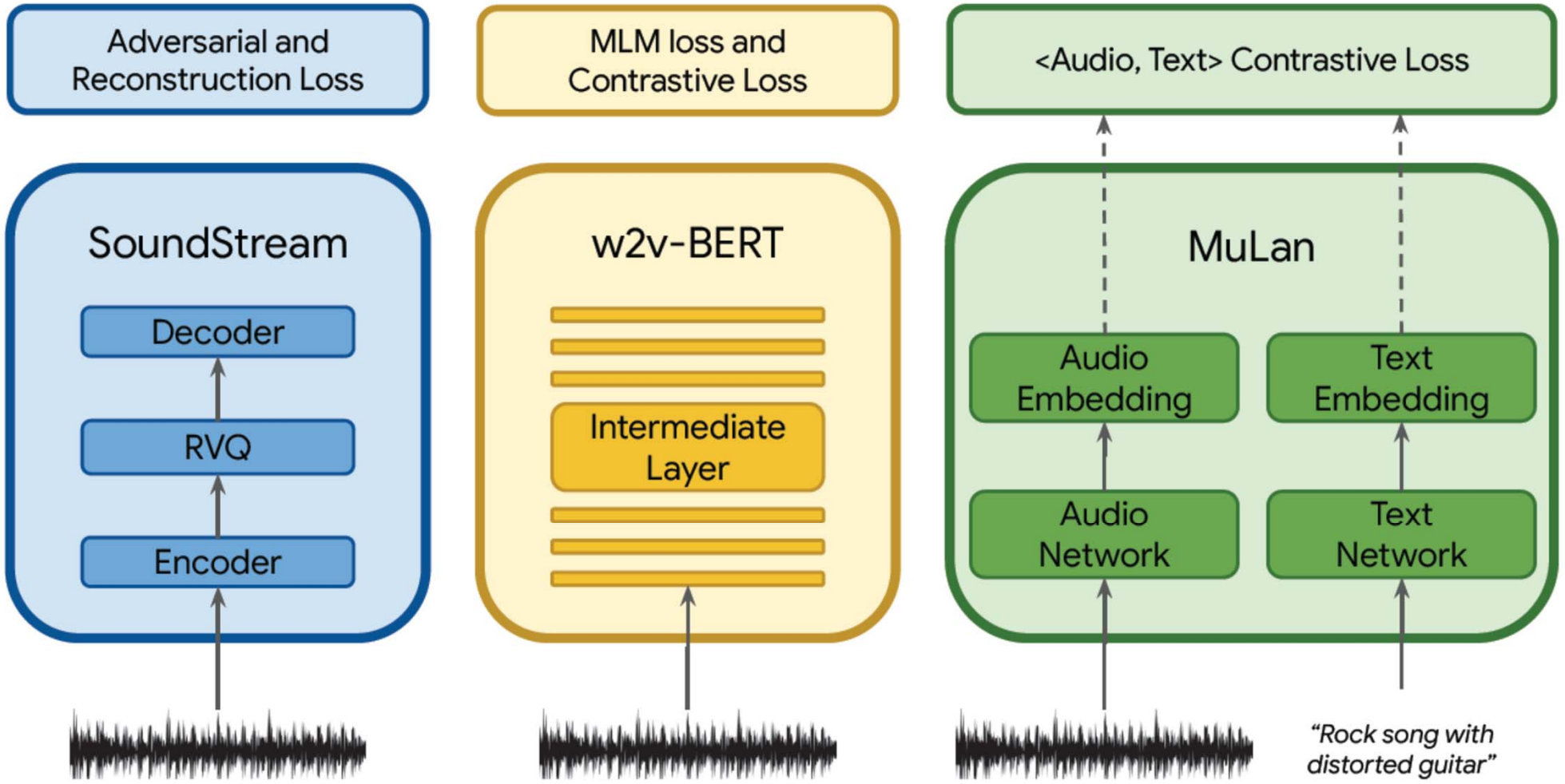
time to meditate (0:00-0:15)
time to wake up (0:15-0:30)
time to run (0:30-0:45)
time to give 100% (0:45-0:60)

- **Text and melody conditioning**
  - Text-conditioned continuation of the provided melody input

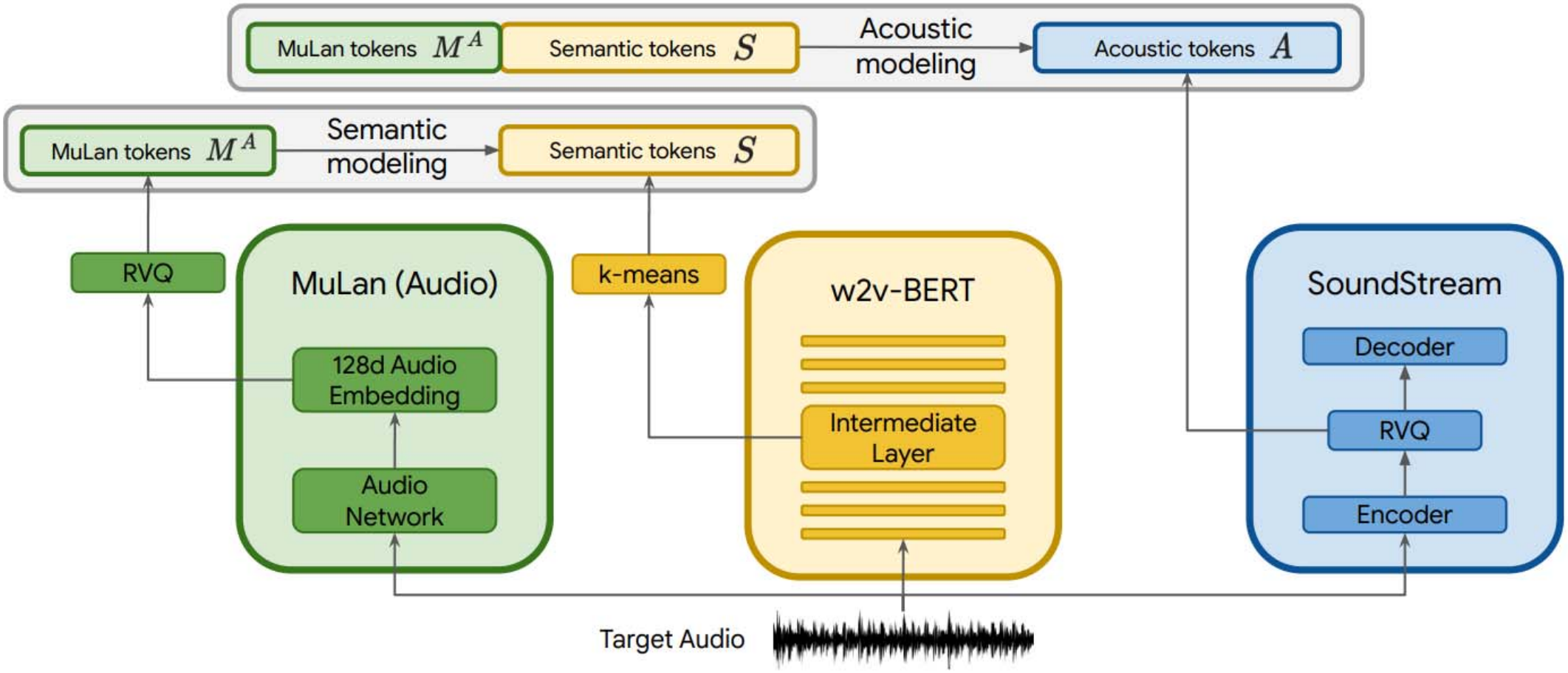
- **Image-to-music**
  - Painting caption conditioning

Ref: Agostinelli et al, "MusicLM: Generating music from text," arXiv 2023

MusicLM's Pretrained Networks



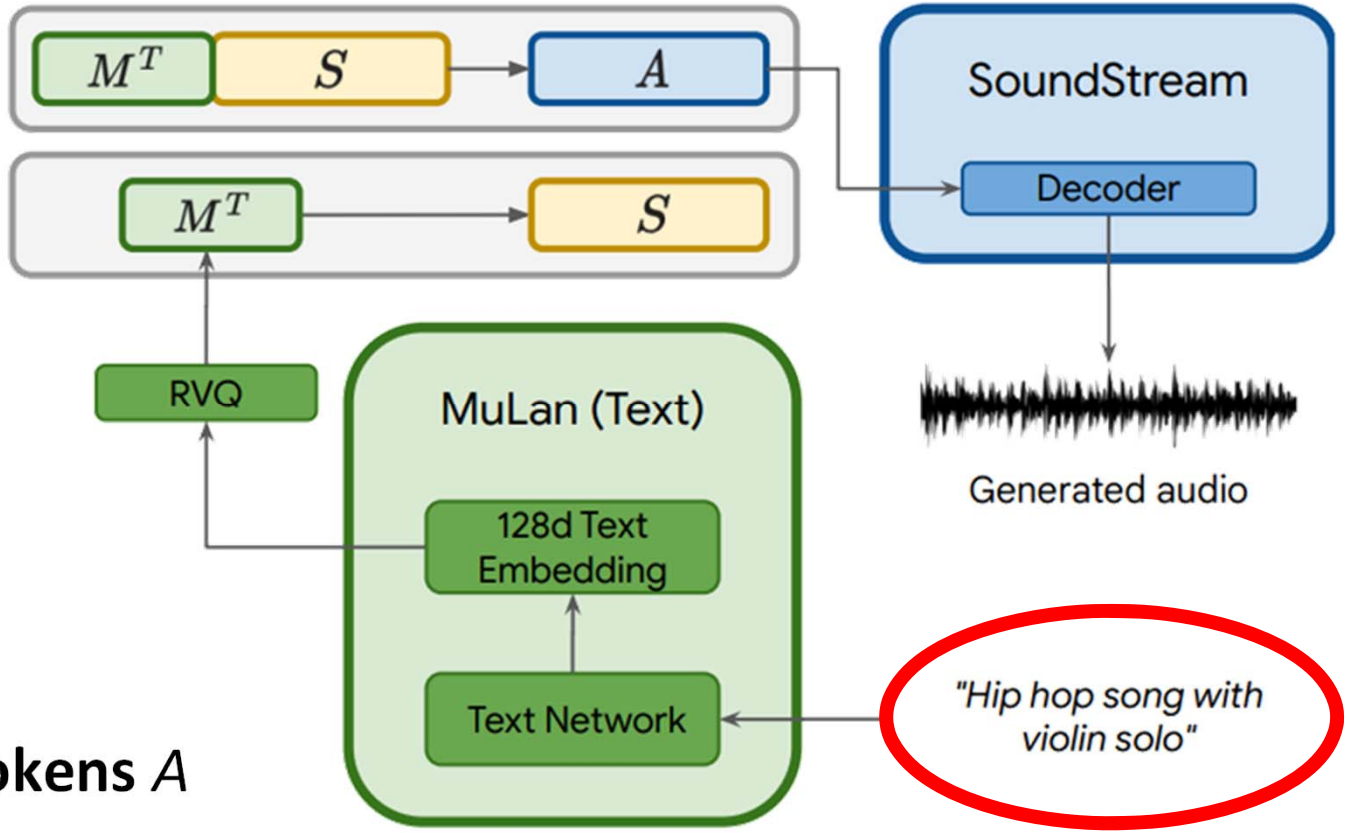
MusicLM Training Procedure



Ref: Agostinelli et al, "MusicLM: Generating music from text," arXiv 2023

MusicLM Inference Procedure

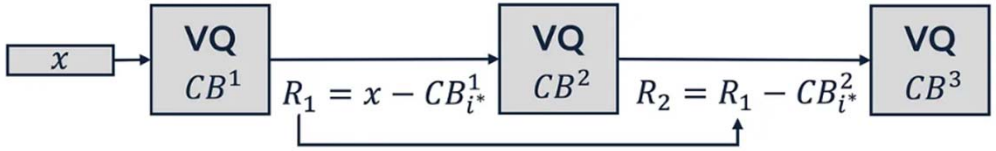
- Given **text tokens** M^T from the input text description
- Generate high-level **semantic tokens** *S* that model the long-term structure of the music to be generated
- Then generate **acoustic tokens** *A* that model the fine details



Ref: Agostinelli et al, "MusicLM: Generating music from text," arXiv 2023

Jukebox vs MusicLM

- **Jukebox**
  - 3 layers of **acoustic tokens**: high, mid, bottom
  - The three layers operate in **different** temporal resolution



- **MusicLM**
  - 12 layers of SoundStream RVQ tokens (each with a vocabulary size of 1024)
  - All the 12 layers operate in the **same** temporal resolution (50 Hz)
  - **600** such **acoustic tokens** per second
  - Plus semantic tokens from w2v-BERT (codebook size also 1024)
  - **25 semantic tokens** per second

Issues of MusicLM

- Not open source; but there are unofficial implementations
  - https://github.com/lucidrains/musiclm-pytorch
  - https://github.com/zhvng/open-musiclm
- Slow (one second 625 tokens; i.e., 625 times of autoregressive sampling)
- Limited controllability
  - The main control signal is the text description (which is usually about the genre and instrumentation of music)
  - Advanced users may want to control, e.g., the **melody** of the singing vocals and the **grooving** (rhythm) of the backing instrumentals

Ref: Agostinelli et al, "MusicLM: Generating music from text," arXiv 2023

Issues of MusicLM

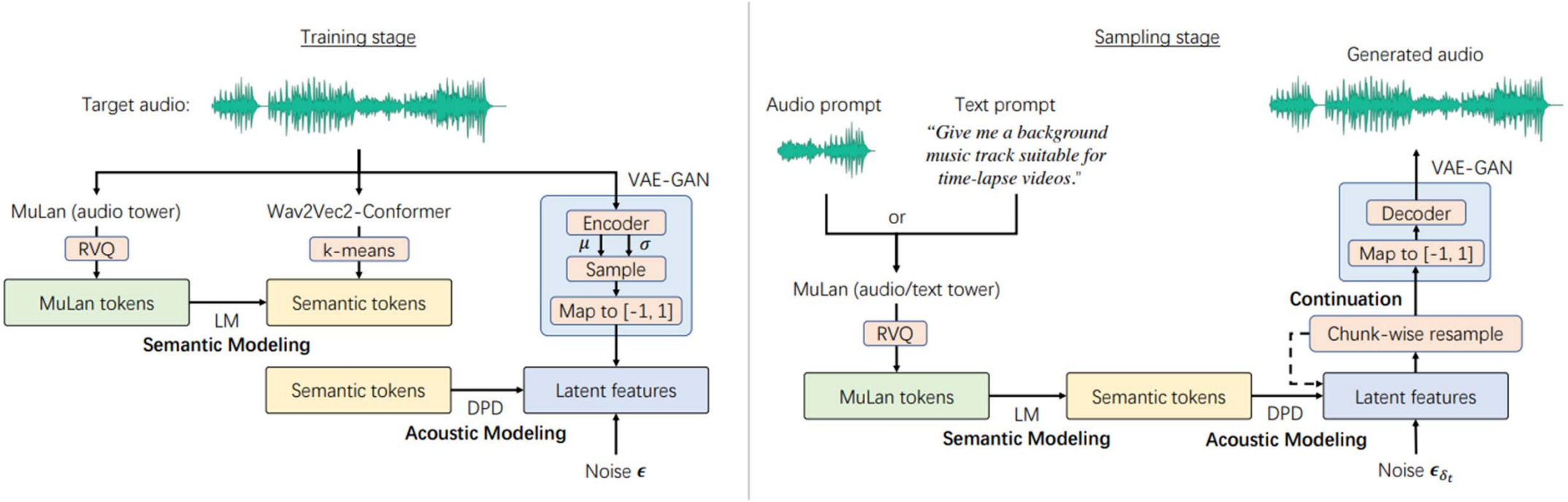
- Text-to-music models are in general costly
  - (Latif 2023 arXiv): “AudioGPT with 137 billion parameters requires approximately 1 million kilowatthours (kWh), which is approximately $137 million USD. Similarly, the training cost of the state-of-the-art AudioPaLM with 530 billion parameters is approximately $530 million USD.”

Ref: Latif et al, “Sparks of large audio models: A survey and outlook,” arXiv 2023

MeLoDy

https://Efficient-MeLoDy.github.io/

- Replace the LM for acoustic tokens in MusicLM by **latent diffusion**

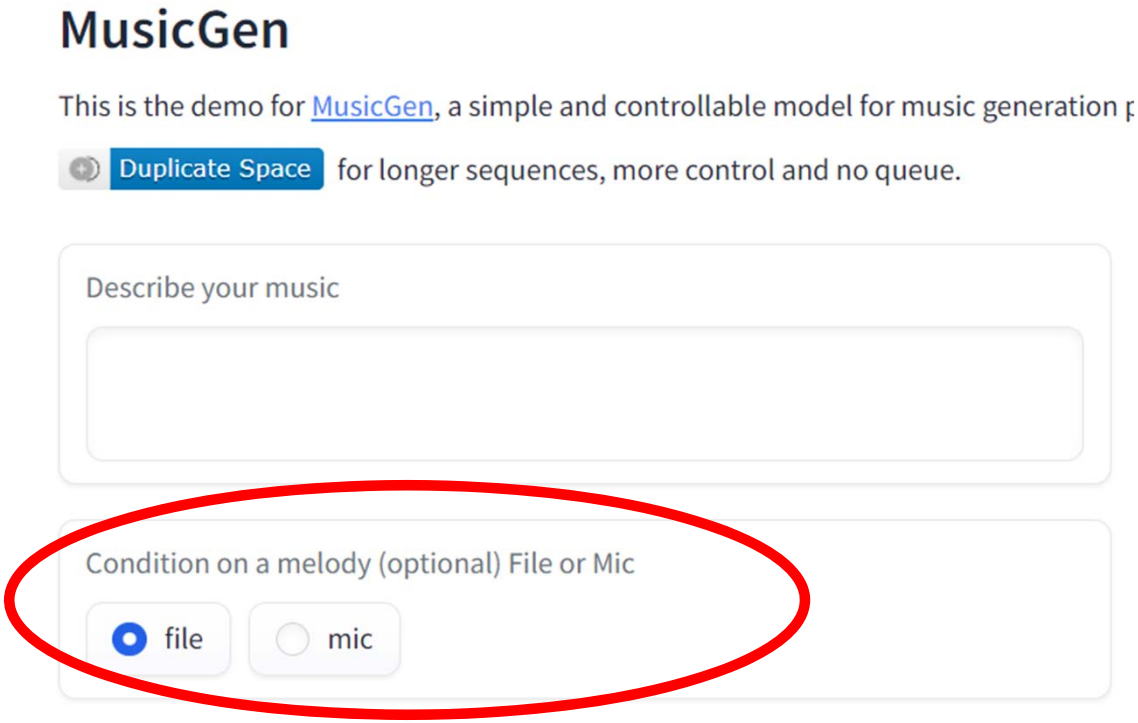


Ref: Lam et al, "Efficient neural music generation," arXiv 2023

MusicGen

https://huggingface.co/spaces/facebook/MusicGen
https://github.com/facebookresearch/audiocraft

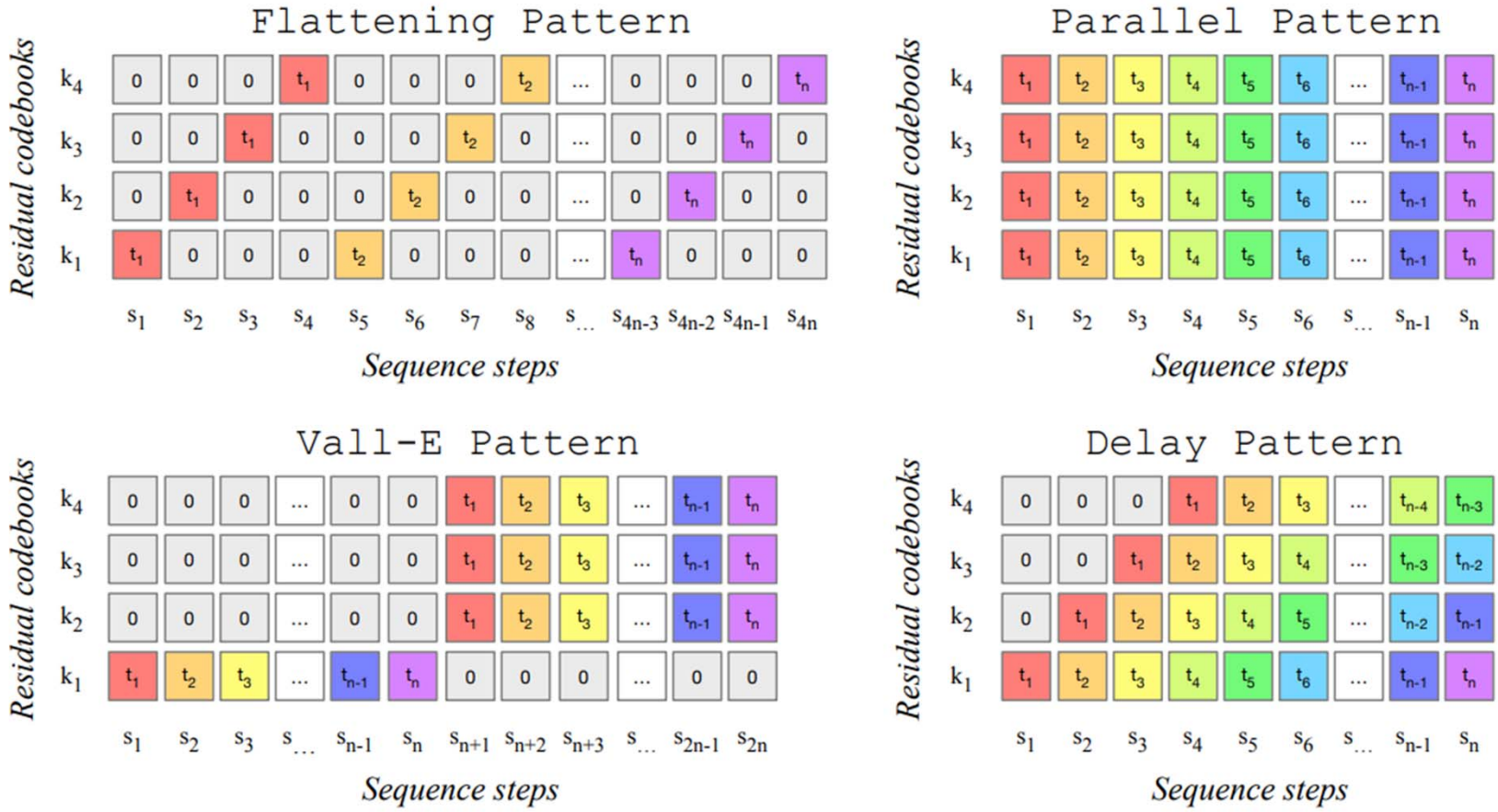
- **Melody conditioning:** Not just as a prompt
  - Tried using **chromagram** of the melody as input but this leads to overfitting
  - Introduce an **information bottleneck** by choosing the **dominant time-frequency bin** in each time step instead



Ref: Copet et al, "Simple and controllable music generation," arXiv 2023

MusicGen

- Compound word-like **parallel prediction** of the RVQ tokens



MusicGen

- **No text-audio joint embedding**; simply use a **pure text encoder** such as T5 or Flan-T5
- And **no semantic tokens**; just RVQ acoustic tokens → simple architecture
- And no diffusion

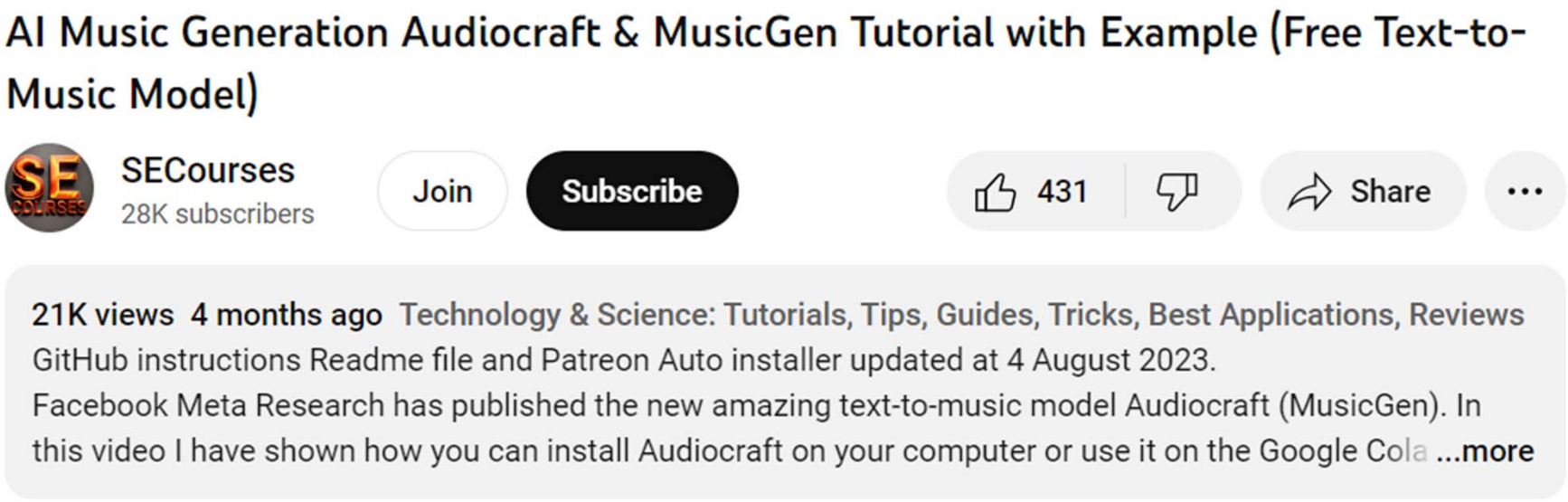
Table A.1: Text encoder results. We report results for T5, Flan-T5, and CLAP as text encoders. We observe similar results for T5 and Flan-T5 on all the objective metrics. CLAP encoder performs consistently worse on all the metrics but CLAP score.

	MusicCaps Test Set				
Model	FAD_{vgg} ↓	KL ↓	$CLAP_{scr}$ ↑	Ovl. ↑	Rel. ↑
T5	**3.12**	**1.29**	0.31	84.89±1.78	**82.52±1.31**
Flan-T5	3.36	1.30	**0.32**	**86.25±1.75**	80.83±1.88
CLAP	4.23	1.53	**0.32**	79.79±1.76	77.28±1.54

Ref: Copet et al, "Simple and controllable music generation," arXiv 2023

MusicGen Tutorial

https://github.com/FurkanGozukara/Stable-Diffusion/blob/main/Tutorials/AI-Music-Generation-Audiocraft-Tutorial.md



https://www.youtube.com/watch?v=v-YpvPkhdO4

JEN-1

https://www.futureverse.com/research/jen/demos/jen1

- No quantization
- Use **latent diffusion models** to generate continuous embeddings

High Fidelity Neural Audio Latent Representation. To facilitate the training on limited computational resources without compromising quality and fidelity, our approach JEN-1 employs a high-fidelity audio autoencoder $\mathcal{E}$ to compress original audio into latent representations $\boldsymbol{z}$. Formally, given a two-channel stereo audio $\boldsymbol{x} \in \mathbb{R}^{L\times 2}$, the encoder $\mathcal{E}$ encodes $\boldsymbol{x}$ into a latent representation $\boldsymbol{z} = \mathcal{E}(\boldsymbol{x})$, where $\boldsymbol{z} \in \mathbb{R}^{L/h\times c}$. L is the sequence length of given music, h is the hop size and c is the dimension of latent embedding. While the decoder reconstructs the audio $\tilde{\boldsymbol{x}} = \mathcal{D}(\boldsymbol{z}) = \mathcal{D}(\mathcal{E}(\boldsymbol{x}))$ from the latent representation. Our audio compression model is inspired and modified based on previous work (Zeghidour et al., 2021; Défossez et al., 2022), which consists of an autoencoder trained by a combination of a reconstruction loss over both time and frequency domains and a patch-based adversarial objective operating at different resolutions. This ensures that the audio reconstructions are confined to the original audio manifold by enforcing local realism and avoids muffled effects introduced by relying solely on sample-space losses with L1 or L2 objectives. Unlike prior endeavors (Zeghidour et al., 2021; Défossez et al., 2022) that employ a quantization layer to produce the discrete codes, our model directly extracts the continuous embeddings without any quality-reducing loss due to quantization. This utilization of powerful autoencoder representations enables us to achieve a nearly optimal balance between complexity reduction and high-frequency detail preservation, leading to a significant improvement in music fidelity.

Ref: Li et al, "JEN-1: Text-guided universal music generation with omnidirectional diffusion models," arXiv 2023

UniAudio: Unified Model for Audio Generation

- UniAudio supports 11 audio generation tasks
  - **TTS**, **VC**, **SE** (speech enhancement), **TSE** (target speech extraction), **SVS**, **TT-Sound** (text-to-sound generation), **TT-Music** (text-to-music generation), **A-Edit** (audio edit), **SD** (speech dereverberation), **I-TTS** (instructed TTS), **S-Edit** (speech edit)

Model	TTS	VC	SE	TSE	SVS	TT-Sound	TT-Music	A-Edit	SD	I-TTS	S-Edit
YourTTS (Casanova et al., 2022)	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
VALL-E (Wang et al., 2023a)	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
MusicLM (Wang et al., 2023a)	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
SPEARTTS (Kharitonov et al., 2023)	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
NaturalSpeech2 (Shen et al., 2023)	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗
Make-A-Voice (Huang et al., 2023b)	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗
Maga-TTS (Jiang et al., 2023)	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
VoiceBox (Le et al., 2023)	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
AudioLDM2 (Liu et al., 2023b)	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗
SpeechX (Wang et al., 2023c)	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓
UniAudio (ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Ref: Yang et al, "UniAudio: An audio foundation model toward universal audio generation," ICLR 2024 submission

UniAudio

- Training of UniAudio is scaled up to 165K hours of audio and **1B** parameters

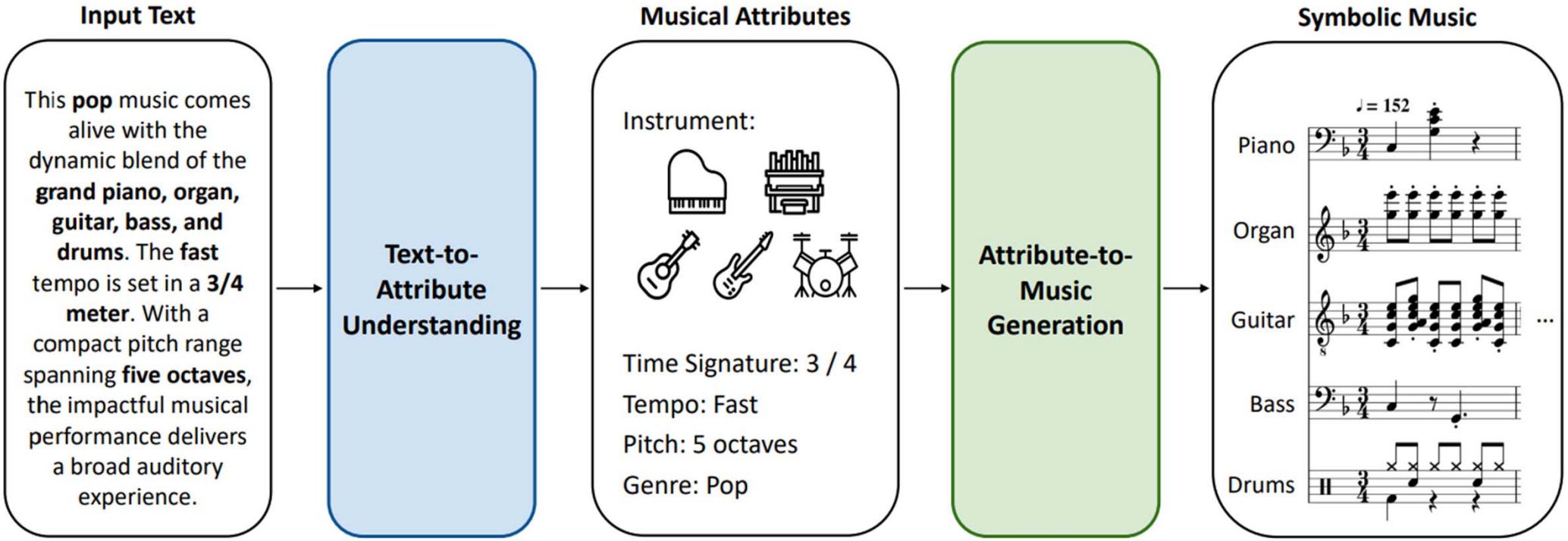
Table 6: Data Statistics

Dataset	Type	Annotation	Volume (hrs)
Training			
LibriLight (Kahn et al., 2020)	speech	-	60k
LibriTTS Zen et al. (2019)	speech	text	1k
MLS (Pratap et al., 2020)	speech	-	20k
AudioSet (Gemmeke et al., 2017)	sounds	-	5.8k
AudioCaps Kim et al. (2019)	sounds	text description	500
WavCaps Mei et al. (2023)	sounds	text description	7k
Million Song Dataset McFee et al. (2012)	music	text description	7k
OpenCPOP Wang et al. (2022)	singing	text, MIDI	5.2h
OpenSinger Huang et al. (2021a)	singing	text, MIDI	50h
AISHELL3 Shi et al. (2020)	speech	text	85h
PromptSpeech Guo et al. (2023)	speech	text, instruction	200h
openSLR26,openSLR28 Ko et al. (2017)	Room Impulse Response	-	100h
Test Only			
LibriSpeech-test clean Panayotov et al. (2015)	speech	text	8h
VCTK Veaux et al. (2017)	speech	text	50h
TUT2017 Task1 Mesaros et al. (2017)	Noise	-	10h
Cloth Drossos et al. (2020)	Sounds	text description	3h
MusicCaps Agostinelli et al. (2023)	Music	text description	15h
M4SingerZhang et al. (2022)	singing	text, MIDI	1h

Ref: Yang et al, “UniAudio: An audio foundation model toward universal audio generation,” ICLR 2024 submission

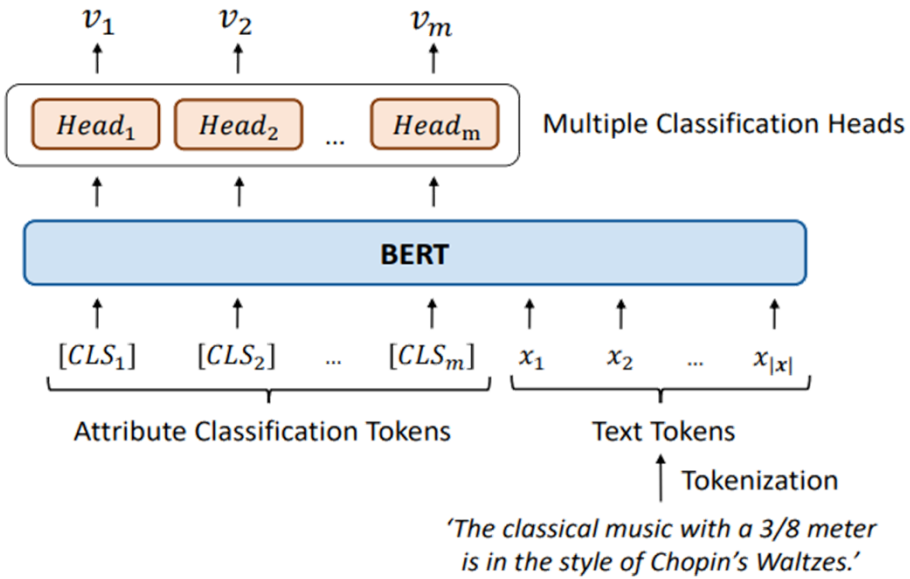
MuseCoCo: Text-to-MIDI

https://ai-muzic.github.io/musecoco/

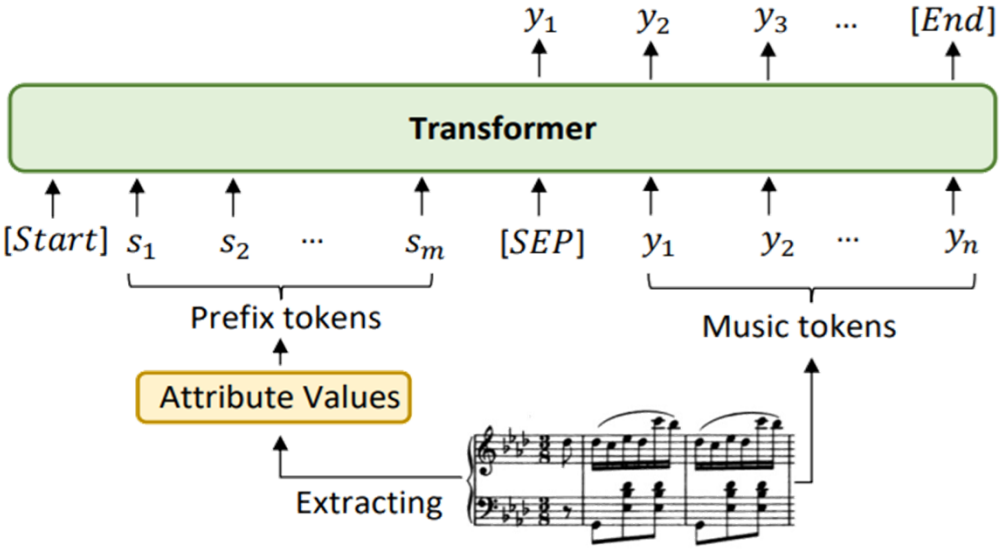


Ref: Lu et al, "MuseCoco: Generating symbolic music from text," arXiv 2023

MuseCoCo



(a) Text-to-attribute understanding



(b) Attribute-to-music generation

Table 7: Comparison of different control methods. Musicality reflects the quality of the generated music. Average attribute control accuracy represents the control accuracy over all attributes, which can reflect controllability.

Method	Musicality ↑	Average attribute control accuracy (%) ↑
Embedding	2.97 ± 0.91	36.94
Conditional LayerNorm	3.11 ± 1.02	47.46
Prefix Control	$\mathbf{3.15} \pm 1.02$	**67.40**

Table 1: Musical attribute descriptions.

Type	Attribute	Description
Objective	Instrument	played instruments in the music clip
	Pitch	the number of octaves covering all pitches in one music
	Rhythm Danceability	whether the piece sounds danceable
	Rhythm Intensity	the intensity of the rhythm
	Bar	the total number of bars in one music clip
	Time Signature	the time signature of the music clip
	Key	the tonality of the music clip
	Tempo	the tempo of the music clip
	Time	the approximate time duration of the music clip
Subjective	Artist	the artist (style) of the music clip
	Genre	the genre of the music clip
	Emotion	the emotion of the music clip

Table 3: Statistics of the used datasets.

Dataset	#MIDI
MMD [36]	1,524,557
EMOPIA [16]	1,078
MetaMidi[37]	612,088
POP909 [38]	909
Symphony [39]	46,360
Emotion-gen	25,730
Total (after filtering)	947,659

Table 8: Comparison of different model sizes in the attribute-to-music generation stage. Average objective attribute control accuracy represents the control accuracy over objective attributes, which can reflect controllability.

Model Size	Layers	d_{model}	Parameters	Average objective attribute control accuracy (%) ↑
large	16	1024	203M	83.63
xlarge	24	2048	1.2B	**87.15**

Ref: Lu et al, “MuseCoco: Generating symbolic music from text,” arXiv 2023

Recap: Elements of a Text-to-Music Model

- **1. Music LM/generator**
  - Transformer-based: operating on discrete tokens
  - Diffusion-based: operating on continuous latents
- **2a. Text encoder** that provides conditions for the music LM/generator
- **2b.** (Optional) **text-music joint embedding space**
  - Not needed if there is a strong text encoder
- **3. Audio encoder/decoder**
  - VQVAE-based (in particular **audio codec** models) if it's for Transformer-based LMs
  - Not necessarily VQVAE-based if it's for diffusion based generator
  - Audio encoder/decoder is not needed if it's for MIDI generation

What's Next

- Better conditioning mechanism
  - Chord progression
  - MIDI
  - Lyrics
  - (like Control Net)
- Source separated tokens
- Efficient fine-tuning