

2024 edition

Deep Learning for Music Analysis and Generation

# Text-to-music Generation

(text → audio)



**Yi-Hsuan Yang** Ph.D.  
[yhyangtw@ntu.edu.tw](mailto:yhyangtw@ntu.edu.tw)

# Reference: ISMIR 2024 Tutorial

<https://ismir2024.ismir.net/tutorials>

---

## Connecting Music Audio and Natural Language

Seung Heon Doh, Ilaria Manco, Zachary Novack, JongWook Kim and Ke Chen

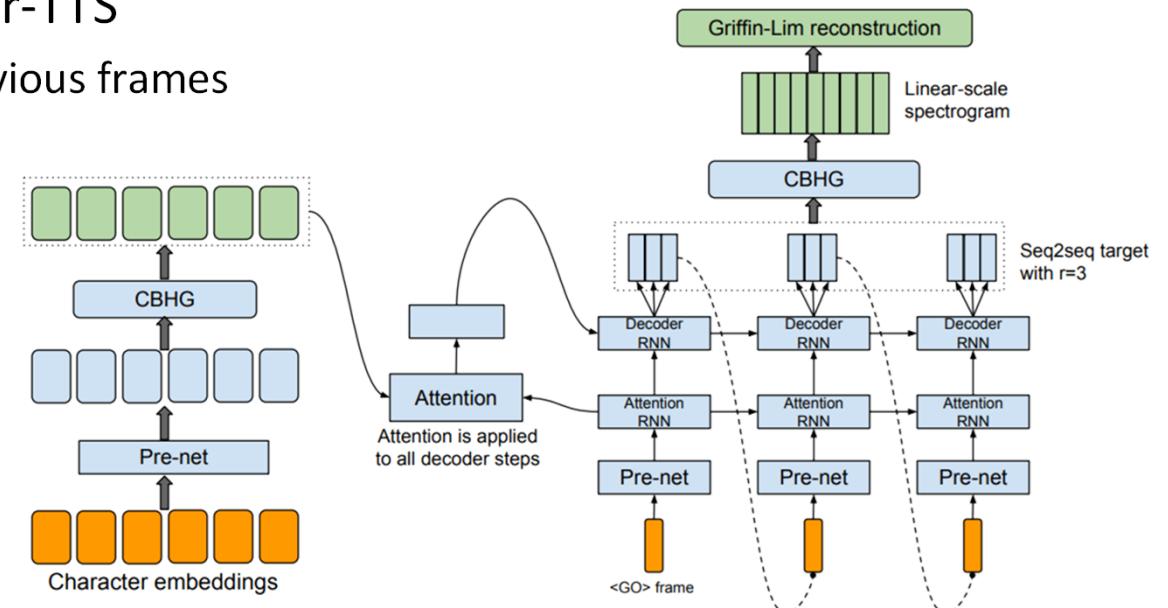
| Timeline      | Content  |
|---------------|--|
| 09:00 - 09:05 | Chapter 1. Introduction / Motivation [SeungHeon]                   |
| 09:05 - 09:40 | Chapter 2. Overview of Language Models [Jong Wook]                 |
| 09:40 - 10:30 | Chapter 3. Music Description [Ilaria]                              |
| 10:30 - 11:00 | Coffee Break   |
| 11:00 - 11:40 | Chapter 4. Text-to-Music Retrieval For Music Discovery [SeungHeon] |
| 11:40 - 12:20 | Chapter 5. Text-to-Music Generation for New Sound [Zachary & Ke]   |
| 12:20 - 12:30 | Chapter 6. Conclusion  |

# 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

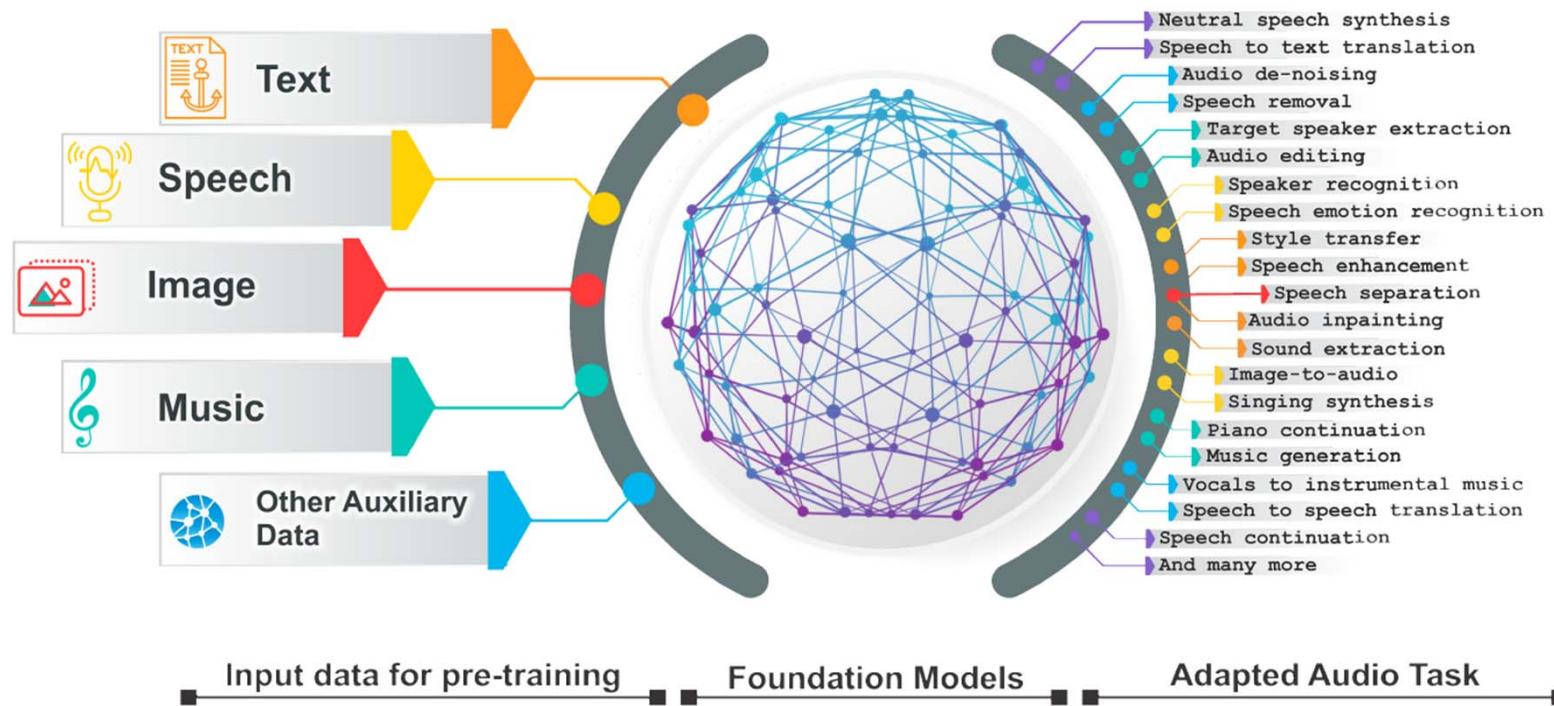
- MIDI events are by nature **discrete**; easy to build an LM for MIDI
- Audio files (waveforms or spectrograms) are **continuous** so it's trickier
- We can still build an autoregressive model for audio
  - Example: TacoTron or Transformer-TTS
    - Predict the next frame given the previous frames



(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**

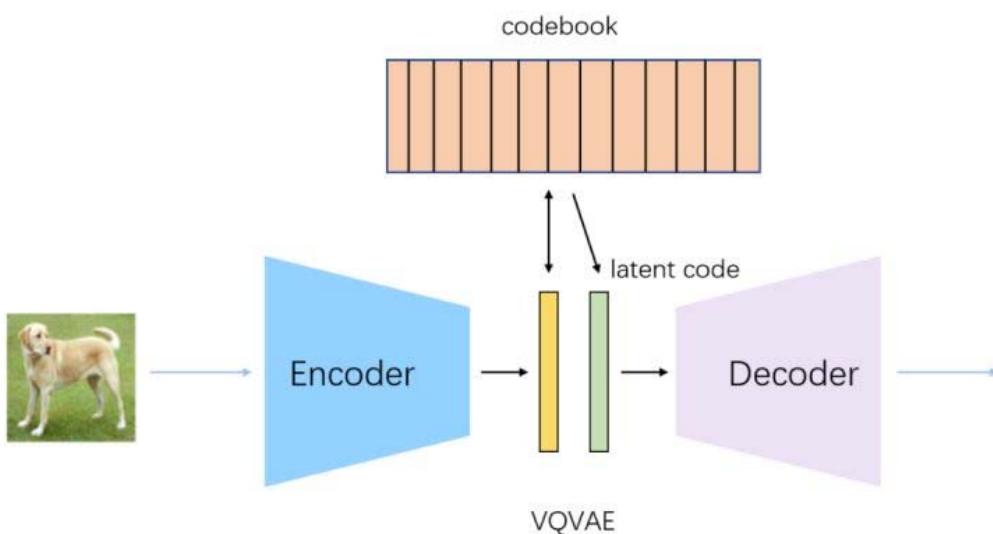


(Figure from Latif et al, “Sparks of large audio models: A survey and outlook,” arXiv 2023)

# Language Modeling (LM) in Music Audio

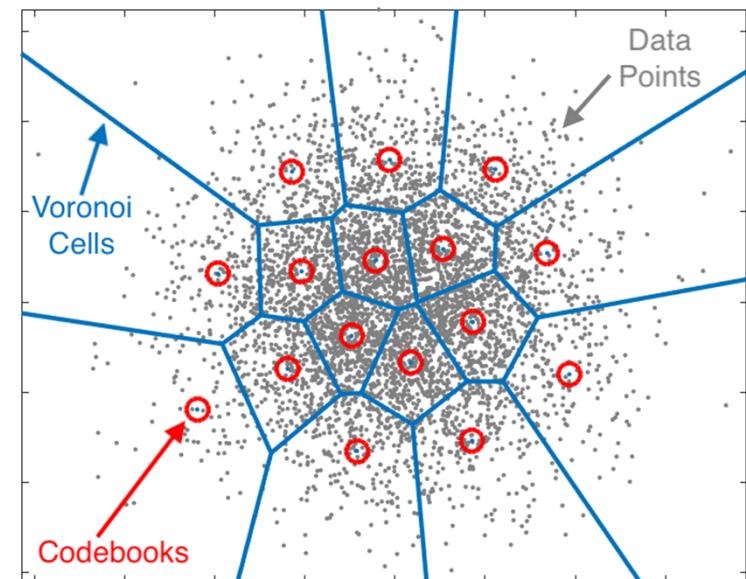
- **Audio domain**

- use some encoder/decoder to convert audio into “**audio tokens**”
- **for example: VQ-VAE**
- then build an LM



<https://zhuanlan.zhihu.com/p/388299884>

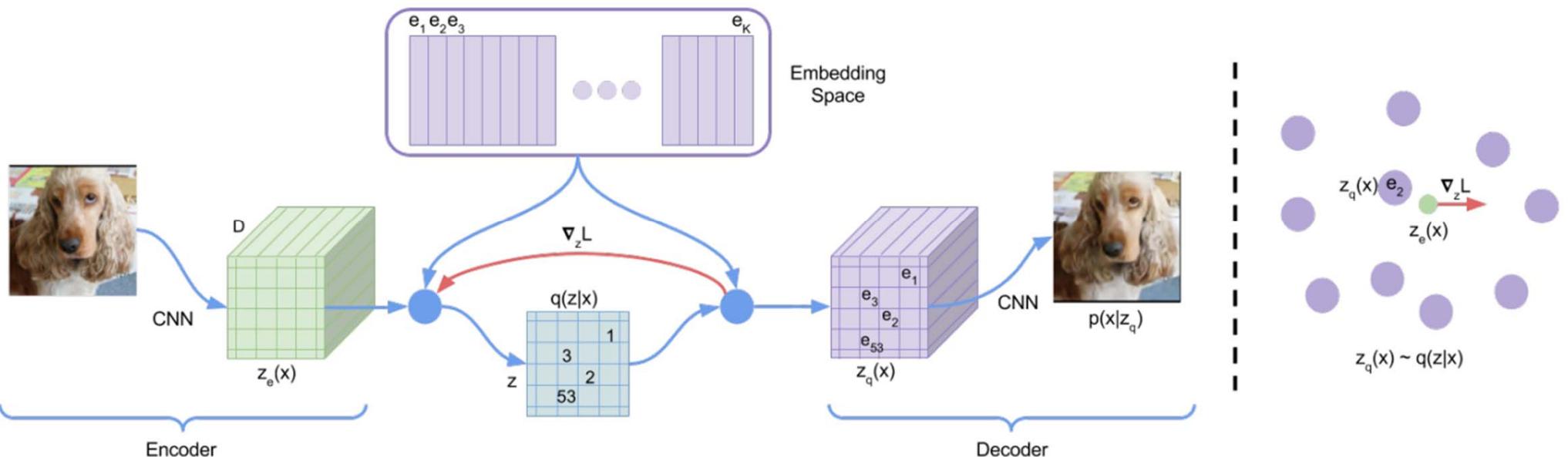
Illustration of vector quantization (VQ)



<https://towardsdatascience.com/improving-vector-quantization-in-vector-quantized-variational-autoencoders-vq-vae-915f6814b5ce>

# Image Tokenization by VQVAE

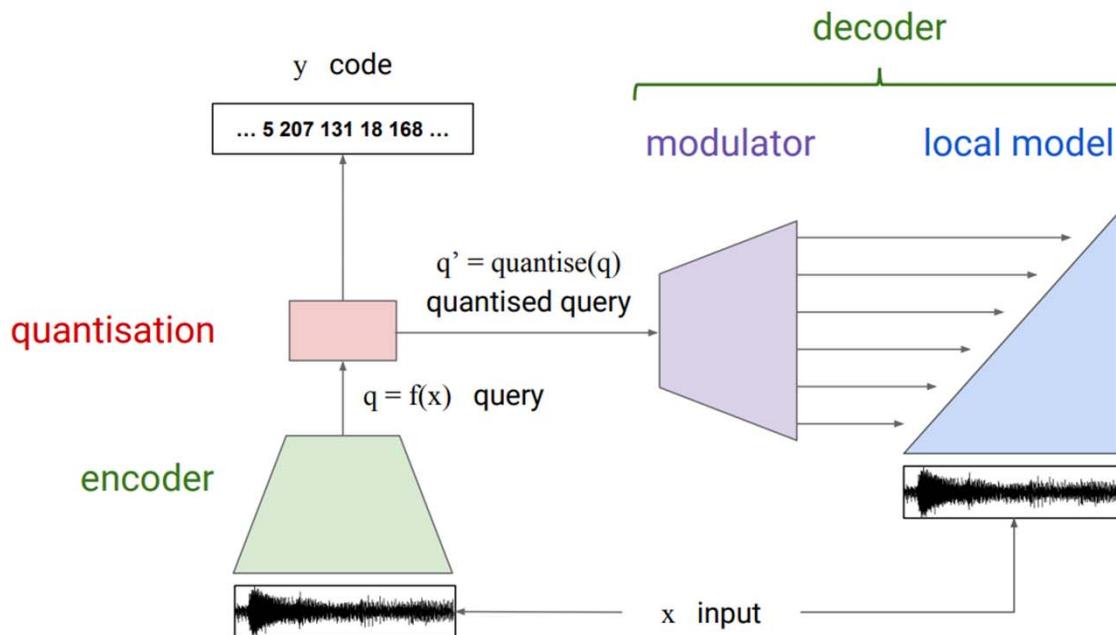
- We can obtain **discrete tokens of continuous data** by doing **vector quantization** at the output of an image **encoder** (Oord et al., 2017)



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

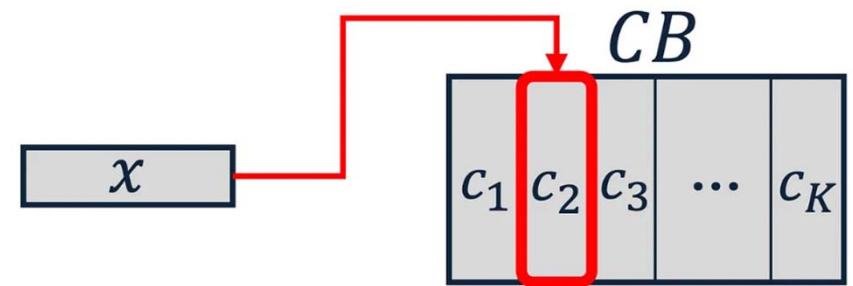
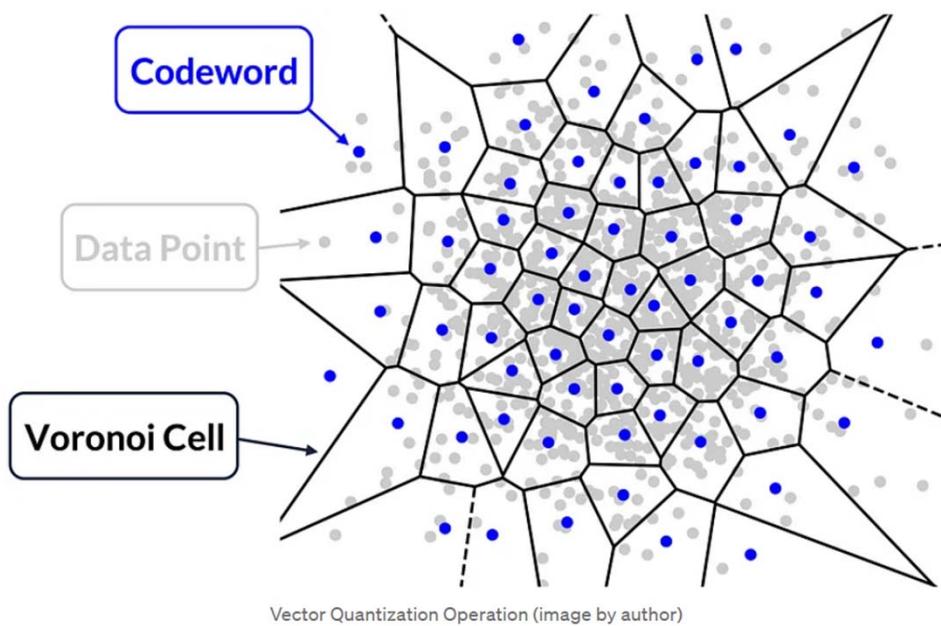
# Audio Tokenization by VQVAE

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



# Vector Quantization

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



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

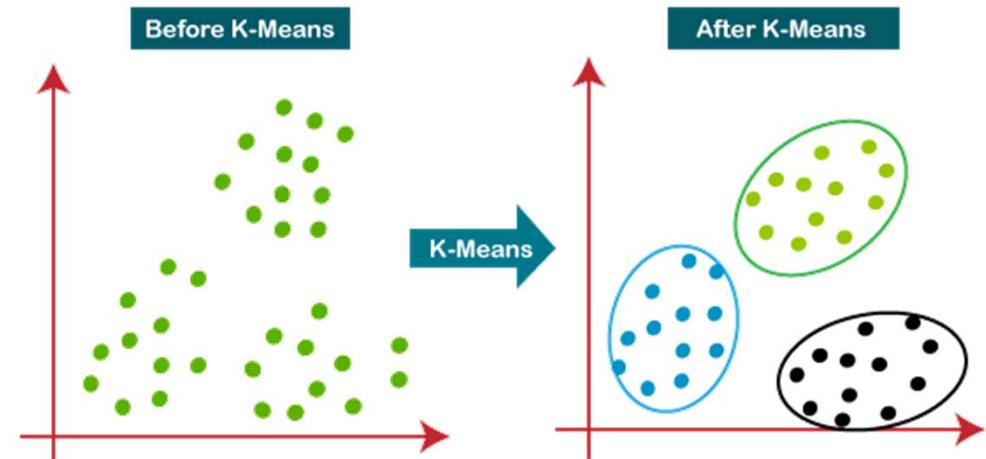
in this case :  $x_{\text{quantized}} = c_2$

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

# K-Means Clustering

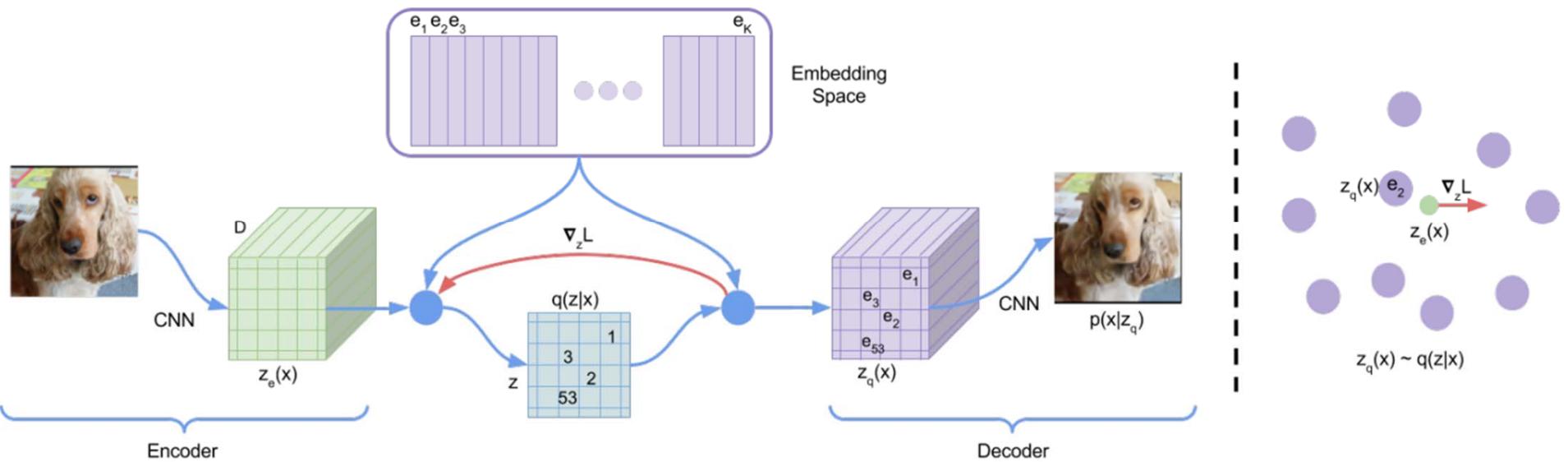
<https://towardsdatascience.com/clear-and-visual-explanation-of-the-k-means-algorithm-applied-to-image-compression-b7fdc547e410>

- K-Means algorithm
  - Initializing a set of cluster centroids
  - Assigning observations to clusters
  - Updating the clusters
- Difference between K-Means and VQVAE
  - In K-Means, the feature space is *given*
  - In VQVAE, the feature space is *learned*



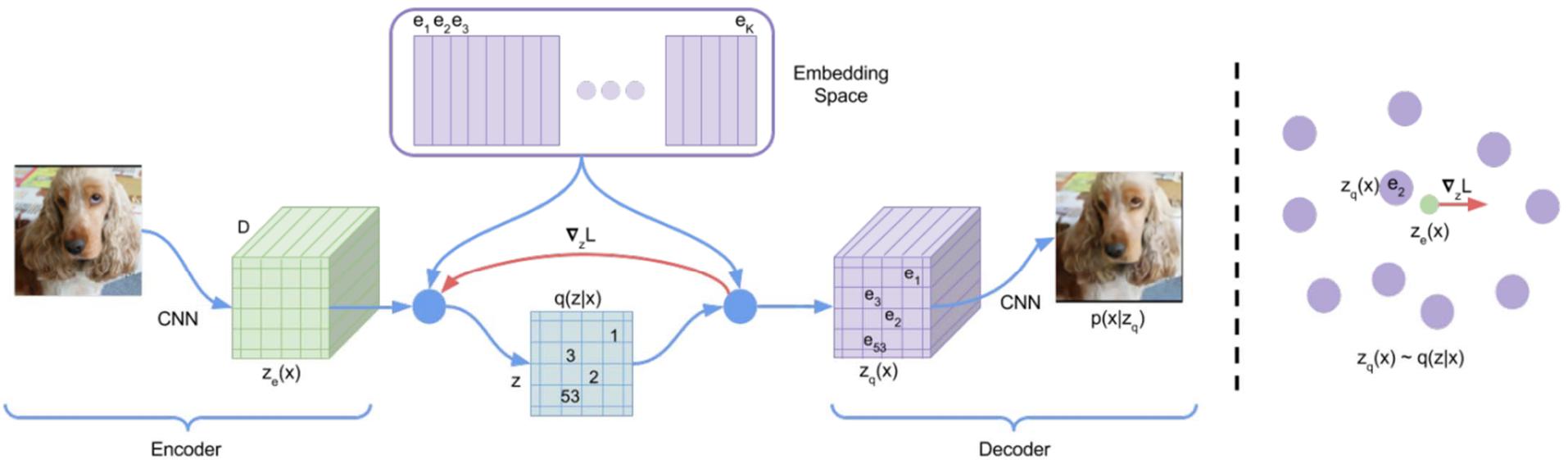
<https://keytodatascience.com/k-means-clustering-algorithm/>

# VQVAE



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

# VQVAE



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

**Reconstruction loss**

Optimizes the encoder  
and decoder

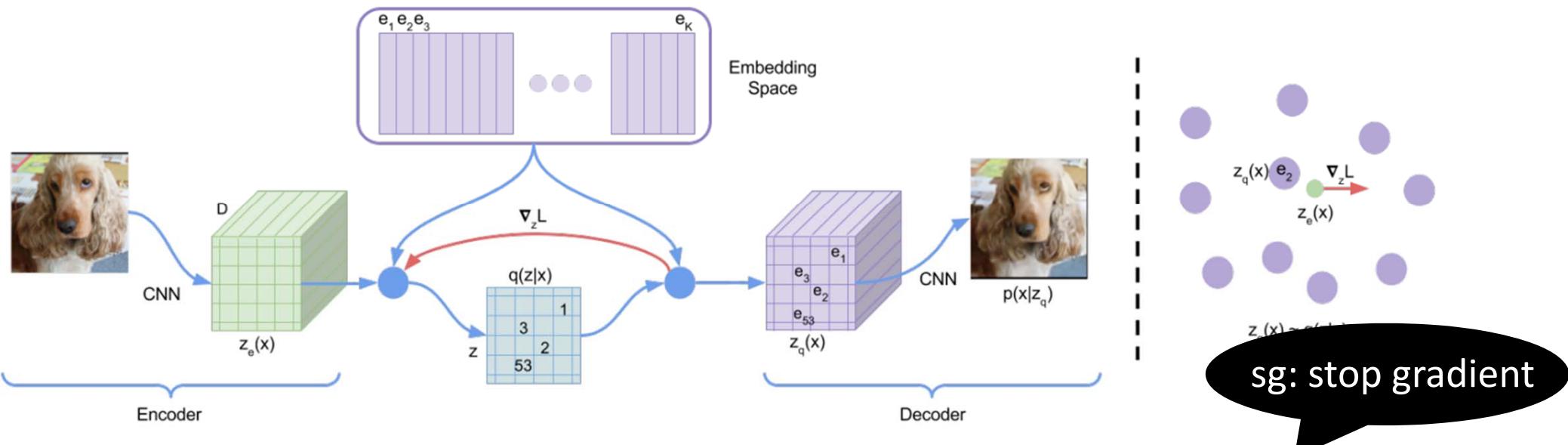
**Codebook loss**

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

**commitment loss**

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

# VQVAE



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

**Reconstruction loss**

**Codebook loss**

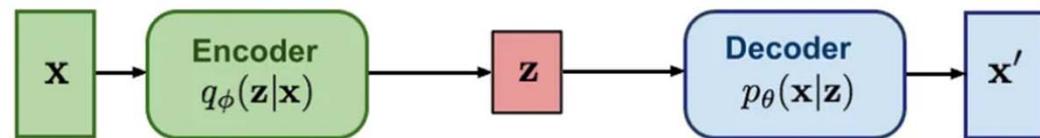
**commitment loss**

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

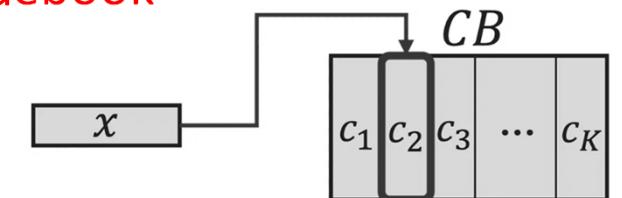
**VAE:** maximize variational lower bound



- **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

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

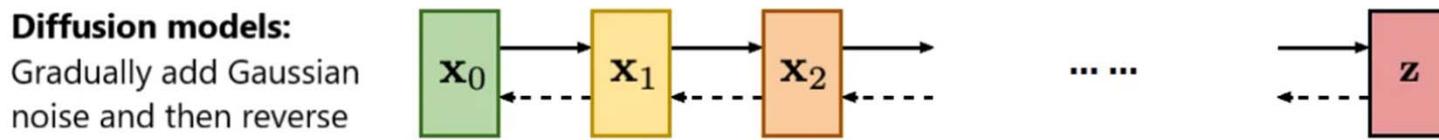
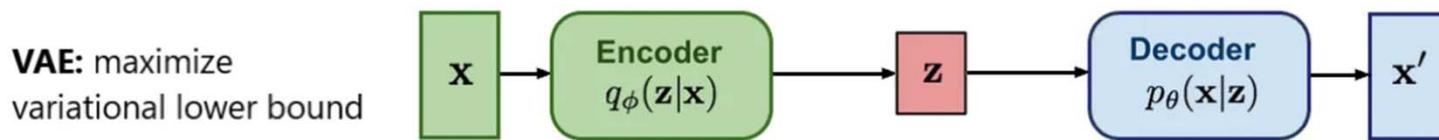
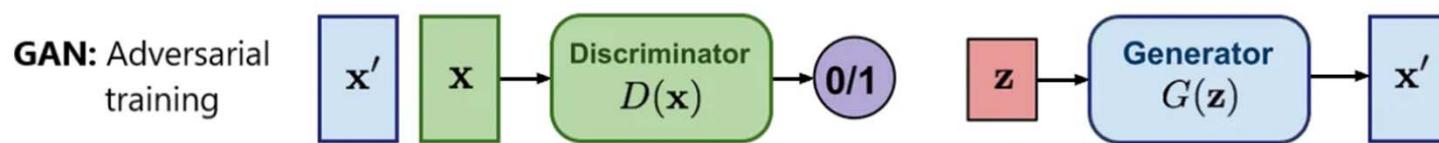
*in this case :  $x_{\text{quantized}} = c_2$*



(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>)

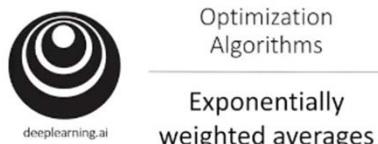
# VQVAE Training Tips: Exponential Moving Average (EMA)

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

Reconstruction loss

Codebook loss  
**(EMA update)**

commitment loss



Exponentially Weighted Averages (C2W2L03)

觀看 >



pedrocg42 commented on Oct 27, 2022

Author ...

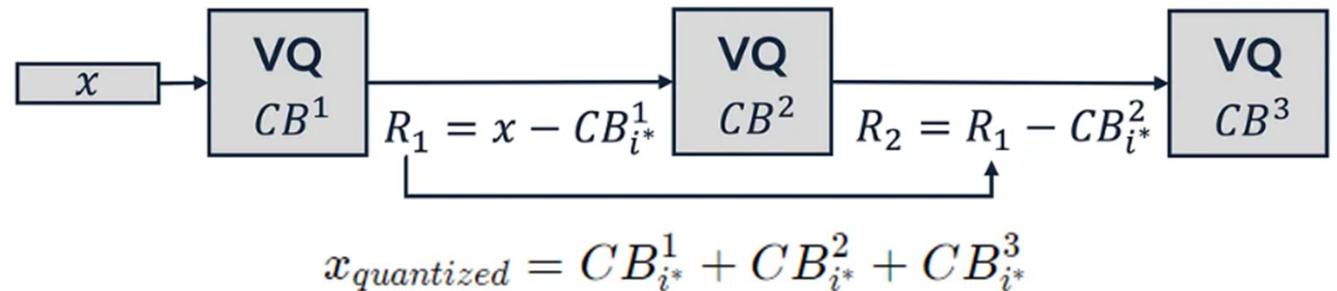
Right now I am following [@lucidrains](#) recommendation of not using any commitmen\_loss and just using EMA to update the VQ codebooks. At the time I tried several configurations of both decay and commitment\_cost with the same outcome,

Some people said that we don't need the commitment loss if we use EMA

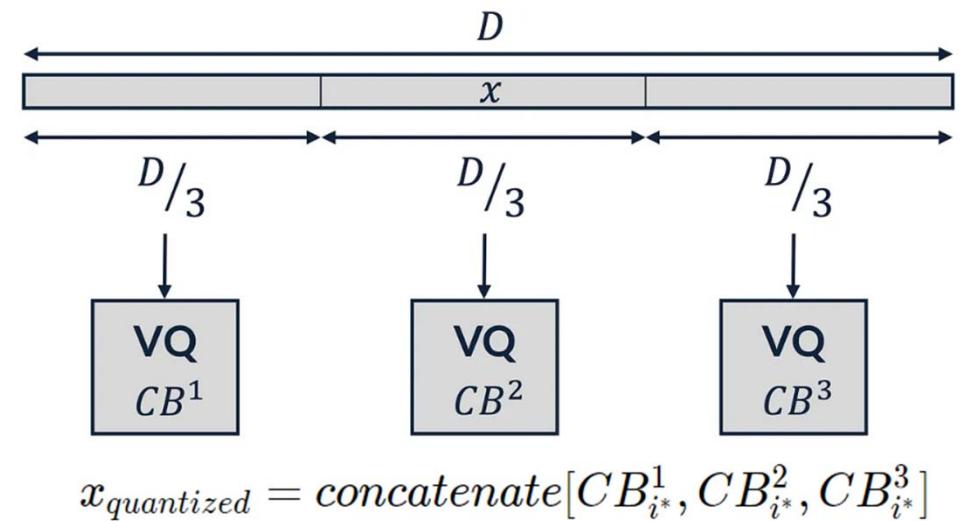
(<https://github.com/lucidrains/vector-quantize-pytorch/issues/27>)

# Variants of VQ

- Residual VQ (**RVQ**)



- Product VQ (**PVQ**)  
(a.k.a., decomposed VQ; DVQ)



(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

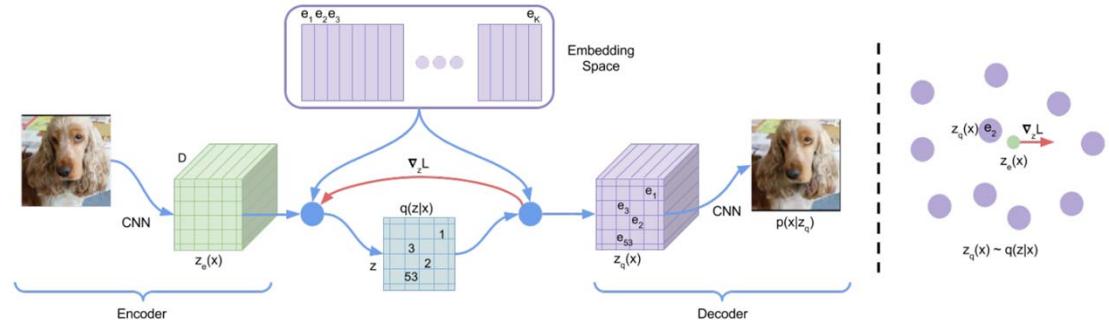
# VQVAE Training Tips: Increasing Codebook Usage

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



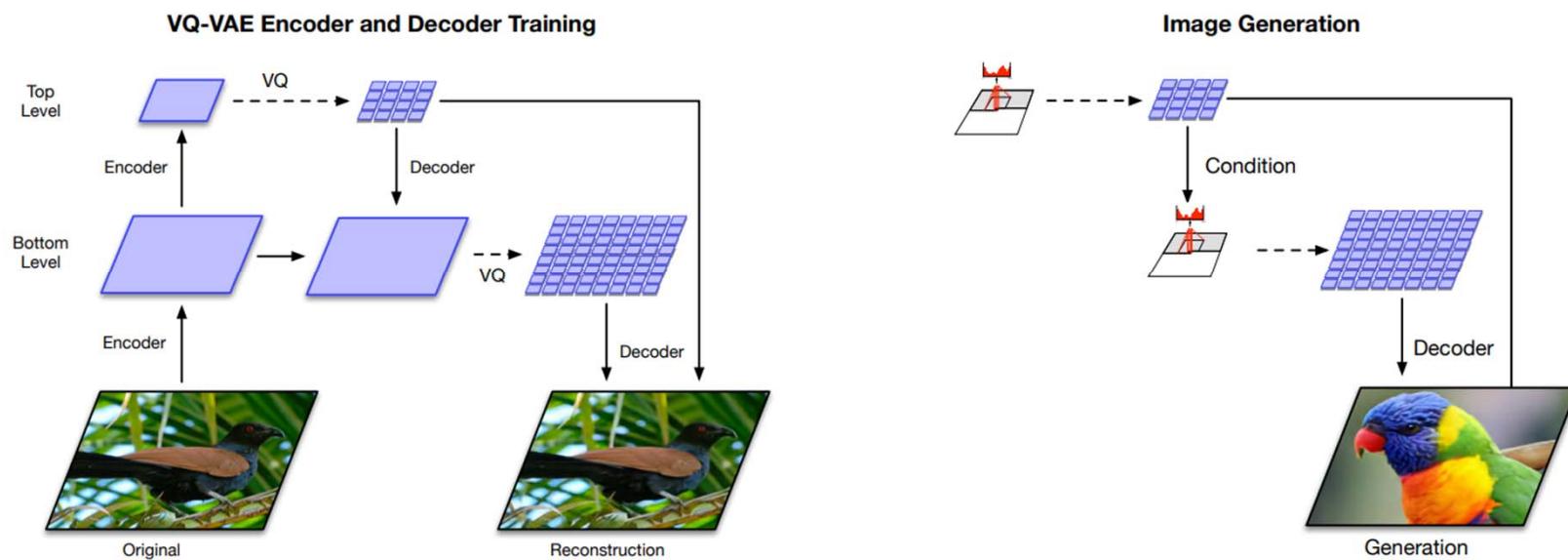
vector-quantize-pytorch

- 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

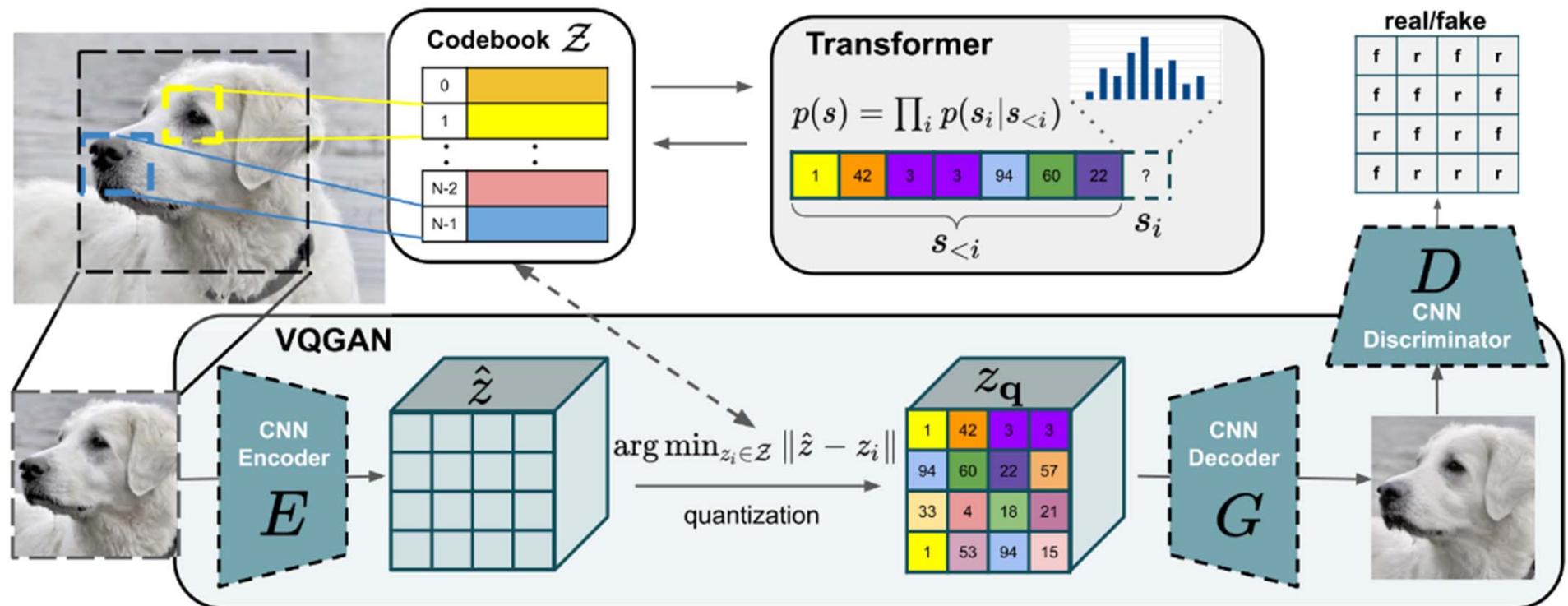
- The input to the model is a  $256 \times 256$  image that is compressed to quantized latent maps of size  $64 \times 64$  and  $32 \times 32$  for the bottom and top levels, respectively. The decoder reconstructs the image from the two latent maps.



# 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

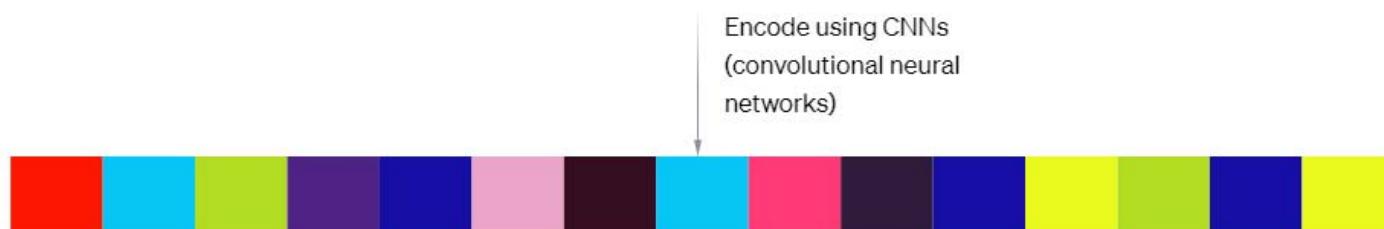
# VQVAE-based Music Generation: JukeBox

<https://openai.com/research/jukebox>

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



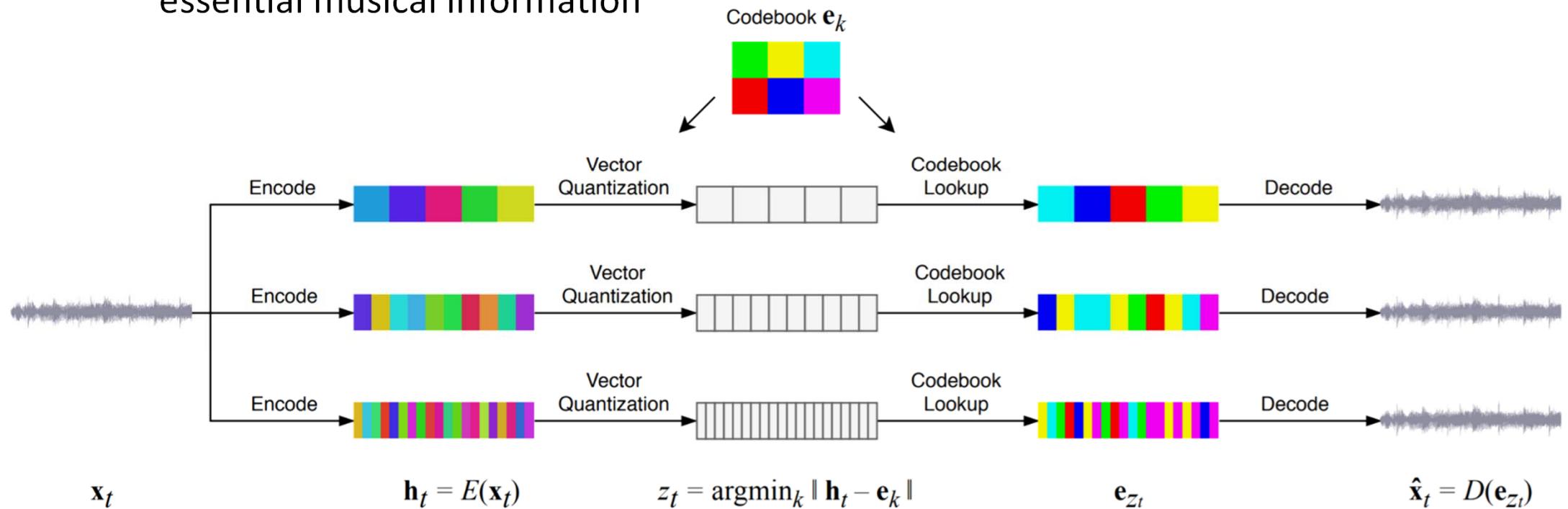
**Raw audio** 44.1k samples per second, where each sample is a float that represents the amplitude of sound at that moment in time



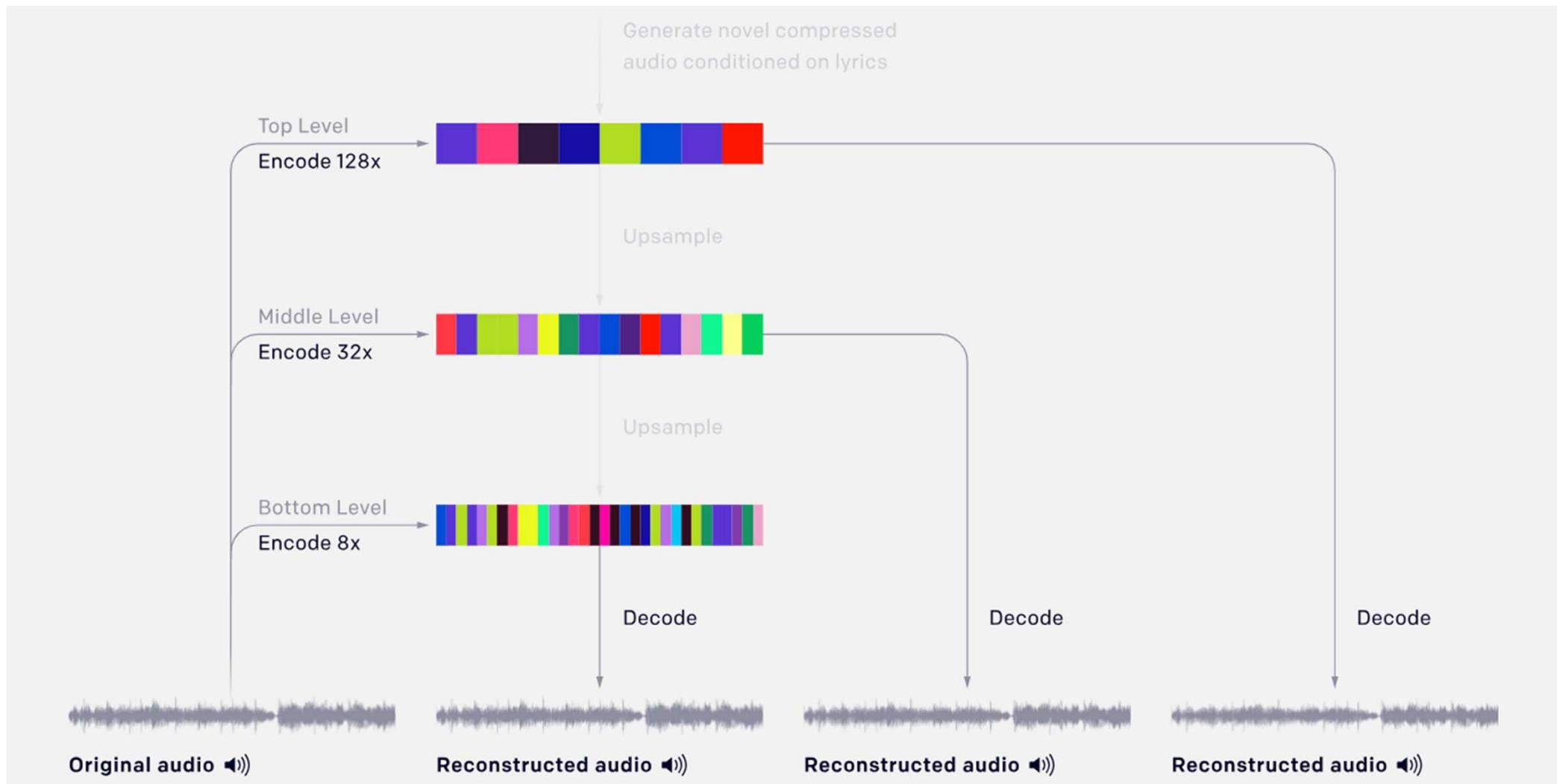
**Compressed audio** 344 samples per second, where each sample is 1 of 2048 possible vocab tokens

# 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

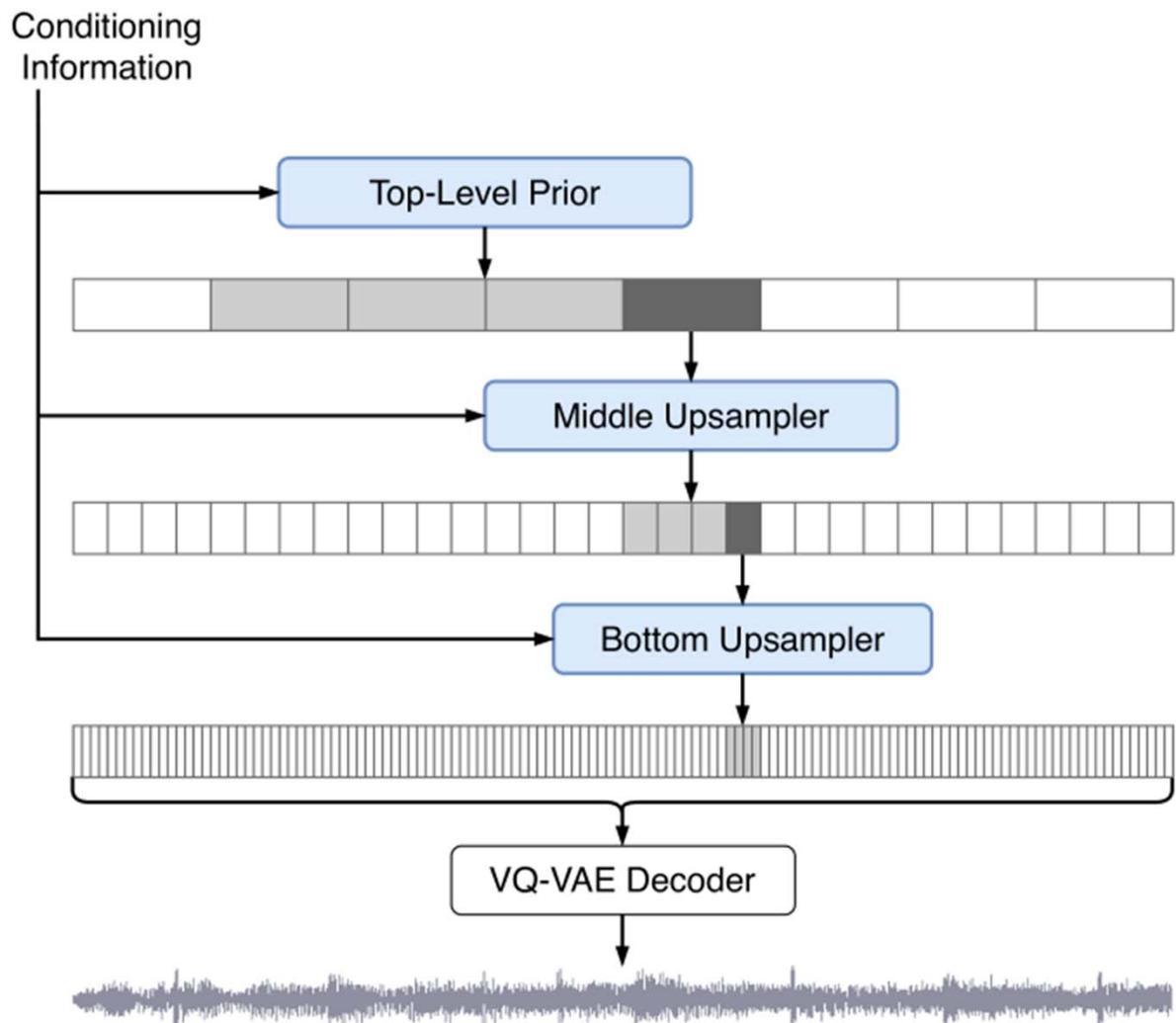


# VQVAE-based Music Generation: JukeBox

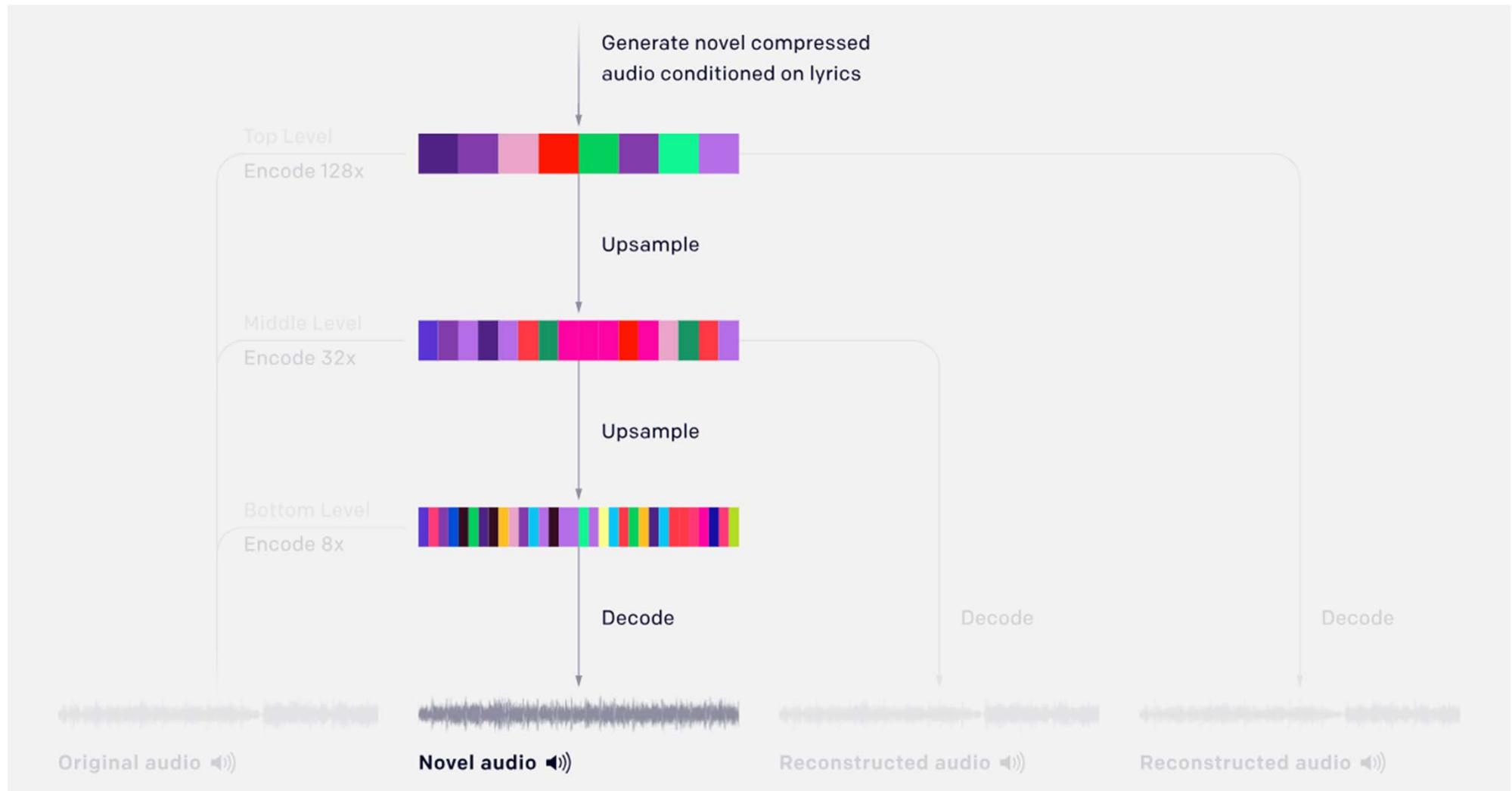


# VQVAE-based Music Generation: JukeBox

- 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>

Unseen lyrics   Re-renditions   Completions   Fun songs

Jukebox produces a wide range of music and singing styles, and generalizes to lyrics not seen during training. All the lyrics below have been co-written by a language model and OpenAI researchers.

▶ Country, in the style of Alan Jackson – Jukebox SOUNDCLOUD

Don't you know it's gonna be alright  
Let the darkness fade away  
And you, you gotta feel the same  
Let the fire burn  
Just as long as I am there  
I'll be there in your night  
I'll be there when the  
condition's right  
And I don't need to  
Call you up and say  
I've changed  
You should stay  
You should stay tonight  
Don't you know it's gonna be alright

# Issues of OpenAI's Jukebox

- **GPU-intensive**
  - **1M** proprietary 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.
- **Low controllability**
  - The only control signal is the lyrics

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

# Outline

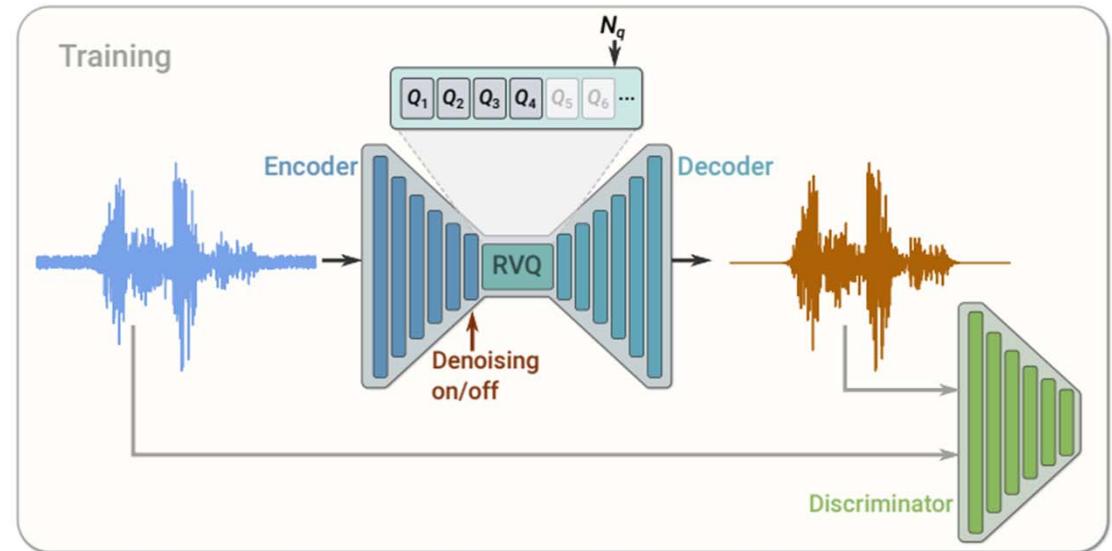
- Image/audio tokenization and generation via VQVAE
- **Audio codec models**
- Text-to-music
  - Linking text and musical audio
  - Transformer-based approach
  - Diffusion-based approach

# Audio Codec

- Use raw waveforms for VQ-VAE  
(like OpenAI's JukeBox)

- Initially developed for  
**audio compression**

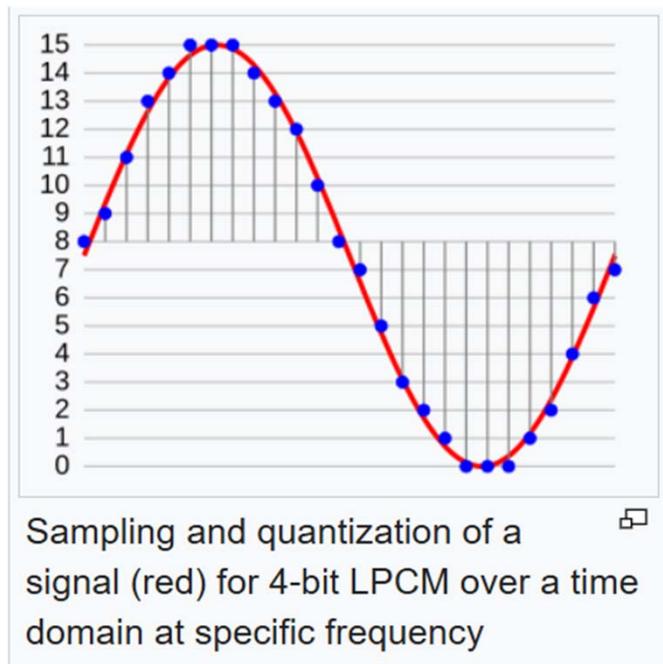
- **Bitrate vs fidelity**
  - Computational complexity vs  
coding efficiency



- Use vector quantization (VQ) for quantization
  - Produce **discrete tokens** that can be used for building a Transformer LM, though training LMs is not the focus of these audio compression papers

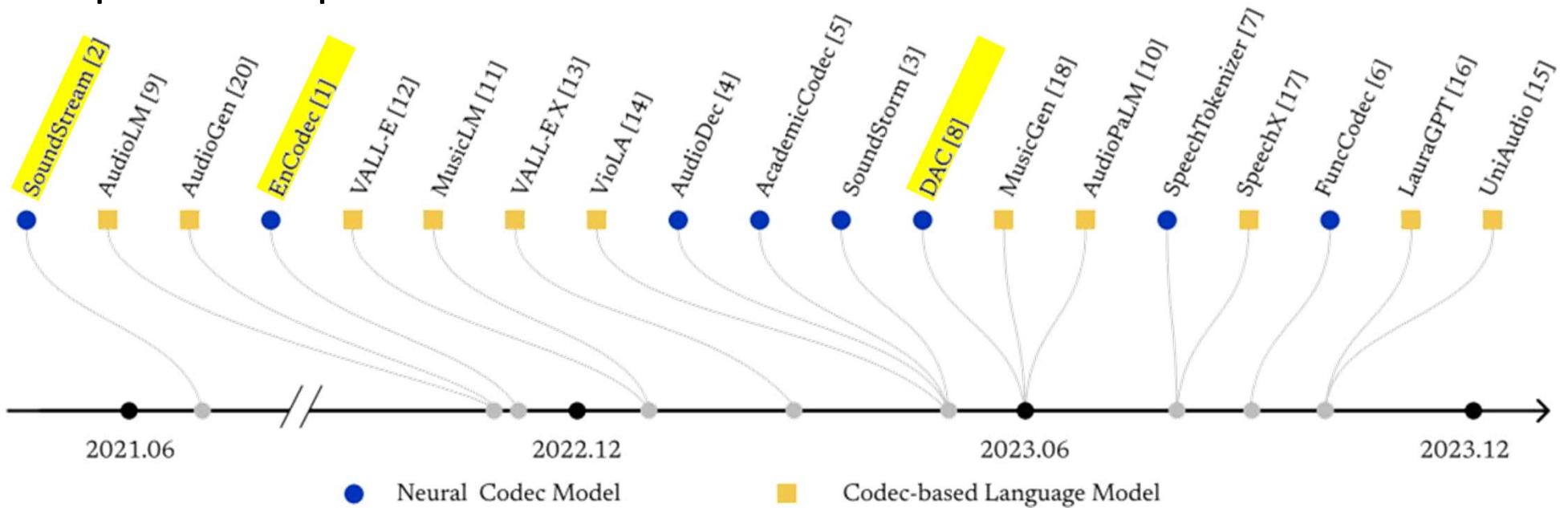
# Bitrate

- Pulse-code modulation (PCM)
  - 4bits: 16 possible values
  - 16bits: 65,536 possible values
- For 16bits 44.1kHz audio
  - 705.6 kbytes per second (**kbytes**)
  - 42.336 Mbytes per minute



# Audio Codec

- Rapid development



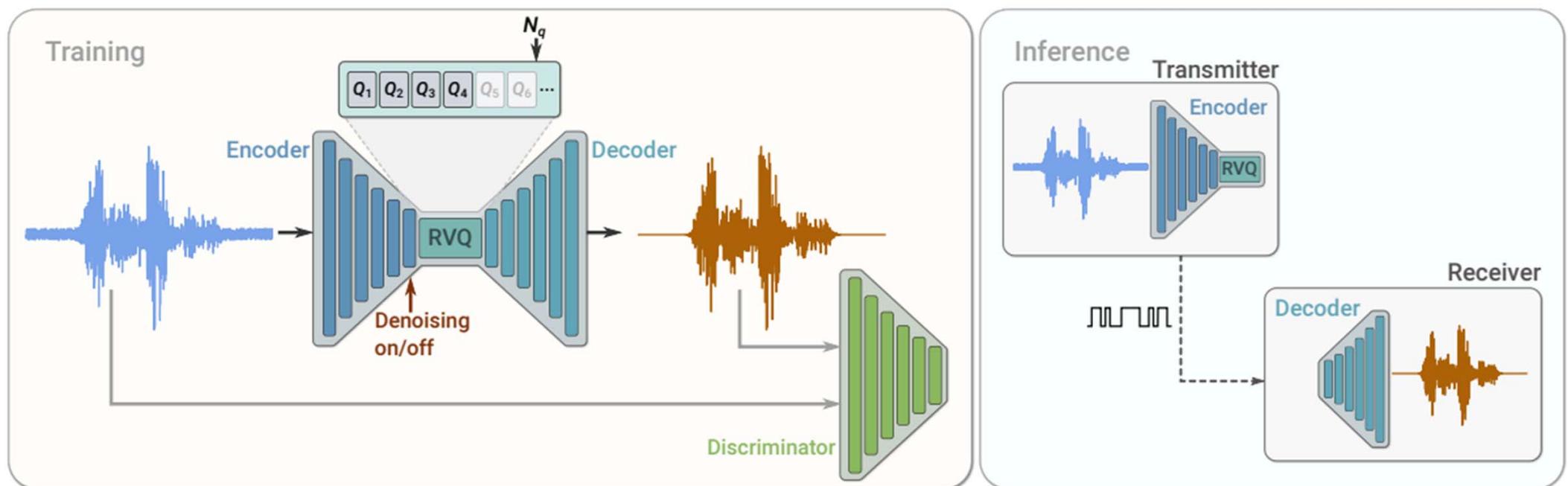
## Towards audio language modeling - an overview

Haibin Wu<sup>1</sup>, Xuanjun Chen<sup>1\*</sup>, Yi-Cheng Lin<sup>1\*</sup>, Kai-wei Chang<sup>1</sup>, Ho-Lam Chung<sup>1</sup>,  
Alexander H. Liu<sup>2</sup>, Hung-yi Lee<sup>1</sup>

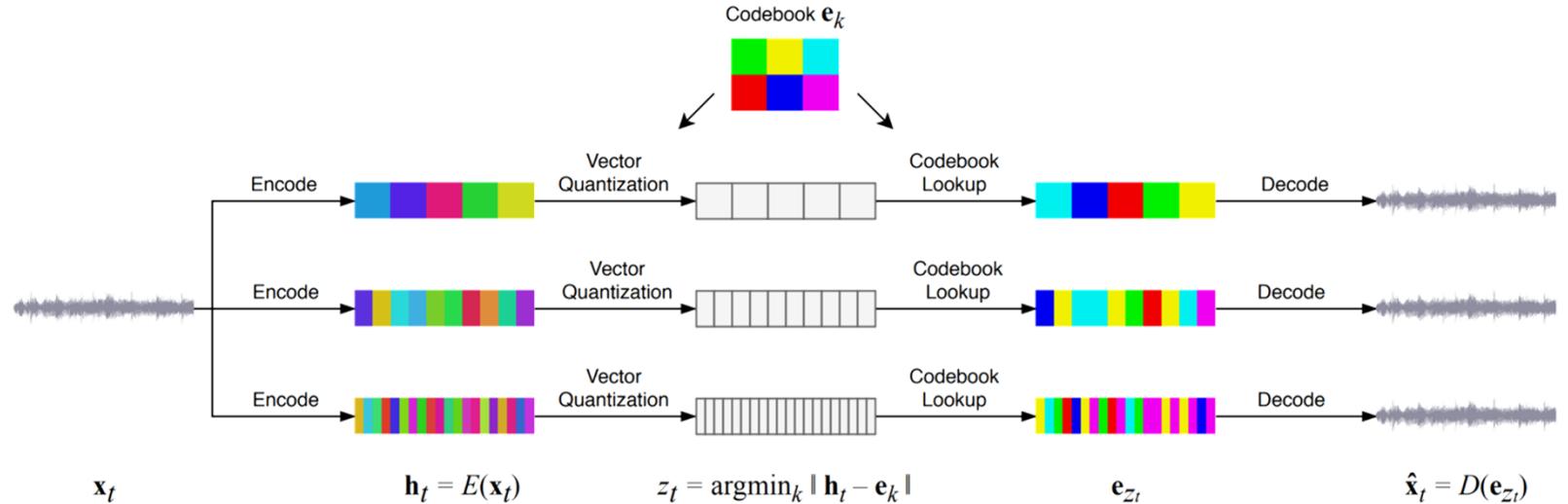
<https://arxiv.org/pdf/2402.13236>

# SoundStream (from Google)

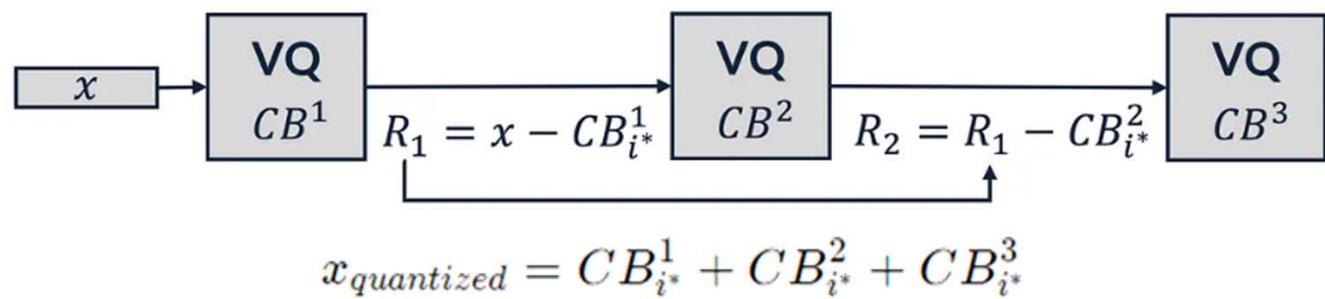
- Use RVQ
  - The same audio chunk is **recursively** VQed by different codebooks



# Hierarchical VQ vs RVQ



- **RVQ** is found to be superior in audio codec research



# RVQ Demonstration

- Use descript audio codec (DAC) → see later slides
- Adding more quantizers (codebook layers) improve fidelity
  - At the cost of increased bitrate (therefore less compression rate)



9 layers



6 layers



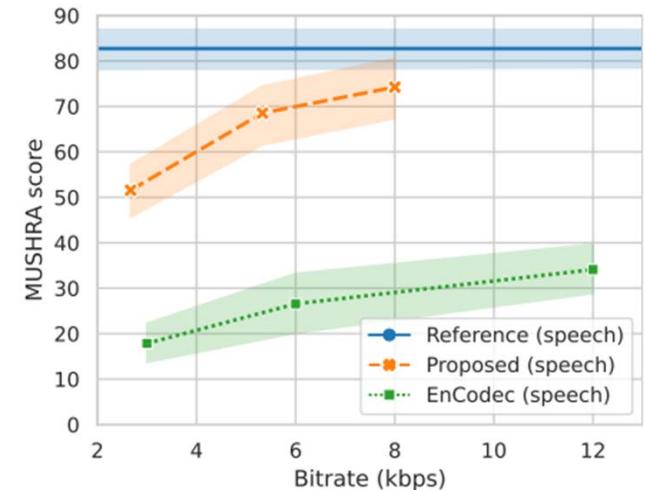
3 layers



2 layers

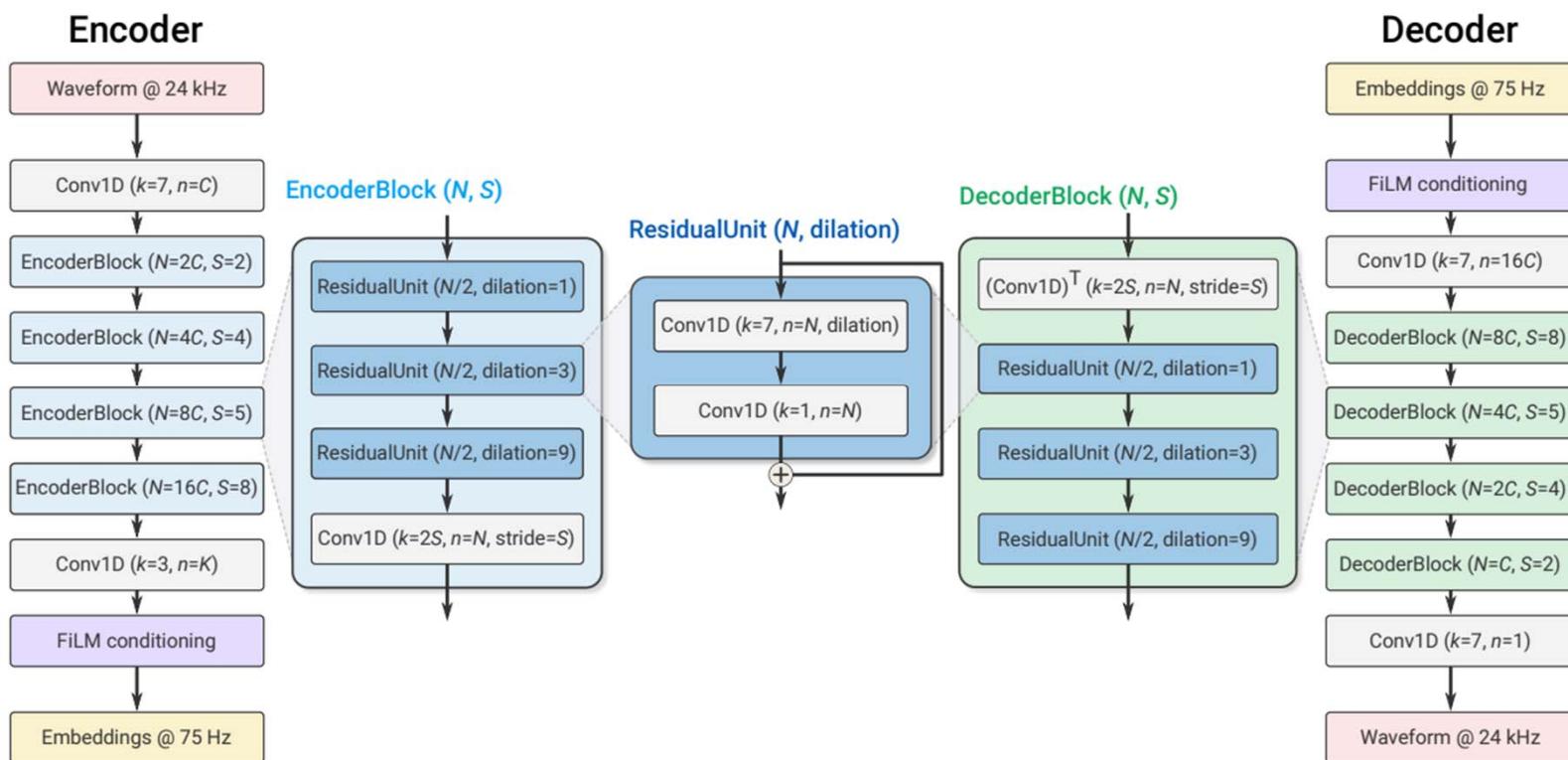


1 layer



# SoundStream (from Google)

- Convolutional encoder/decoder



# SoundStream (from Google)

- 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**

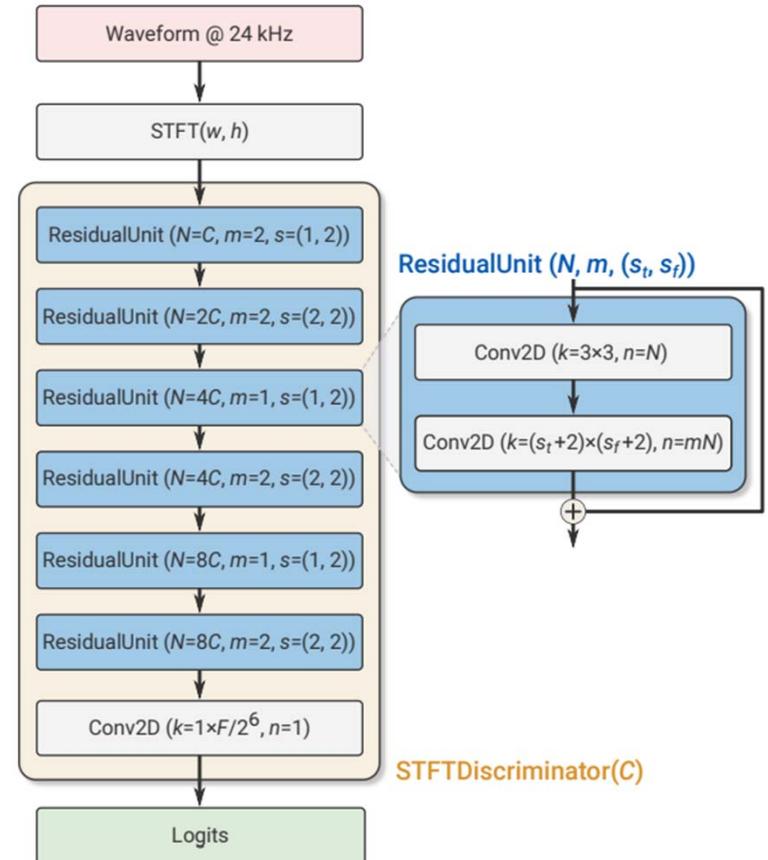


Fig. 4: STFT-based discriminator architecture.

## SoundStream (from Google)

- Use **8** layers of RVQ
- Codebook size set to **1024** ( $2^{10}$ ) for each quantizer
- Evaluation metric: Google's **ViSQOL** (Virtual Speech Quality Objective Listener)
- **Not open source**

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                     | $4.01 \pm 0.03$ | $3.98 \pm 0.03$ | $3.92 \pm 0.03$ |

(the effective number of unique codewords of these three configurations is all  $2^{80}$ , why?)

# EnCodec (from Meta)

## Open source

<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]
```



# EnCodec (from Meta)

## Extracting discrete representations

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

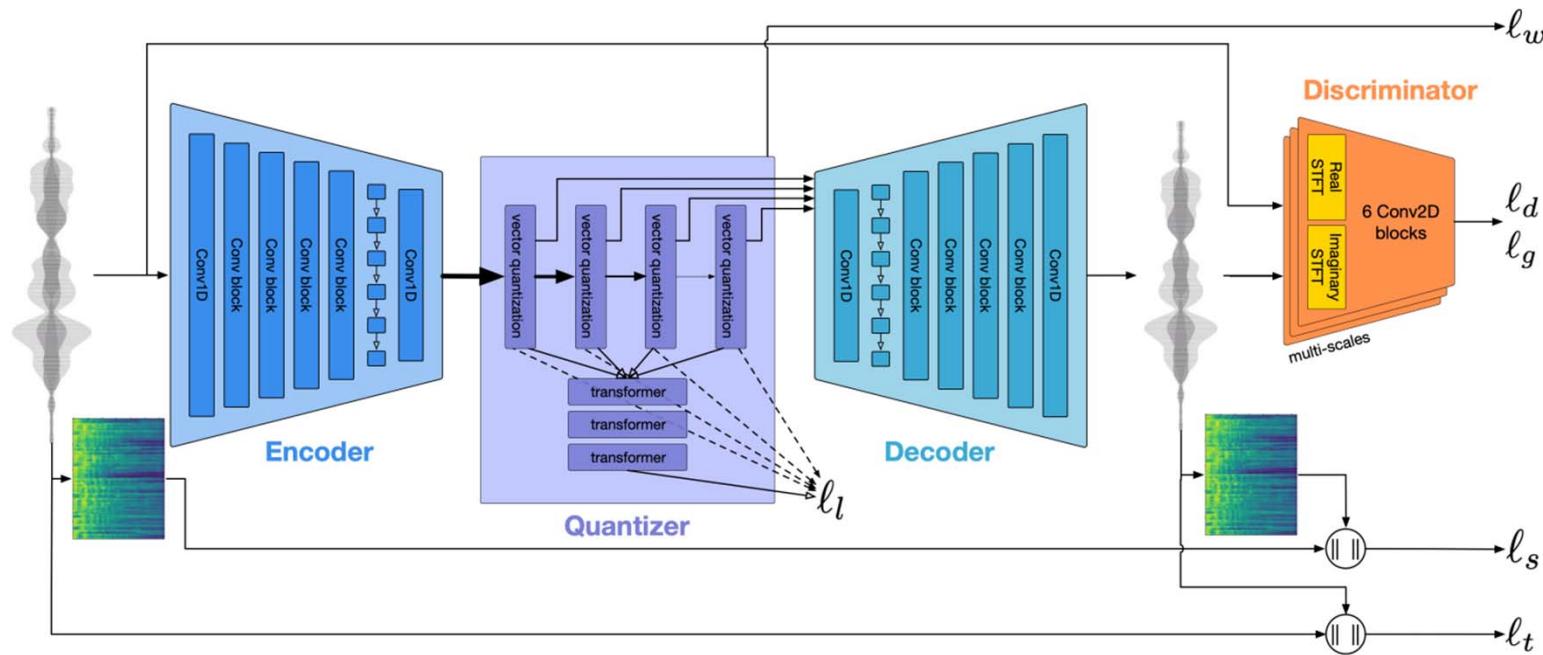
# Instantiate a pretrained EnCodec model
model = EncodecModel.encodec_model_24khz()
# The number of codebooks used will be determined by the bandwidth selected.
# E.g. for a bandwidth of 6 kbps, `n_q = 8` codebooks are used.
# Supported bandwidths are 1.5 kbps (n_q = 2), 3 kbps (n_q = 4), 6 kbps (n_q = 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]
```

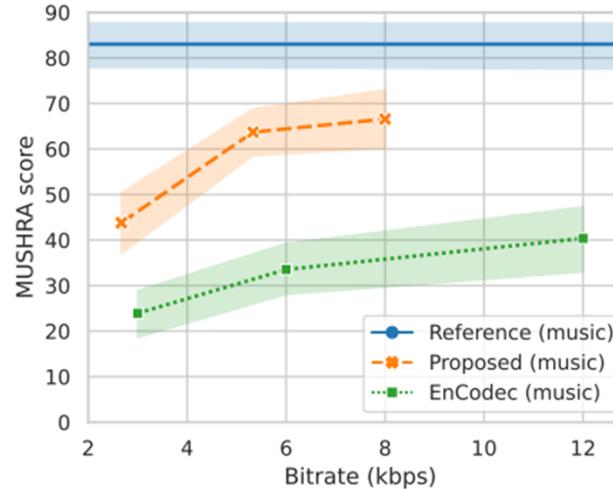
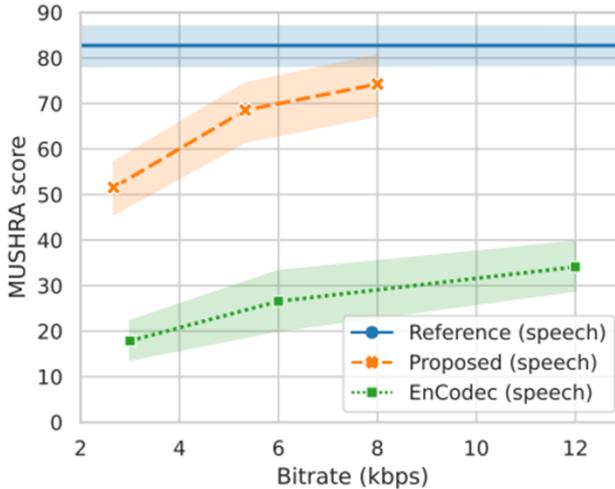
# Encodec

- RVQVAE+GAN, convolutional encoder/decoder
- Trained using **8 A100 GPUs** for 300 epochs
- Use the Jamendo dataset for the music version of Encodec



# Descript Audio Codec (DAC)

- Can compress 44.1 KHz audio into discrete codes at **8 kbps** bitrate (~90x compression)
  - Why 90x? (16bits 44.1kHz audio: 705.6 kbps)
- Outperform EnCodec in both objective and subjective evaluations



| Codec    | Bitrate (kbps) |       | Bandwidth (kHz) |      | Mel distance ↓ |        | STFT distance ↓ |  | ViSQOL ↑ |  | SI-SDR ↑ |  |
|----------|----------------|-------|-----------------|------|----------------|--------|-----------------|--|----------|--|----------|--|
| Proposed | 1.78           | 22.05 | 1.39            | 1.95 | 3.76           | 2.16   |                 |  |          |  |          |  |
|          | 2.67           | 22.05 | 1.28            | 1.85 | 3.90           | 4.41   |                 |  |          |  |          |  |
|          | 5.33           | 22.05 | 1.07            | 1.69 | 4.09           | 8.13   |                 |  |          |  |          |  |
|          | 8              | 22.05 | 0.93            | 1.60 | 4.18           | 10.75  |                 |  |          |  |          |  |
| EnCodec  | 1.5            | 12    | 2.11            | 4.30 | 2.82           | -0.02  |                 |  |          |  |          |  |
|          | 3              | 12    | 1.97            | 4.19 | 2.94           | 2.94   |                 |  |          |  |          |  |
|          | 6              | 12    | 1.83            | 4.10 | 3.05           | 5.99   |                 |  |          |  |          |  |
|          | 12             | 12    | 1.70            | 4.02 | 3.13           | 8.36   |                 |  |          |  |          |  |
|          | 24             | 12    | 1.61            | 3.97 | 3.16           | 9.59   |                 |  |          |  |          |  |
| Lyra     | 9.2            | 8     | 2.71            | 4.86 | 2.19           | -14.52 |                 |  |          |  |          |  |
| Opus     |                | 8     | 3.60            | 5.72 | 2.06           | 5.68   |                 |  |          |  |          |  |
|          |                | 14    | 1.23            | 2.14 | 4.02           | 8.02   |                 |  |          |  |          |  |
|          |                | 24    | 0.88            | 1.90 | 4.15           | 11.65  |                 |  |          |  |          |  |

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

# Descript Audio Codec (DAC)

Improved  
RVQGAN

| 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%                  |
|                     | 1536         | snake      | ✗            | ✗            | ✗               | ✓                | 8          | Proj.         | 1.0            | 8              | ✓              | 1.13           | 1.92            | 4.12     | 1.07     | 62%                  |
| Discriminator       | 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     | 44 99%               |

# Descript Audio Codec (DAC)

<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)
```

# DAC vs Encoded

| Codec       | Sampling rate (kHz) | Target bitrate (kbps) | Striding factor | Frame rate (Hz) | # of 10-bit codebooks | Compression factor |
|-------------|---------------------|-----------------------|-----------------|-----------------|-----------------------|--------------------|
| Proposed    | 44.1                | 8                     | 512             | 86              | 9                     | 91.16              |
| EnCodec     | 24                  | 24                    | 320             | 75              | 32                    | 16                 |
|             | 48                  | 24                    | 320             | 150             | 16                    | 32                 |
| SoundStream | 24                  | 6                     | 320             | 75              | 8                     | 64                 |

Q: why 8 kbps?

A:  $9 * 10 * 86$

Sampling rate (kHz)

Target bitrate (kbps)

Striding factor

Frame rate (Hz)

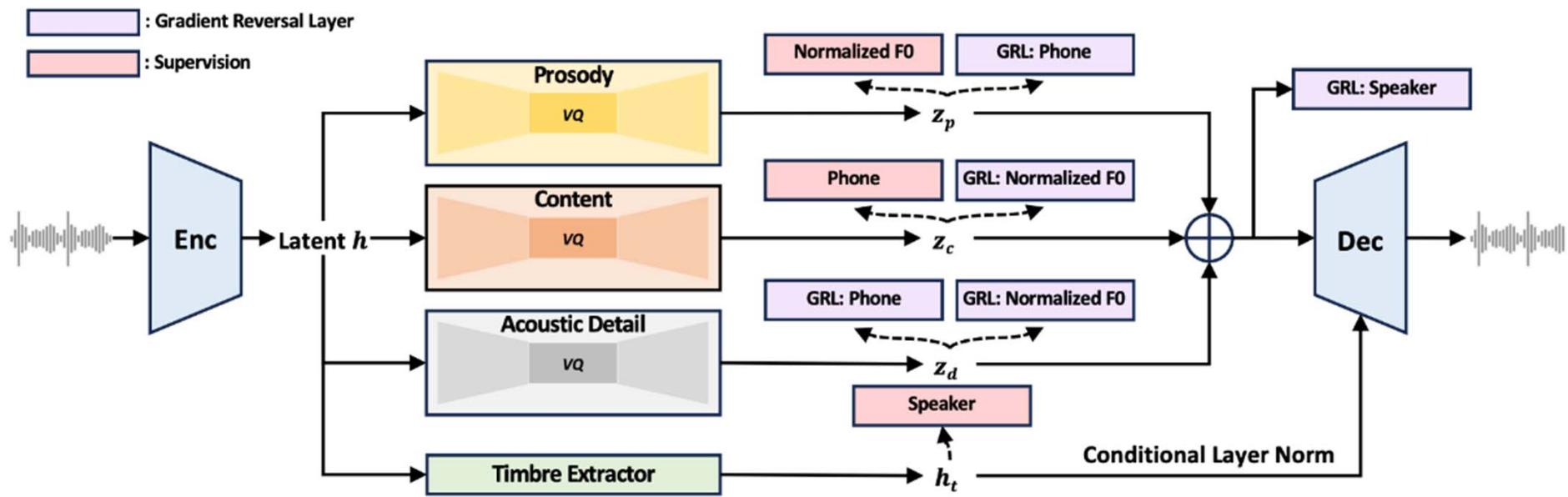
# of 10-bit codebooks

Compression factor

(Frame rate is related to the design of the encoder/decoder in VQVAE)

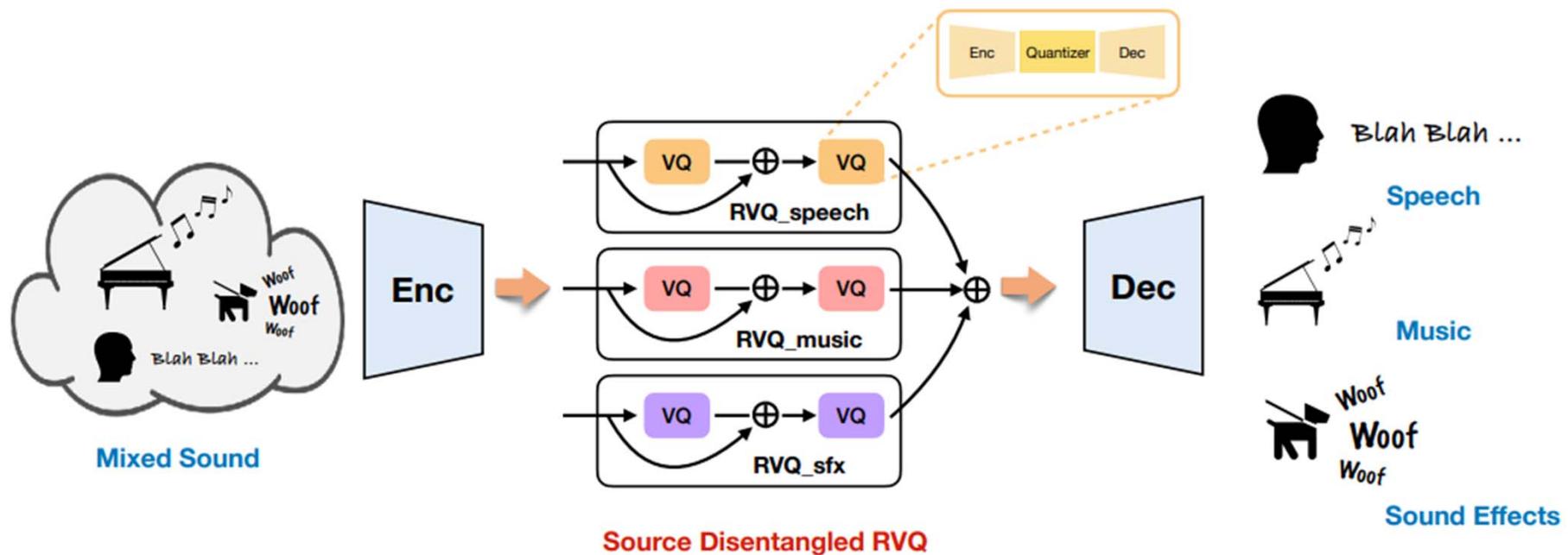
# Many More Codec Models...

- **FACodec:** factorized neural speech codec (arxiv:2403.03100)



# Many More Codec Models...

- **SD-Codec:** source-disentangled neural audio codec (arxiv:2409.11228)



Ref: Bie et al, "Learning source disentanglement in neural audio codec," arXiv 2024

# 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 music
- **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**” 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<br>Inflexible to expand the functions.   | ✗    |
| WavJourney [127]   | AudioCaps   | TTM        | The process of remixing and deteriorating may push the synthetic audio away from the real.<br>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        | —<br>—   | ✗    |
| MeLoDy [232]       | 6.4 Million Samples based on MusicCaps  | TTM<br>MTM | Training data mostly contain non-vocal music only<br>Training on LM and DPD on 10-second audio chunks can affect the long generation   | ✓    |
| MuseCoco [233]     | Million MIDI Dataset<br>EMPOIA<br>MetaMidi<br>POP909<br>Symphony<br>Emotion-gen | TSM        | Model primarily focuses on producing symbolic music based on textual descriptions, with little consideration on long sequence modelling.<br>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

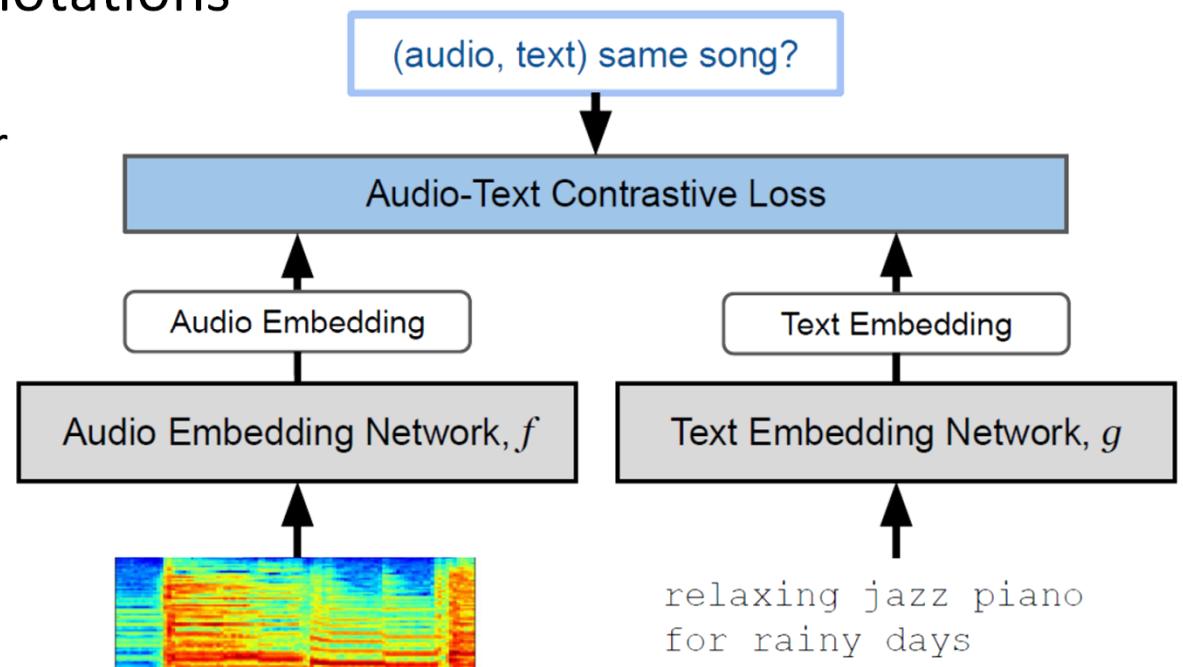
- **1. Audio encoder/decoder**
  - So that we can build an LM over the (discrete or continuous) decoder input
- **2. Music LM**
  - Can do unconditional generation
- **3. Text encoder or text-music joint embedding space**
  - Provides conditions for the music LM to realize text-to-music generation

# Key Technologies that Enable Text-to-Music Generation

- **Language model (LM) for music:** RNN, Transformers
  - Model long term dependency
  - Initially applied to MIDI music generation as it's easier to convert MIDI events into tokens
  - **Significance:** A few bars of MIDI music → full-length MIDI music
- **Audio waveform encoder/decoder**
  - Thanks to advances in Mel-vocoders, source separation, audio codec models, etc
  - Tokenize audio files (spectrograms or waveforms)
    - Enable us to build LM for musical audio
    - And learn the dependency between text tokens and audio tokens
  - **Significance:** LM for MIDI music → LM for musical audio

# 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...**



# MuLan: Text-Music Joint Embedding Learning

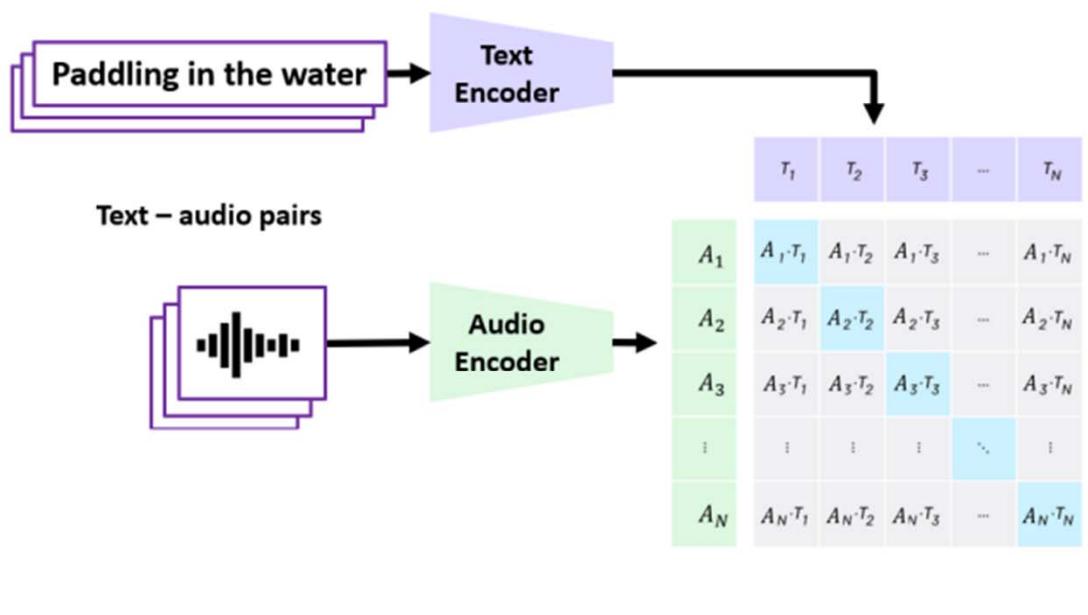
- MuLan takes the form of a two-tower, joint audio-text embedding model trained using **44 million music recordings** (370K hours) and weakly-associated, free-form text annotations
  - Data cleansing
    - Fine-tune a pre-trained BERT with a binary classification task on a small curated set of 700 sentences, 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.

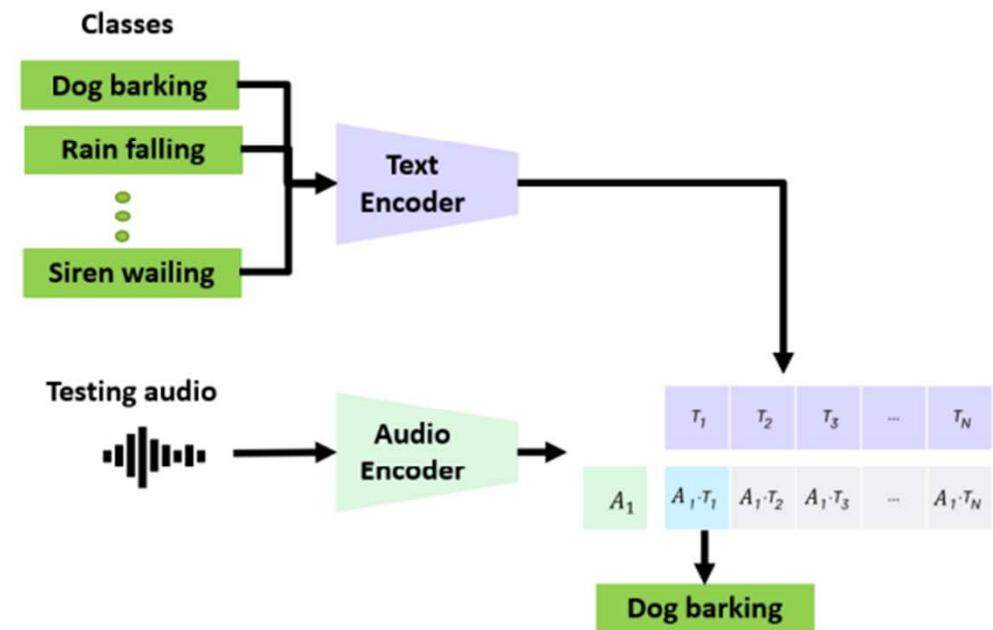
| Type            | Examples  |
|-----------------|---|
| Short-form (SF) | tags like genre, mood, instrument, artist name, song title, album name                                |
| Long-form (LF)  | ‘Hip-hop features rap with an electronic backing.’<br>‘The melody is so nostalgic and unforgettable.’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

# Microsoft's CLAP: Text-Audio Joint Embedding Learning

## 1. Contrastive Pretraining

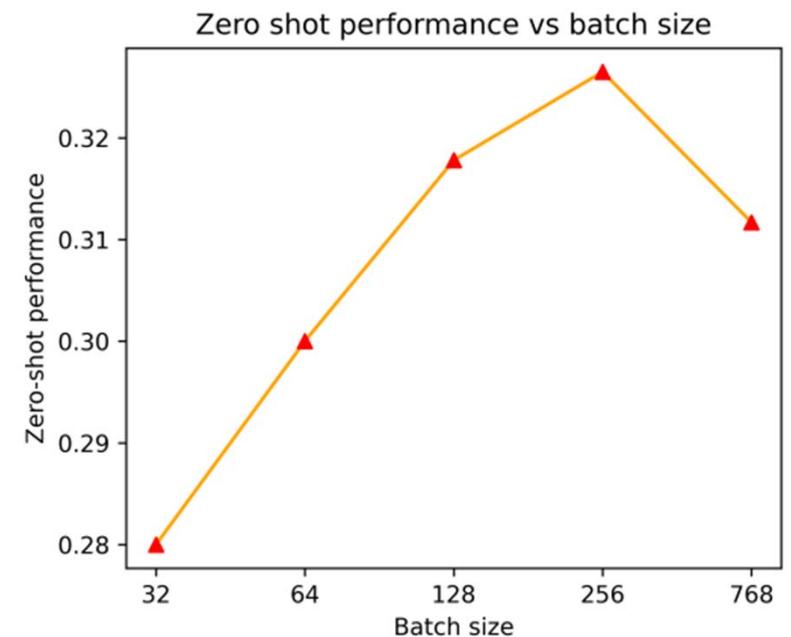


## 2. Use pretrained encoders for zero-shot prediction in a new dataset or task



# Microsoft's CLAP: Text-Audio Joint Embedding Learning

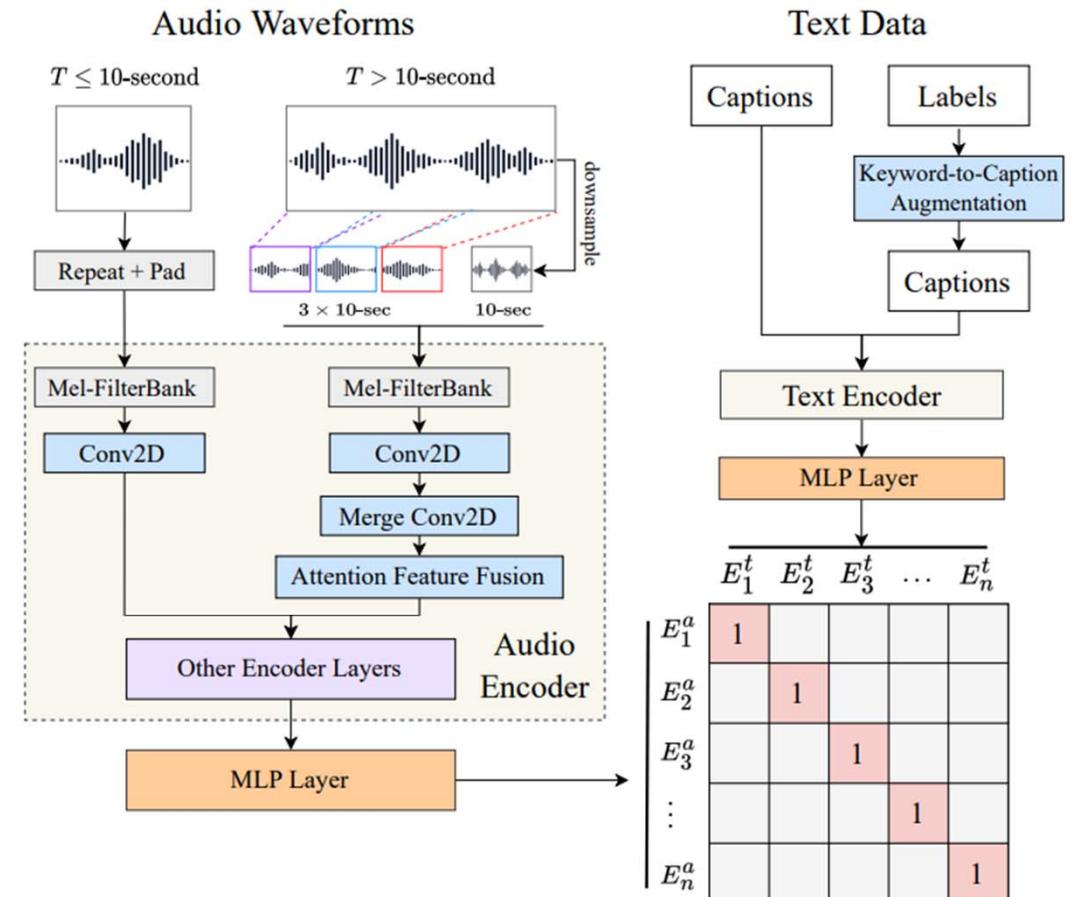
- 128,010 audio/text pairs from 4 datasets (FSD50k, ClothoV2, AudioCaps, MACS)
  - Each audio clip is randomly truncated to 5 secs
- Audio encoder: **PANNs-CNN14**
  - 80.8 million parameters & an embedding size of 2048; pretrained on AudioSet
- Text encoder: **BERT**
  - 110 million parameters
- Use 8-24 V100 GPUs (16GB VRAM)
- **Batch size is important (as large as 256)**



# 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     | <b>45.7</b>        | <b>51.3</b> |



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

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

- More than 633,526 audio/text pairs
- Use 10-second audio segments
- Tokenize the text with a maximum token length of 77
- Very large **batch size**:
  - Train the model using a batch size of 768 on AudioCaps+Clotho dataset, 2304 on training dataset containing LAION-Audio-630K, and 4608 on training dataset containing AudioSet

# 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](#)

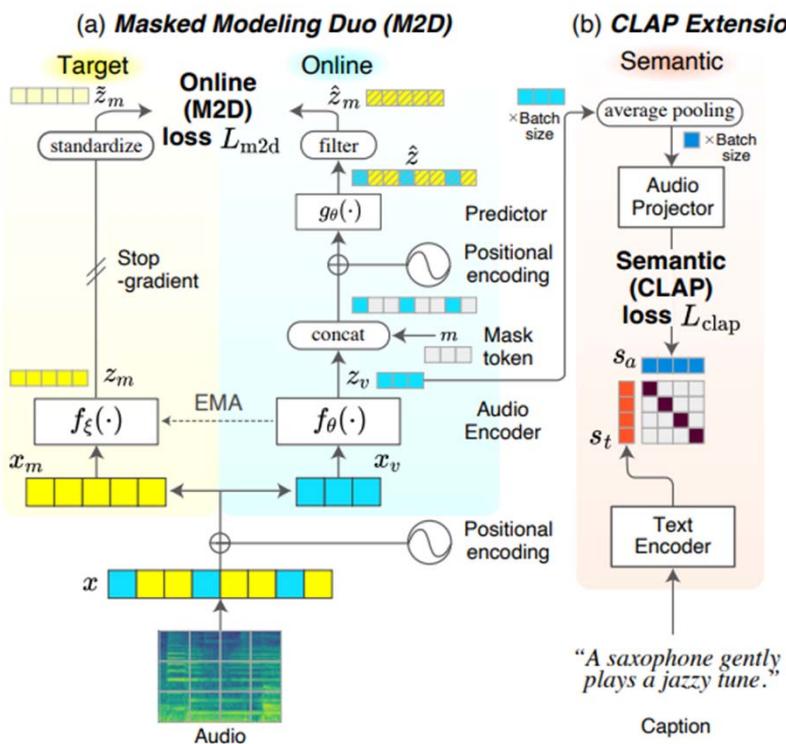
*clapcap* is the audio captioning model that uses the 2023 encoders.

<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](#)

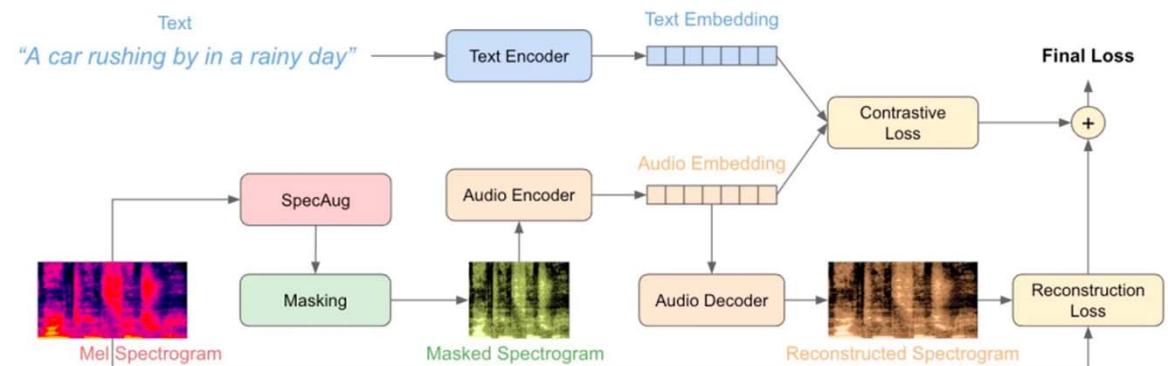
# Variants of CLAP

- M2D-CLAP



Ref: Niizumi et al, "M2D-CLAP: Masked modeling duo meets clap for learning general-purpose audio-language representation," INTERSPEECH 2024

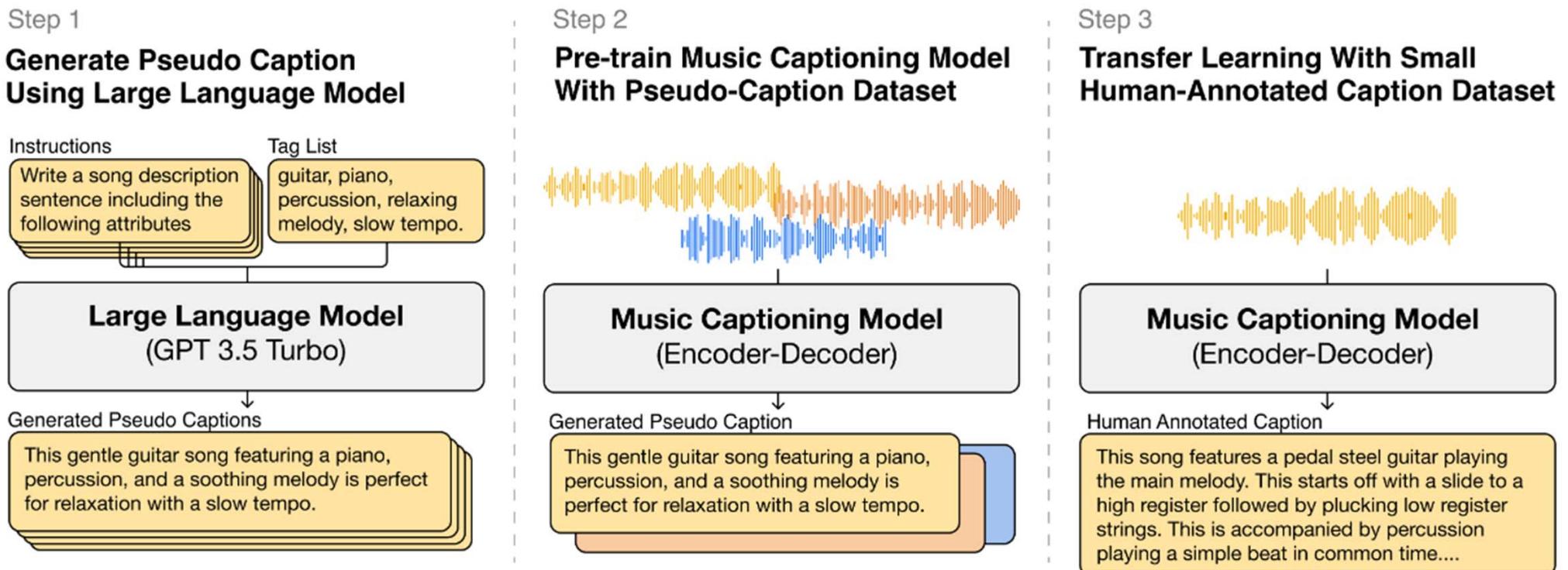
- FLAP



Ref: Yeh et al, "FLAP: Fast language-audio pre-training," ASRU 2023

# LP-MusicCaps: LLM-Based Pseudo Music Captioning

<https://github.com/seunghheondoh/lp-music-caps>



# 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        |
| <b>LP-MusicCaps-MC</b>  | 6k     | 15.3         | 4   | 44.9±21.3  |
| <b>LP-MusicCaps-MTT</b> | 22k    | 180.3        | 4   | 24.8±13.6  |
| <b>LP-MusicCaps-MSD</b> | 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.

# 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       | <a href="https://sound-effects.bbcrewind.co.uk/">https://sound-effects.bbcrewind.co.uk/</a> |
| Audiostock           | 43    | Caption       | <a href="https://audiostock.net/se">https://audiostock.net/se</a>                           |
| Filter AudioSet      | 2084  | Label         | Gemmeke et al.  |
| ESC-50               | 3     | Label         | Piczak  |
| FSD50K               | 108   | Label         | <a href="https://annotator.freesound.org/fsd/">https://annotator.freesound.org/fsd/</a>     |
| Sonniss Game Effects | 20    | Label         | <a href="https://sonniss.com/gameaudiogdc/">https://sonniss.com/gameaudiogdc/</a>           |
| WeSoundEffects       | 11    | Label         | <a href="https://wesoundeffects.com/">https://wesoundeffects.com/</a>                       |
| Epidemic Sound       | 220   | Label         | <a href="https://www.epidemicsound.com/">https://www.epidemicsound.com/</a>                 |
| 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

# Multimodal Audio-Language Models: LLM for Musical Audio

Question: How does the structure of the melody change throughout the song?

(A) It's a 90s dance pop song  
(B) The base melody stays the same, and other melodies are layered on top  
(C) The song uses acoustic instruments  
(D) The entire melody changes every verse

(B) The base melody stays the same, and other melodies are layered on top ✓

| Model           | Audio encoder                                 | LLM             |
|-----------------|---|-----------------|
| MusiLingo [9]   | MERT [40]                                     | Vicuna 7B [41]  |
| MuLLaMa [8]     | MERT [40]                                     | LLaMA-2 7B [42] |
| M2UGen [12]     | MERT [40]                                     | LLaMA-2 7B [42] |
| SALMONN [11]    | BEATS [43] & Whisper <sub>large-v2</sub> [44] | Vicuna 7B [41]  |
| Qwen-Audio [13] | Whisper <sub>large-v2</sub> [44]              | Qwen 7B [45]    |

**Table 2. Overview of models** we evaluate in our study.

# LLM for Symbolic Music

## Music Theory Exercise

**Default:** "You will see JSON-formatted instruction data followed by questions. Your responses should only indicate the selected option (using uppercase letters), without providing any analysis."

**CoT:** "You will see a JSON-formatted instruction data followed by questions. Your responses should include an analysis step by step. The returned JSON format is as follows: {"reason": "**Let's think step by step**", "answer": "A"} "

**ICL: CoT +** "Here is an example of a question and its answer:  
Read the following questions from the four options (A, B, C and D) given in each question. Choose the best option. Which of the following is the name of the note in the example?",

"L:1/4 M:4/4 K:Cb, D,4 |]",  
Options: {"A": "B-flat", "B": "D", "C": "B", "D": "D-flat"},  
Answer: "D".

## Tasks

Music theory exercise

Motif extraction (ME)

Musical form extraction (MFE)

Chord-conditioned generation (CCG)

Melody harmonization (MH)

Musical-form-and-motif-conditioned generation

Ref: Zhou et al, "Can LLMs "reason" in music? An evaluation of LLMs' capability of music understanding and generation," ISMIR 2024

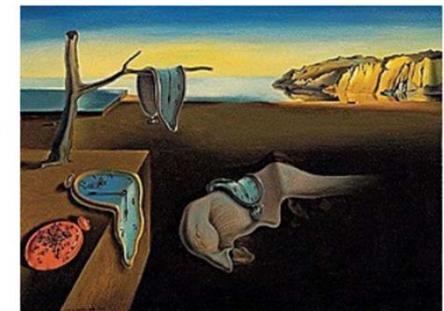
# Outline

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

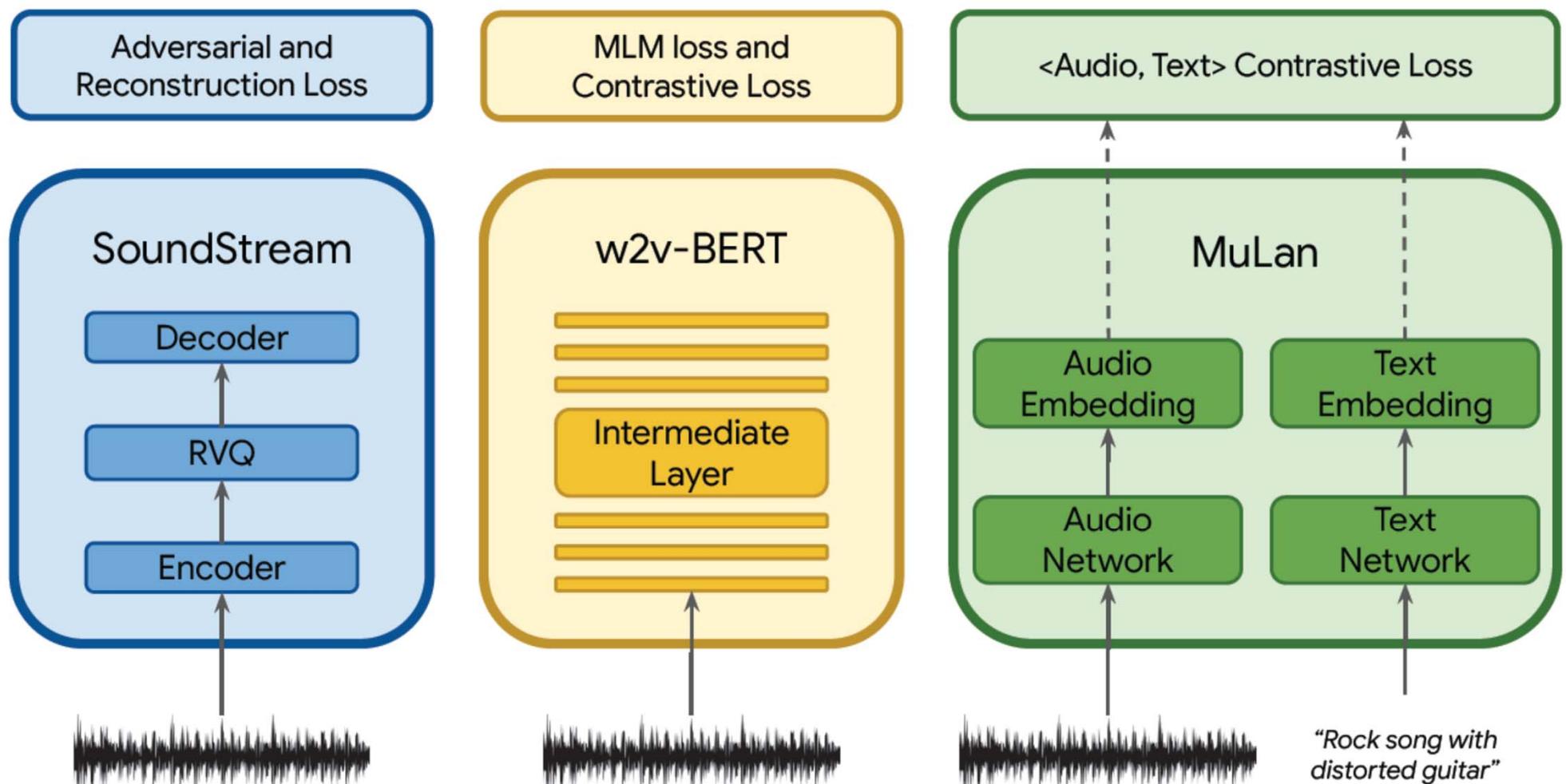
# MusicLM (from Google)

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

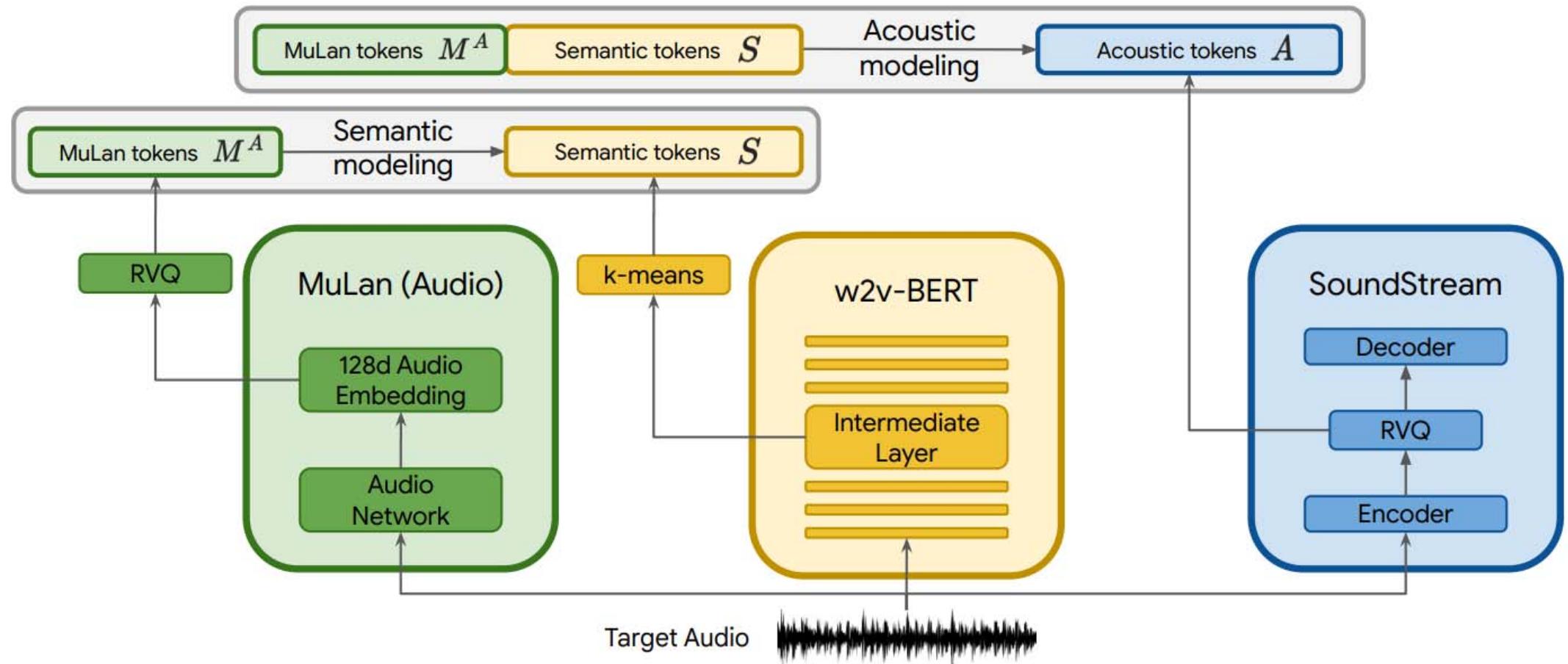
- Given a **text description of music**, generate **waveforms** directly
  - “Piano for study” | “Hip-hop song with violin solo” | “Latin workout” | “Feel-good mandopop indie” | “Salsa for broken hearts”
- **Story mode**
  - Text prompts
    - time to meditate (0:00-0:15)
    - time to wake up (0:15-0:30)
    - 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



# MusicLM's Pretrained Networks

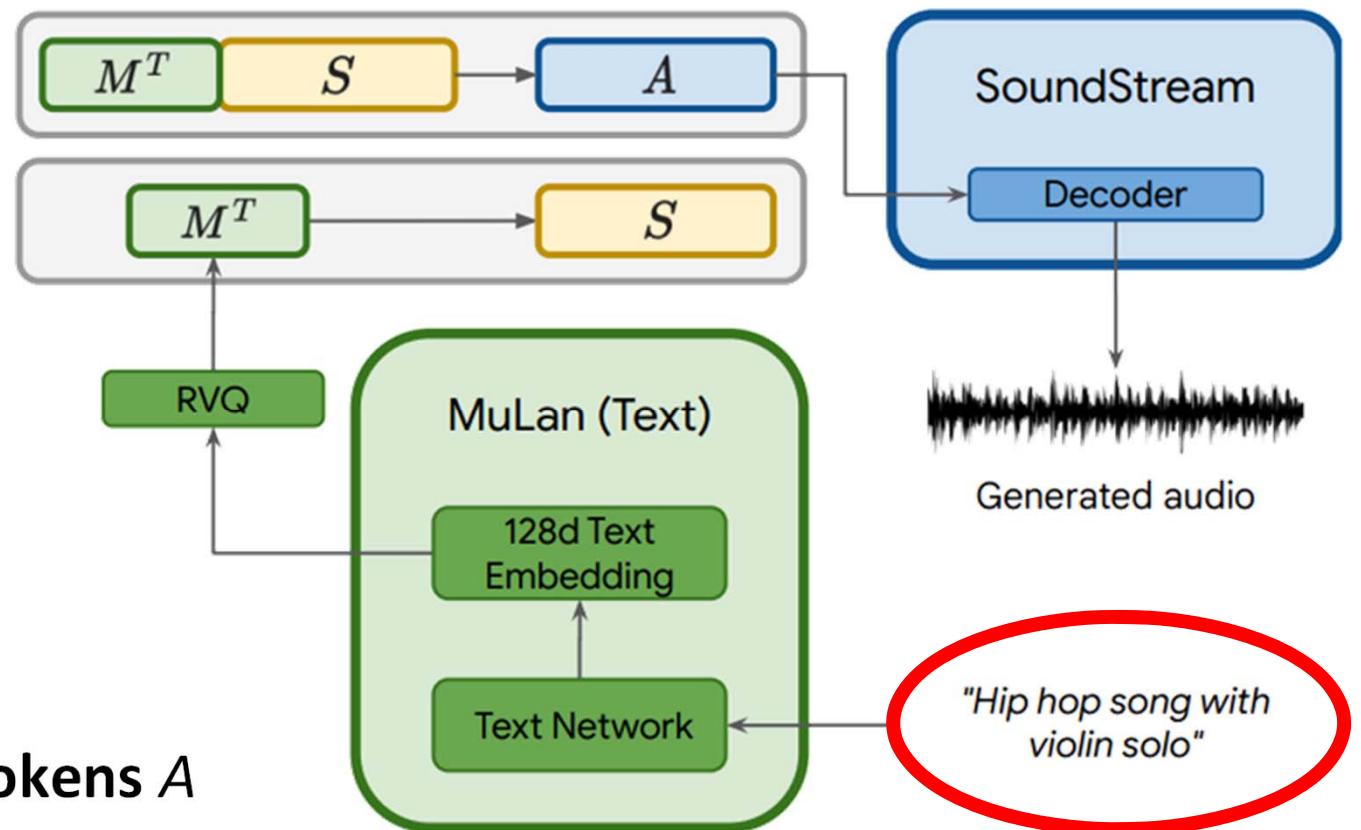


# MusicLM Training Procedure



# MusicLM Inference Procedure

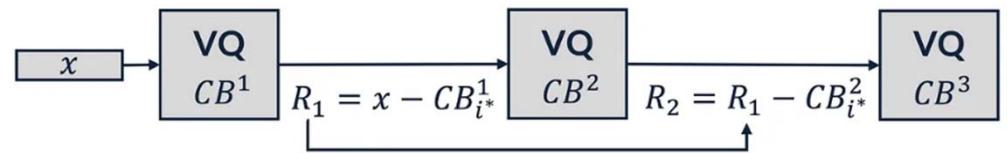
- Given **text tokens**  $M^T$  from the input text description
- Generate high-level **semantic tokens**  $S$  that model the long-term structure of the music to be generated
- Then generate **acoustic tokens**  $A$  that model the fine details



# Jukebox vs MusicLM

- **Jukebox**

- 3 layers of **acoustic tokens**: high, mid, bottom
  - The three layers operate in **different** temporal resolution
  - About **7,234** tokens per second



- **MusicLM**

- **600 acoustic tokens** per second
    - 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)
  - Plus **25 semantic tokens** per second
    - Semantic tokens from w2v-BERT (codebook size also 1024)

# Issues of MusicLM

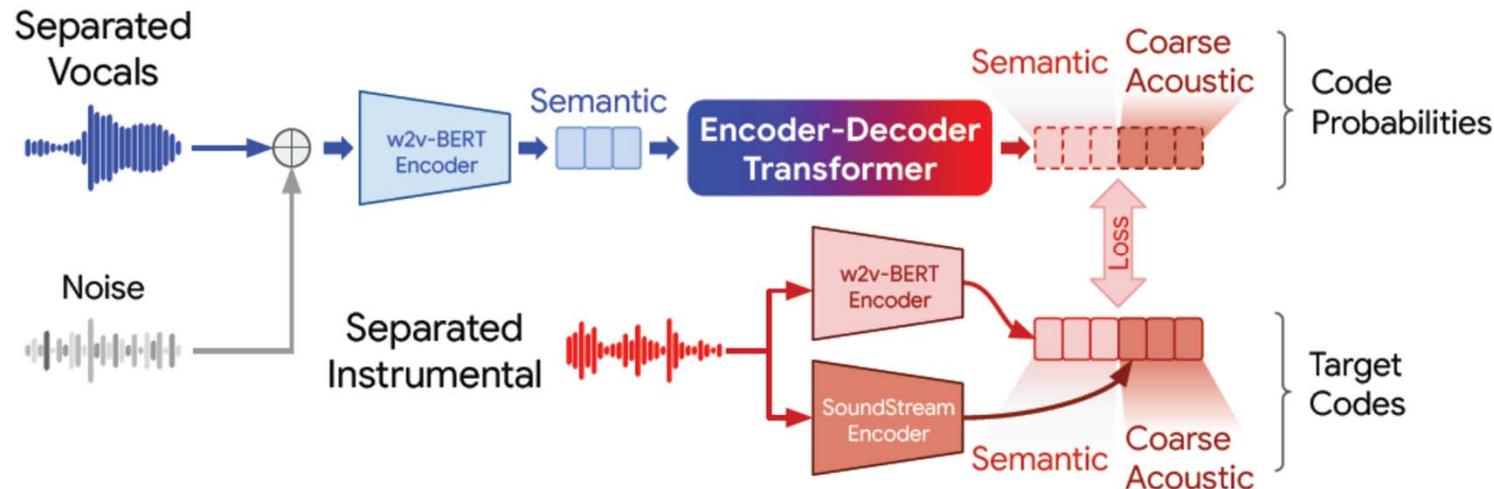
- Not open source; but there are unofficial implementations
  - <https://github.com/lucidrains/musiclm-pytorch>
  - <https://github.com/zhvng/open-musiclm>
- Slow (one second 625 tokens; i.e., **625** times of autoregressive sampling)
- Limited controllability
  - The main control signal is the text 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

## 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.”

# SingSong (from Google)

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

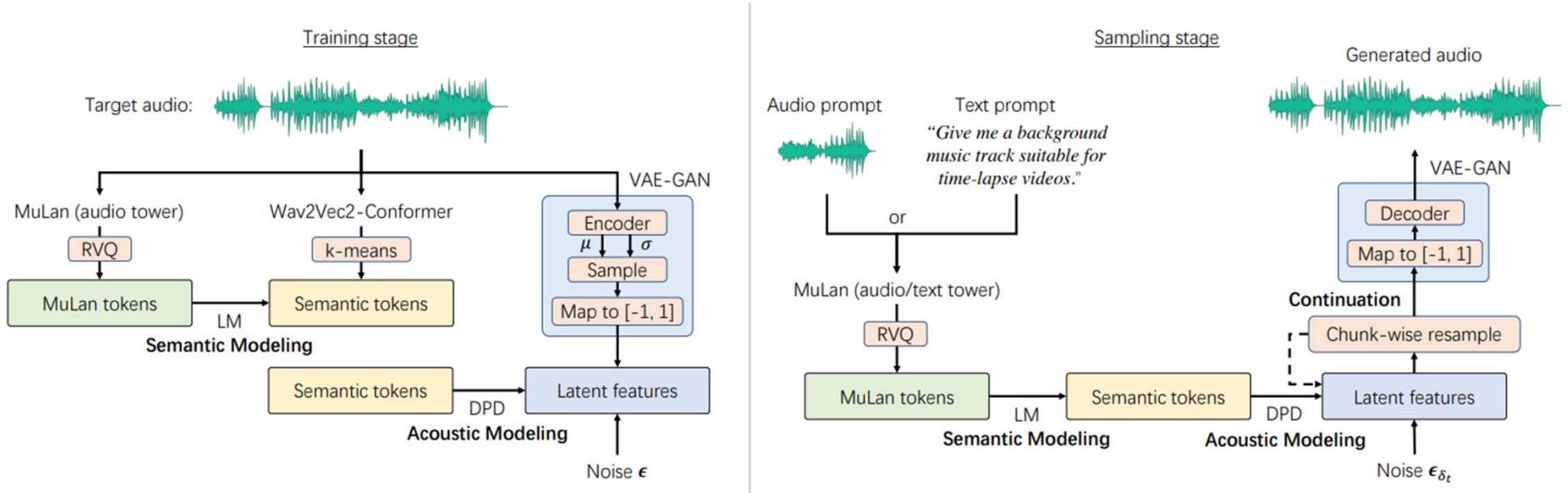


- **Given vocal, generate multi-instrument accompaniment**
  - Apply “blind source separation” to get pairs of vocal/backing tracks
  - Seq2seq: **vocal tokens** as the source, generate the backing tokens
- **Not open source**

# MeLoDy (from ByteDance)

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

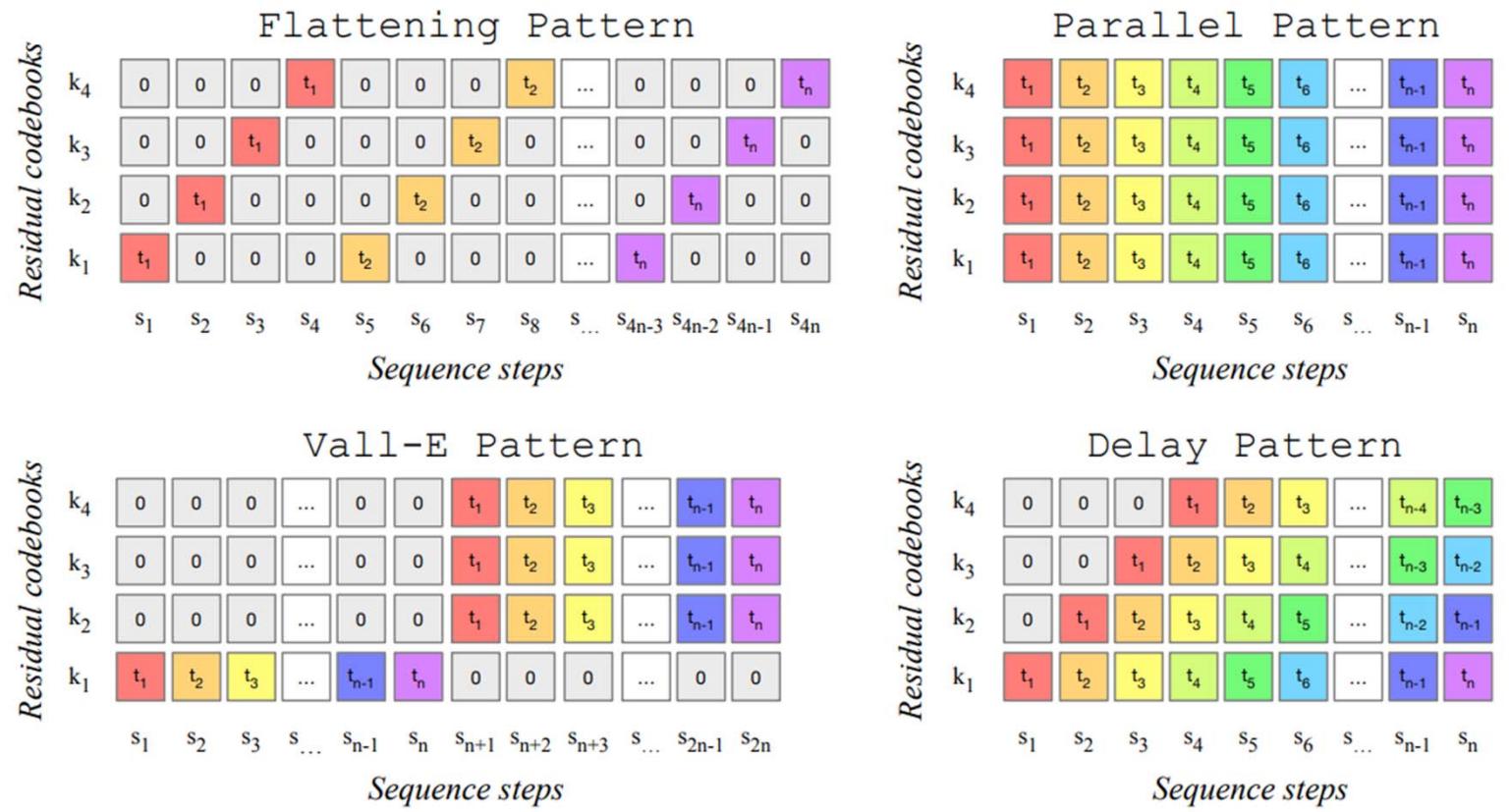
- Replace the LM for acoustic tokens in MusicLM by **latent diffusion**
- **Not open source**



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

# MusicGen (from Meta)

- Compound word (CP)-like **parallel prediction** of the Encodenc's RVQ tokens
- Open source!**



# MusicGen (from Meta)

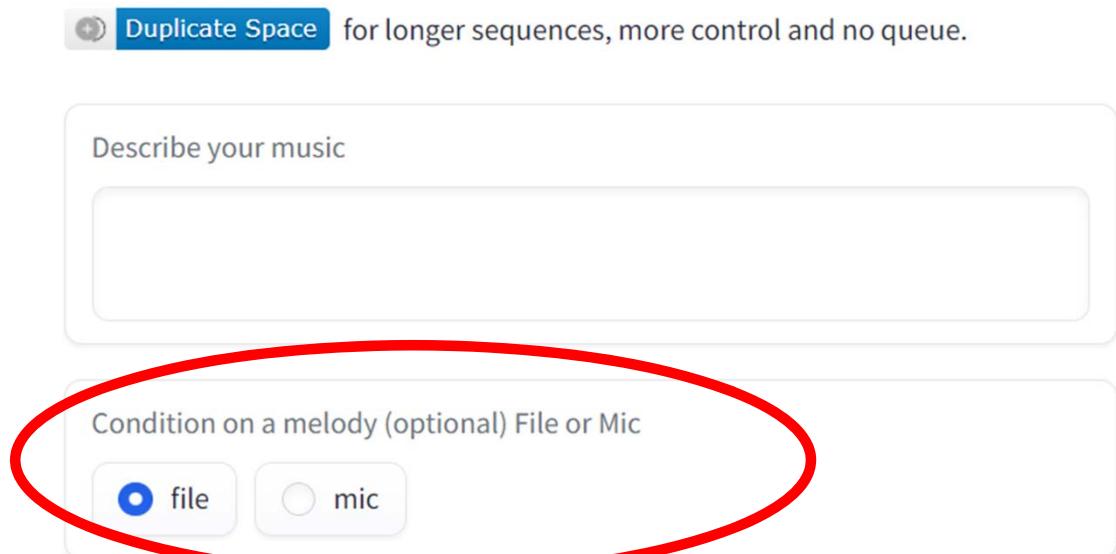
<https://huggingface.co/spaces/facebook/MusicGen>

<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

## MusicGen

This is the demo for [MusicGen](#), a simple and controllable model for music generation ↗



# MusicGen: Audio Encoder

- Audio encoder
  - We use a non-causal five layers **EnCodec** model for 32 kHz monophonic audio with a stride of 640, resulting in a frame rate of **50 Hz**
  - The embeddings are quantized with an RVQ with **4 quantizers**, each with a codebook size of **2048**
  - Train the model on one-second audio segments cropped at random
  - One second → **50 times** of autoregressive sampling with *Parallel* or *Delay* pattern
- Transformer model
  - Flash attention from the xFormers
  - Train on 30-sec clips sampled at random from the full track (batch 192), for 1M steps
  - Train 300M, 1.5B and 3.3B parameter models, using respectively 32, 64 and 96 GPUs
  - Top-k sampling with keeping the top 250 tokens and a temperature of 1.0

# MusicGen

- While flattening improves generation, it comes at a high computational cost and similar performance can be reached for a fraction of this cost using a simple delay approach

| In Domain Test Set |           |                      |             |                       |                    |                   |
|--------------------|-----------|----------------------|-------------|-----------------------|--------------------|-------------------|
| CONFIGURATION      | Nb. steps | FAD <sub>vgg</sub> ↓ | KL ↓        | CLAP <sub>scr</sub> ↑ | OVL. ↑             | REL. ↑            |
| Delay              | 1500      | 0.96                 | 0.52        | 0.35                  | <b>79.69</b> ±1.46 | 79.67±1.41        |
| Partial Delay      | 1500      | 1.51                 | 0.54        | 0.32                  | 79.13±1.56         | 79.67±1.46        |
| Parallel           | 1500      | 2.58                 | 0.62        | 0.27                  | 72.21±2.49         | 80.30±1.43        |
| VALL-E             | 3000      | 1.98                 | 0.56        | 0.30                  | 74.42±2.28         | 76.55±1.67        |
| Partial Flattening | 3000      | 1.32                 | 0.54        | 0.34                  | 78.56±1.86         | 79.18±1.49        |
| Flattening         | 6000      | <b>0.86</b>          | <b>0.51</b> | <b>0.37</b>           | <b>79.71</b> ±1.58 | <b>82.03</b> ±1.1 |

| In Domain Test Set |       |       |          |             |                      |             |                       |                  |
|--------------------|-------|-------|----------|-------------|----------------------|-------------|-----------------------|------------------|
| Dim.               | Heads | Depth | # Param. | PPL↓        | FAD <sub>vgg</sub> ↓ | KL ↓        | CLAP <sub>scr</sub> ↑ | OVL. ↑           |
| 1024               | 16    | 24    | 300M     | 56.1        | 0.96                 | 0.52        | 0.35                  | 78.3±1.4         |
| 1536               | 24    | 48    | 1.5B     | 48.4        | 0.86                 | <b>0.50</b> | 0.35                  | <b>81.9</b> ±1.4 |
| 2048               | 32    | 48    | 3.3B     | <b>46.1</b> | <b>0.82</b>          | <b>0.50</b> | <b>0.36</b>           | 79.2±1.3         |

# MusicGen: Text Encoder

- **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
- 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.

| MODEL   | MUSiCCAPS Test Set   |             |                       |                     |                     |
|---------|----------------------|-------------|-----------------------|---------------------|---------------------|
|         | FAD <sub>vgg</sub> ↓ | KL ↓        | CLAP <sub>scr</sub> ↑ | OVL. ↑              | REL. ↑              |
| T5      | <b>3.12</b>          | <b>1.29</b> | 0.31                  | $84.89 \pm 1.78$    | <b>82.52 ± 1.31</b> |
| Flan-T5 | 3.36                 | 1.30        | <b>0.32</b>           | <b>86.25 ± 1.75</b> | $80.83 \pm 1.88$    |
| CLAP    | 4.23                 | 1.53        | <b>0.32</b>           | $79.79 \pm 1.76$    | $77.28 \pm 1.54$    |

# Jukebox vs. MusicLM vs. MusicGen

|                     | <b>Jukebox (2020)</b>   | <b>MusicLM (2023.01)</b>   | <b>MusicGen (2023.06)</b>  |
|---------------------|---|--|--|
| <b>Audio tokens</b> | <b>7,234 acoustic tokens</b><br>(3-layer hierarchical VQVAE: high, mid, bottom) | <b>600 acoustic tokens</b><br>(12-layer SoundStream codec)<br>+ <b>25 semantic tokens</b><br>(w2v-BERT) per second     | <b>200 acoustic tokens</b><br>(4-layer EnCodec codec)              |
| <b>Text tokens</b>  | No text tokens  | <b>Text tokens representing the user input</b> (via VQed MuLan: contrastive learning based joint text-audio embedding) | <b>Text tokens representing the user input</b> (via T5 or FLAN-T5) |
| <b>LM</b>           | 72-layer Transformer  | 24-layer Transformer   | 24- or 48- layer Transformer                                       |
| <b>#Param.</b>      | >6B   | N/A  | 300M, 1.5B, or 3.3B  |

# MusicGen Tutorial

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

AI Music Generation Audiodraft & MusicGen Tutorial with Example (Free Text-to-Music Model)



SECourses  
28K subscribers

Join

Subscribe

431



Share

...

21K views 4 months ago Technology & Science: Tutorials, Tips, Guides, Tricks, Best Applications, Reviews GitHub instructions Readme file and Patreon Auto installer updated at 4 August 2023.

Facebook Meta Research has published the new amazing text-to-music model Audiodraft (MusicGen). In this video I have shown how you can install Audiodraft on your computer or use it on the Google Cola ...[more](#)

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

# 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)                    | ✓   | ✓  | ✓  | ✓   | ✓   | ✓        | ✓        | ✓      | ✓  | ✓     | ✓      |

# 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          |
| M4Singer Zhang et al. (2022)                   | singing               | text, MIDI        | 1h           |

# 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**

# Denoising Diffusion Probabilistic Models (DDPM)

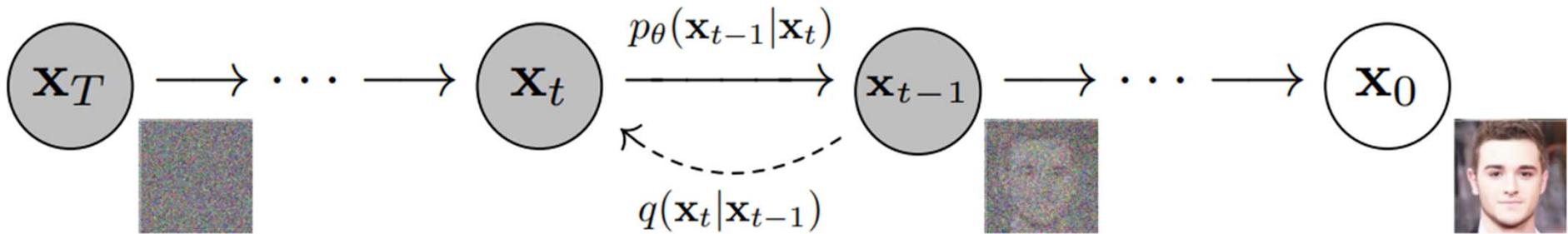


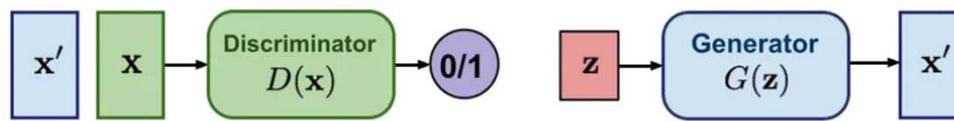
Figure 2: The directed graphical model considered in this work.

- **Forward** (or diffusion) process:  $\mathbf{x}_0$  (real data)  $\rightarrow \mathbf{x}_T$  (Gaussian noise)
- **Reverse** process:  $\mathbf{x}_T \rightarrow \mathbf{x}_0$  [denoise]
- The intermediate noisy data serve as latent codes ( $\mathbf{z}$ ) and have the same size as the training data

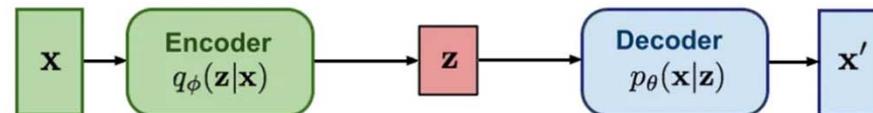
# Diffusion Models

<https://pub.towardsai.net/diffusion-models-vs-gans-vs-vaes-comparison-of-deep-generative-models-67ab93e0d9ae>

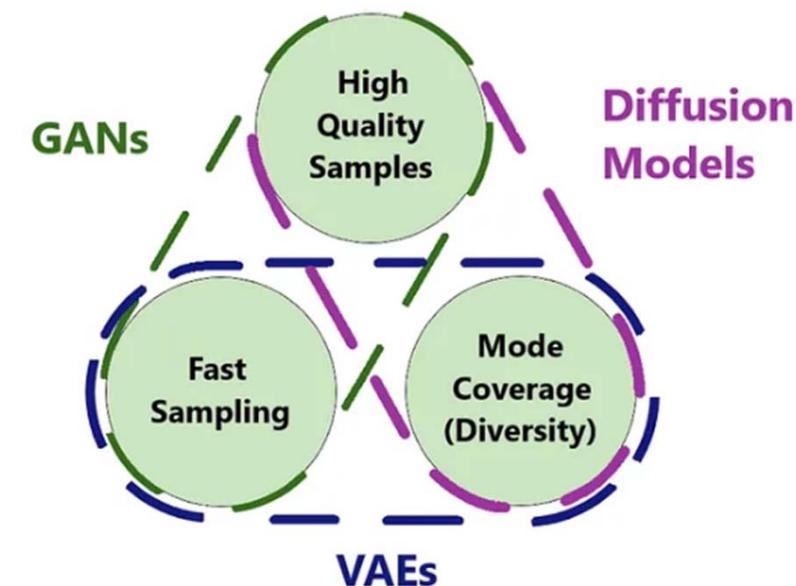
**GAN:** Adversarial training



**VAE:** maximize variational lower bound



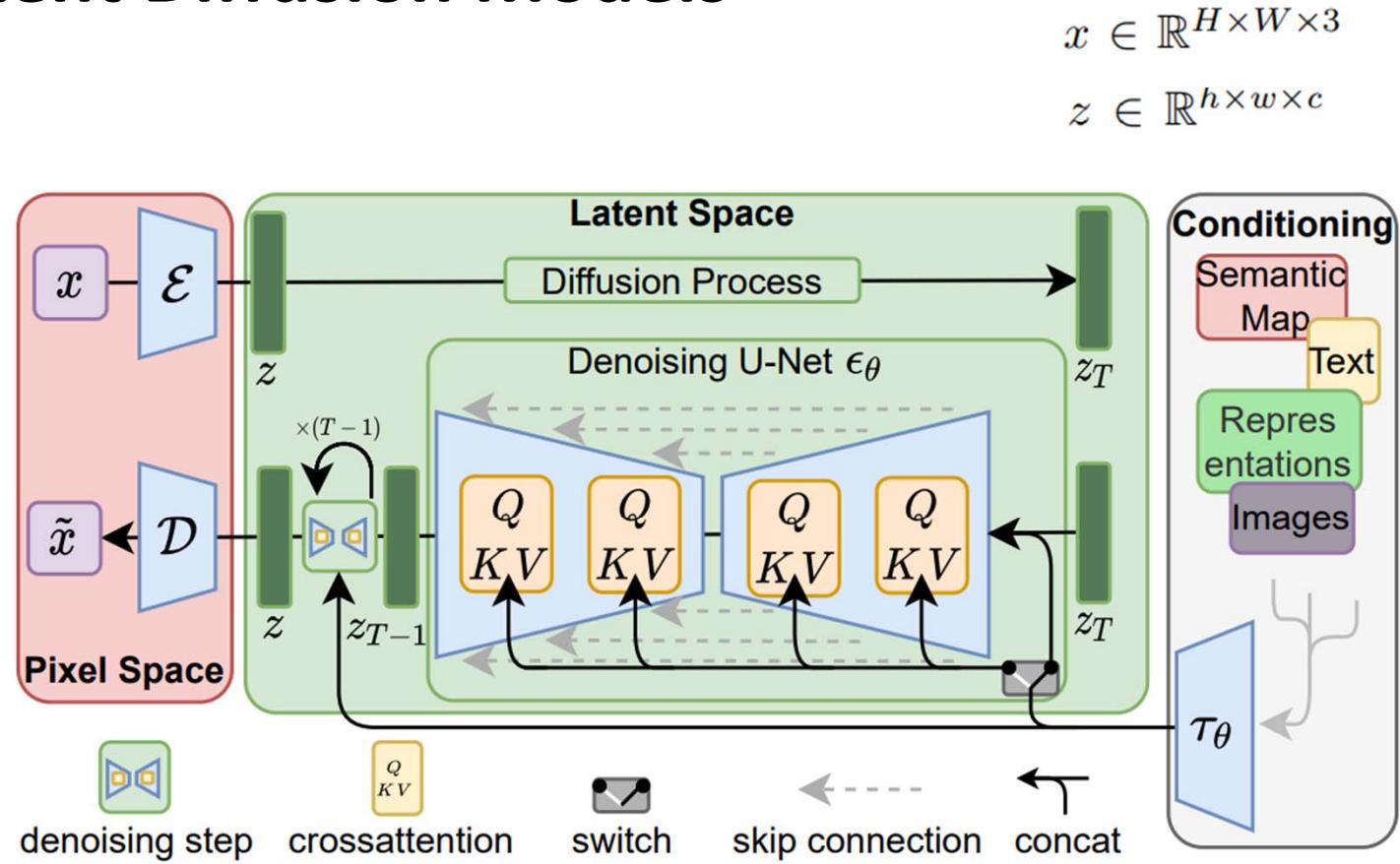
**Diffusion models:**  
Gradually add Gaussian noise and then reverse



- **High-fidelity samples:** due to the nature of gradually removing noise
- **High diversity samples:** likelihood maximization covers all data modes
- **Slow sample generation:** requires multiple runs to gradually generate samples

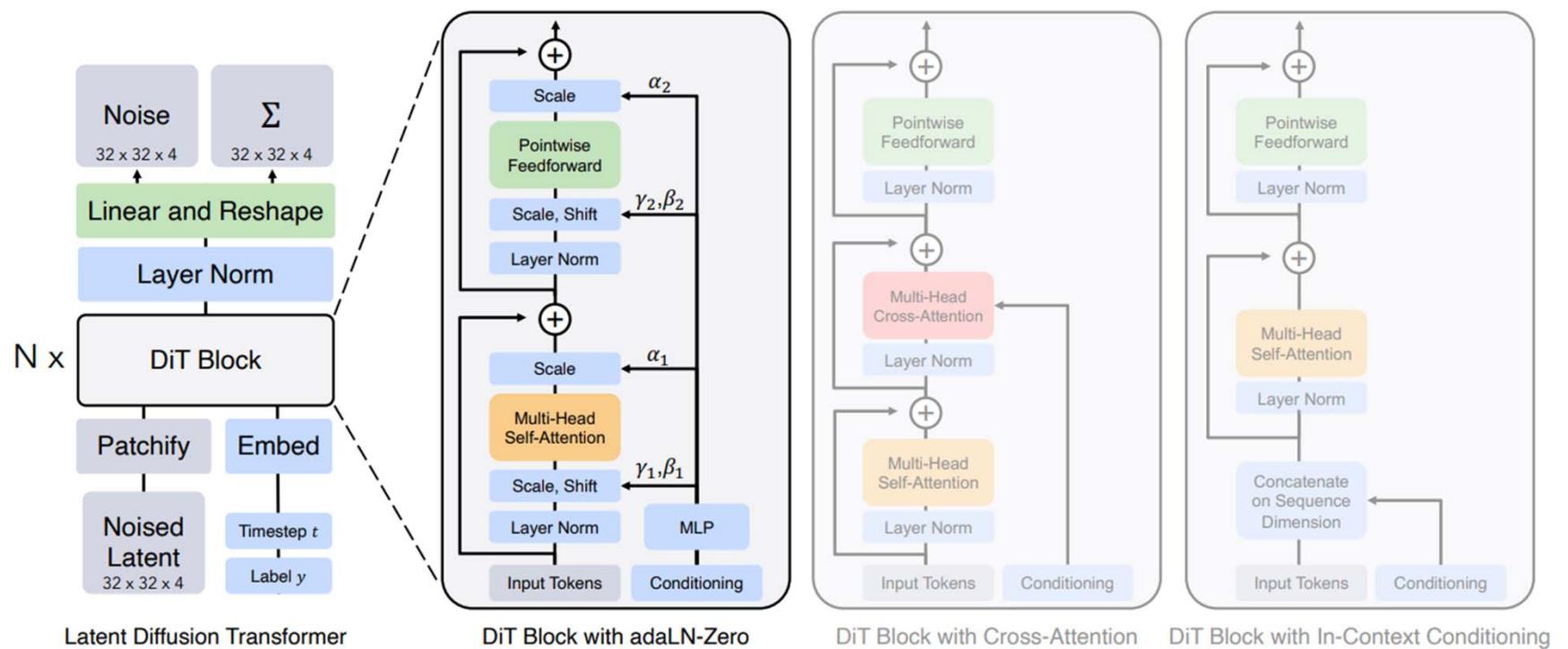
# Latent Diffusion Models

- Work on a **latent** space instead of pixel space
  - focus on the important, semantic bits of data
  - computational efficiency
- Latent space regularization
  - KL-reg
  - VQ-reg
- Adding condition
  - cross-attention

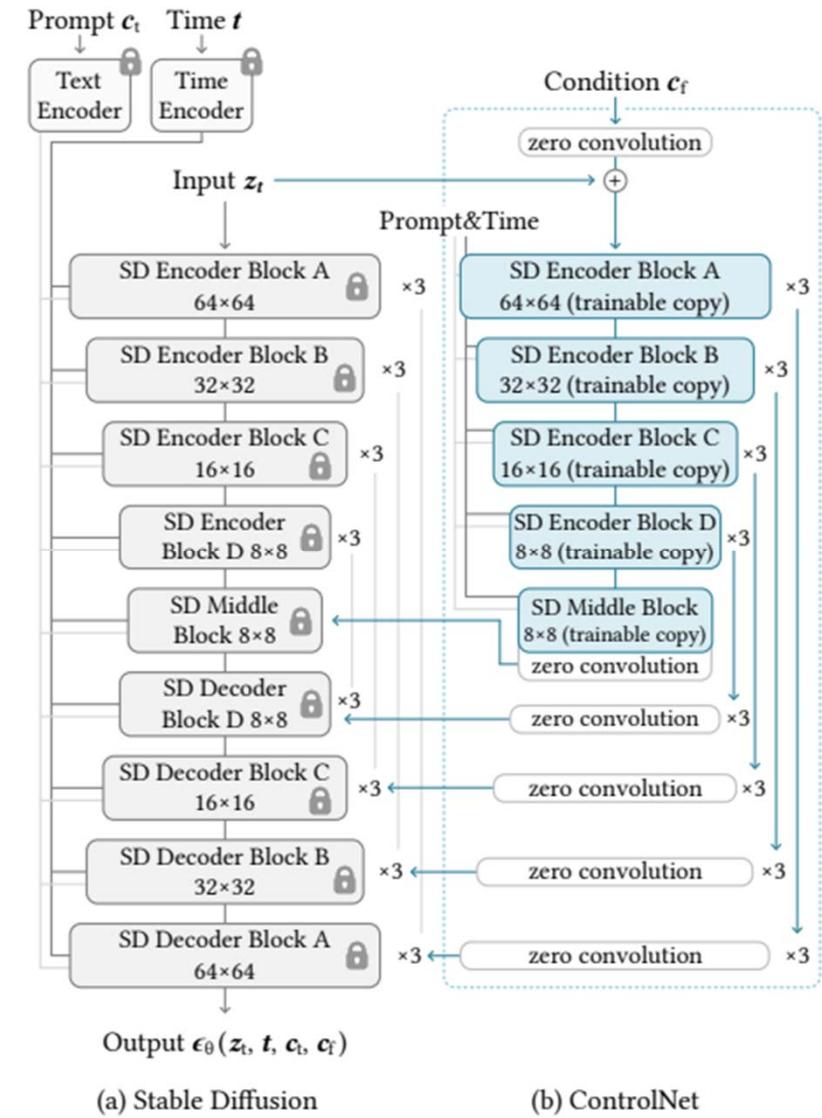
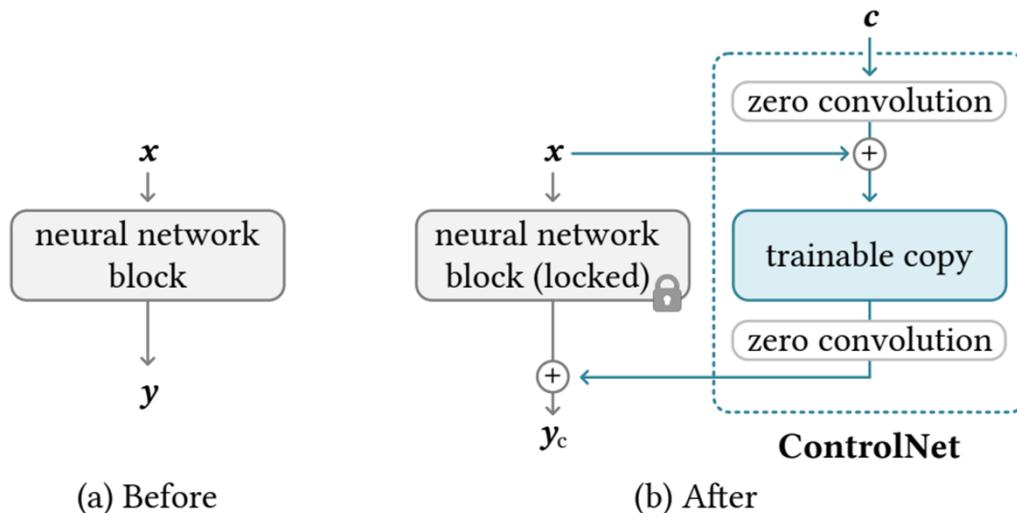


# Diffusion Transformer (DiT)

- Self-attention layers instead of convolution layers
- Use adaptive layer norm to incorporate conditioning

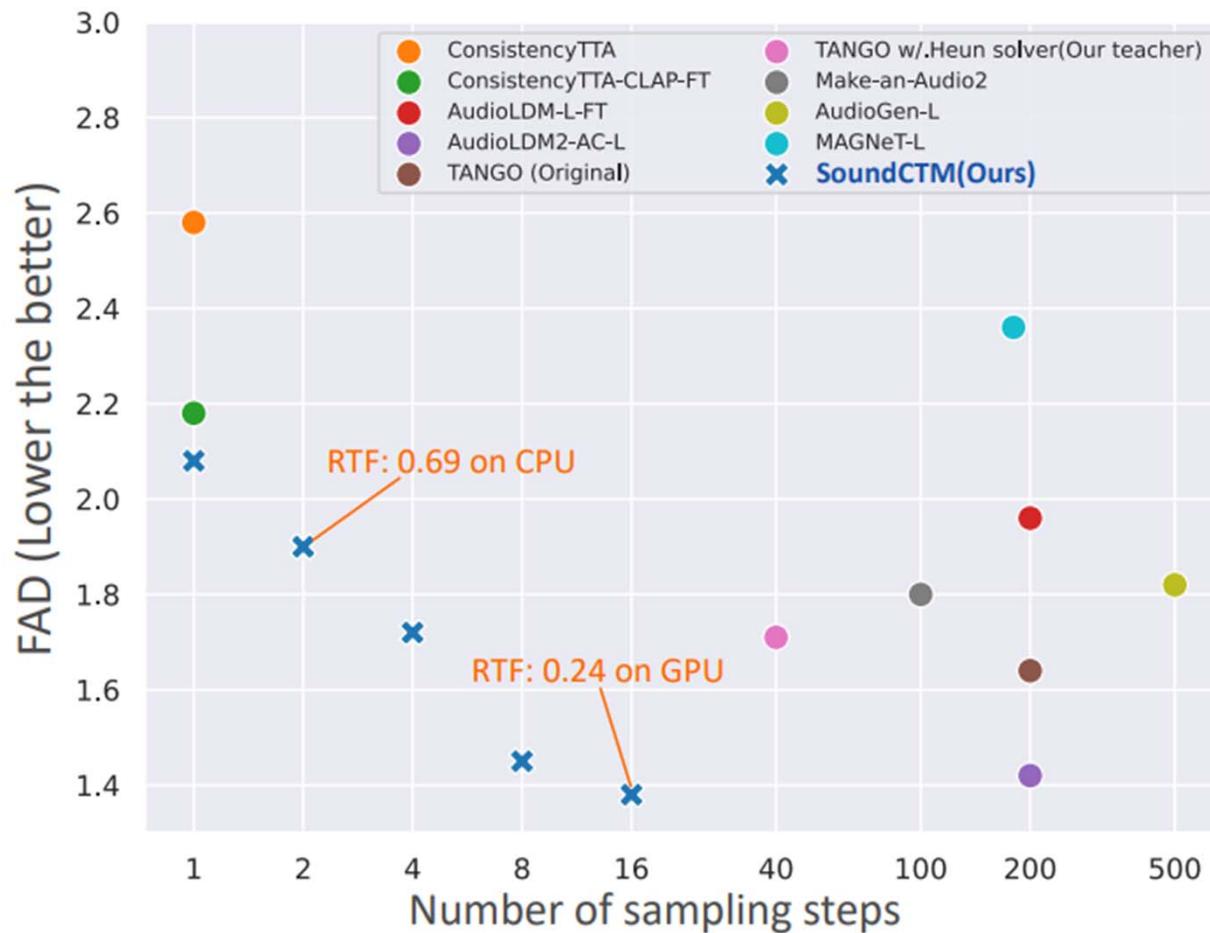


# ControlNet



Ref: Zhang et al, "Adding conditional control to text-to-image diffusion models," CVPR 2023

# Consistency Model: One-Step Generation



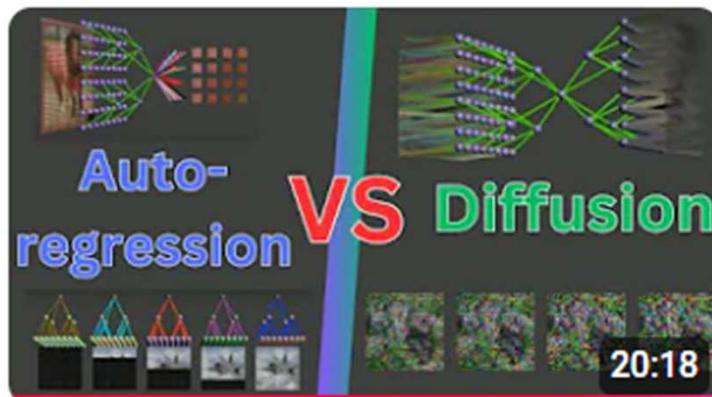
Ref: Saito et al, "SoundCTM: Uniting score-based and consistency models for text-to-sound generation," arXiv 2024

# References for Diffusion Models

- References
  - Diffusion model tutorial @ ISMIR 2024 (<https://sites.google.com/view/diffusion-tutorial-ismir24/materials>)A thumbnail image for a video titled "FROM WHITE NOISE TO SYMPHONY" featuring a treble clef icon. Below the title is the subtitle "DIFFUSION MODELS FOR MUSIC AND SOUND". At the bottom, it says "November 10 @ISMIR 2024".
  - Prof. Hung-Yi Lee's videos (<https://www.youtube.com/watch?v=ifCDXFdeaaM>)Four thumbnail images for a series of four YouTube videos. Each thumbnail has a light blue background with a circular wood-grain pattern. The text "Diffusion Model 原理剖析" is followed by a fraction (e.g., "(4/4)", "(3/4)", "(2/4)", "(1/4)"). A small black rectangle at the bottom right of each thumbnail contains the video duration: "17:48", "25:00", "27:08", and "13:56" respectively.

# References for Diffusion Models

<https://www.youtube.com/watch?v=zc5NTeJbk-k>



## Why Does Diffusion Work Better than Auto-Regression?

Algorithmic Simplicity • 374K views

Have you ever wondered how generative AI actually works? Well the short answer is, in exactly the same way as regular AI! In this...

# Diffusers

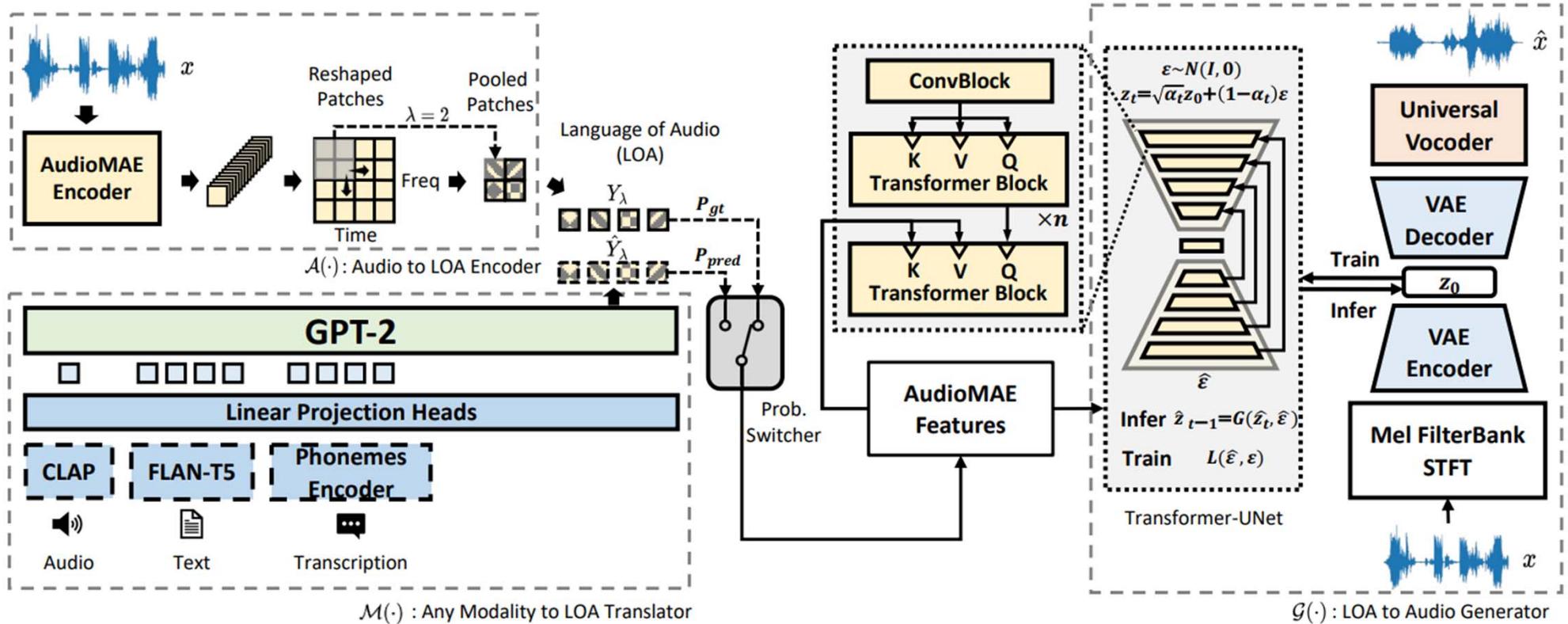
<https://github.com/huggingface/diffusers>

| Task                           | Pipeline   | 🤗 Hub   |
|--------------------------------|--|---|
| Unconditional Image Generation | <a href="#">DDPM</a>                             | <a href="#">google/ddpm-ema-church-256</a>                  |
| Text-to-Image                  | <a href="#">Stable Diffusion Text-to-Image</a>   | <a href="#">stable-diffusion-v1-5/stable-diffusion-v1-5</a> |
| Text-to-Image                  | <a href="#">unCLIP</a>                           | <a href="#">kakaobrain/karlo-v1-alpha</a>                   |
| Text-to-Image                  | <a href="#">DeepFloyd IF</a>                     | <a href="#">DeepFloyd/IF-I-XL-v1.0</a>                      |
| Text-to-Image                  | <a href="#">Kandinsky</a>                        | <a href="#">kandinsky-community/kandinsky-2-2-decoder</a>   |
| Text-guided Image-to-Image     | <a href="#">ControlNet</a>                       | <a href="#">Illyasviel/sd-controlnet-canny</a>              |
| Text-guided Image-to-Image     | <a href="#">InstructPix2Pix</a>                  | <a href="#">timbrooks/instruct-pix2pix</a>                  |
| Text-guided Image-to-Image     | <a href="#">Stable Diffusion Image-to-Image</a>  | <a href="#">stable-diffusion-v1-5/stable-diffusion-v1-5</a> |
| Text-guided Image Inpainting   | <a href="#">Stable Diffusion Inpainting</a>      | <a href="#">runwayml/stable-diffusion-inpainting</a>        |
| Image Variation                | <a href="#">Stable Diffusion Image Variation</a> | <a href="#">lambdalabs/sd-image-variations-diffusers</a>    |
| Super Resolution               | <a href="#">Stable Diffusion Upscale</a>         | <a href="#">stabilityai/stable-diffusion-x4-upscaler</a>    |
| Super Resolution               | <a href="#">Stable Diffusion Latent Upscale</a>  | <a href="#">stabilityai/sd-x2-latent-upscaler</a>           |

# AudioLDM2

<https://github.com/haoheliu/AudioLDM2>

- Open source!
- Latent diffusion

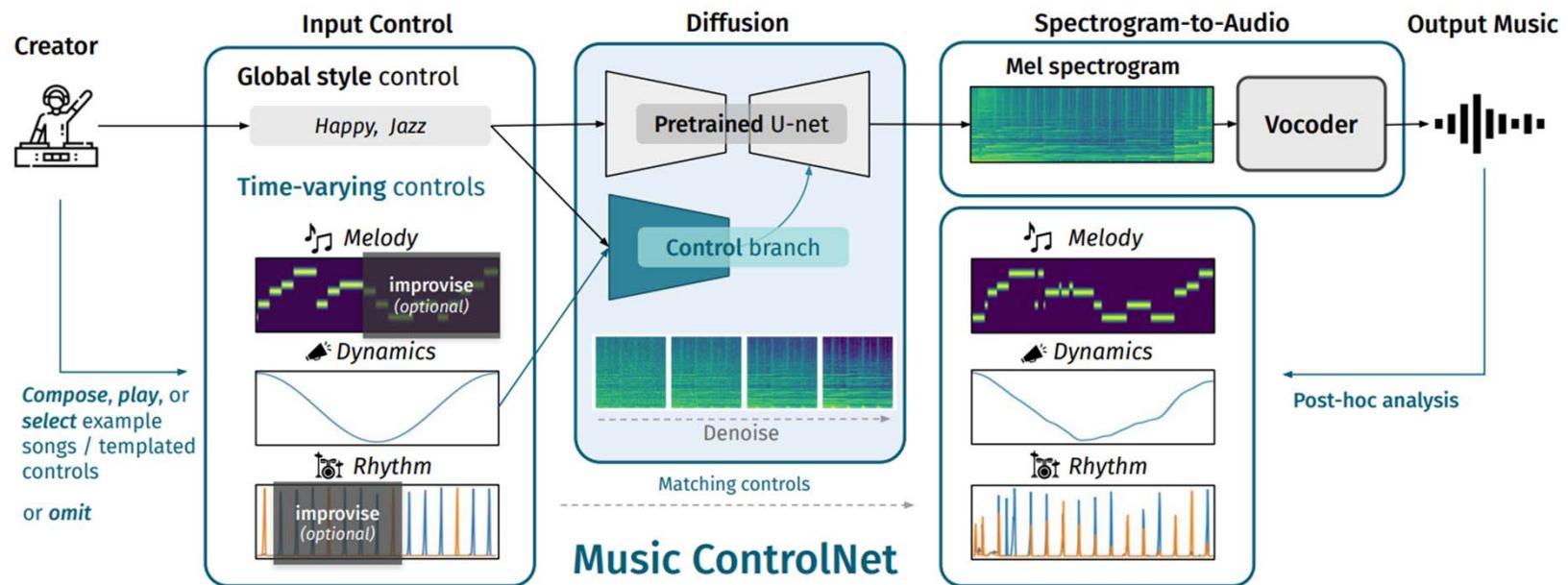


Ref: Liu et al, “AudioLDM 2: Learning holistic audio generation with self-supervised pretraining,” TASLP 2024

# Music ControlNet (for Attribute Control)

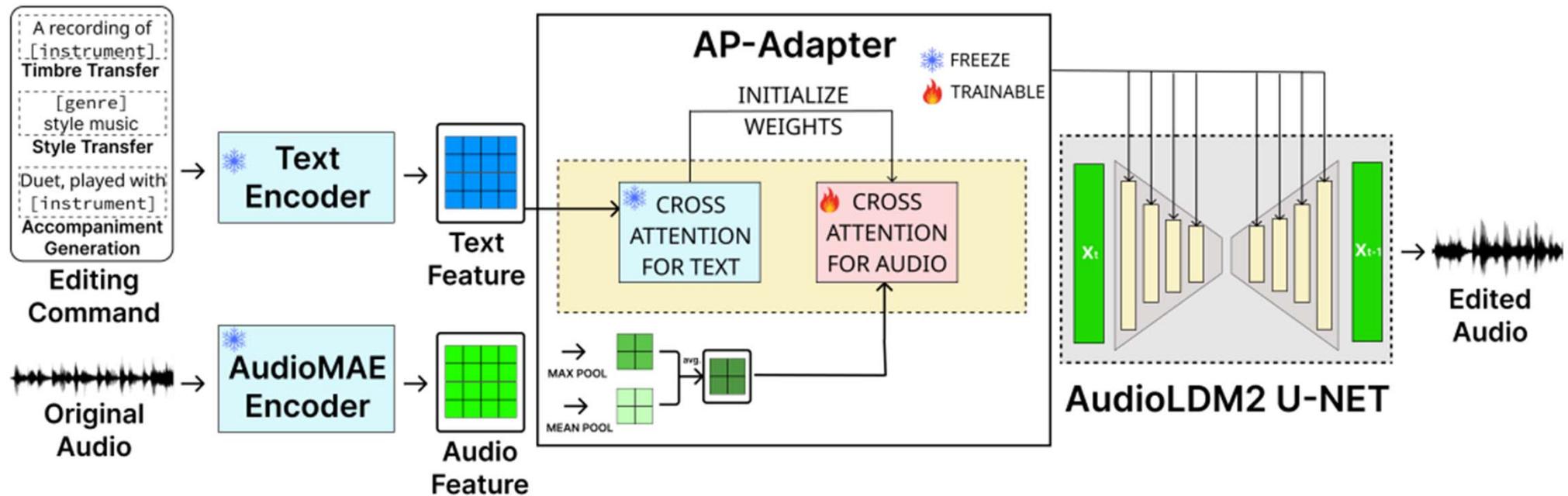
<https://musiccontrolnet.github.io/web/>

- Work in the “pixel” domain
- Not open source



# AP-Adapter (for Text+Audio Control)

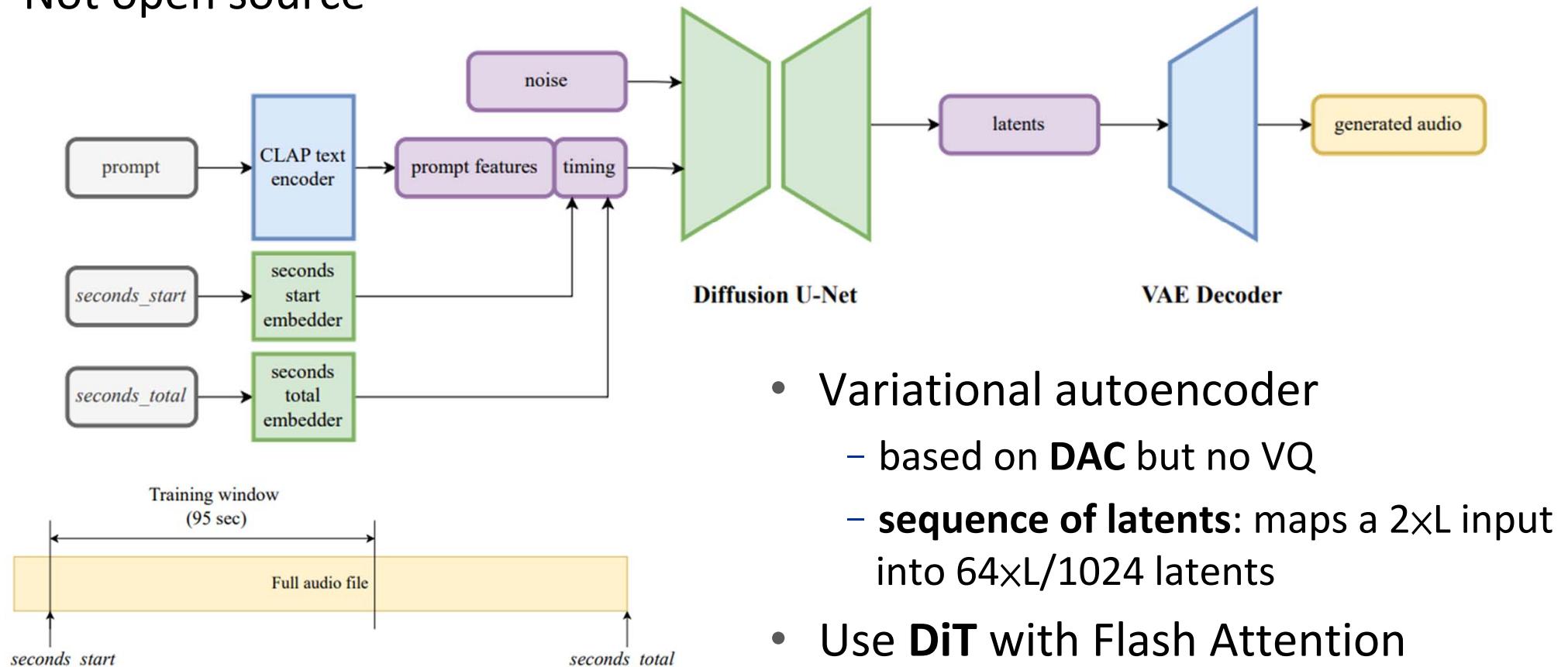
<https://young-almond-689.notion.site/Audio-Prompt-Adapter-Unleashing-Music-Editing-Abilities-For-Text-To-Music-with-Lightweight-Finetuni-fbbfeb0608664f61a6bf894d56e85820>



Ref: Tsai et al, "Audio Prompt Adapter: Unleashing music editing abilities for text-to-music with lightweight finetuning," ISMIR 2024

# Stable Audio

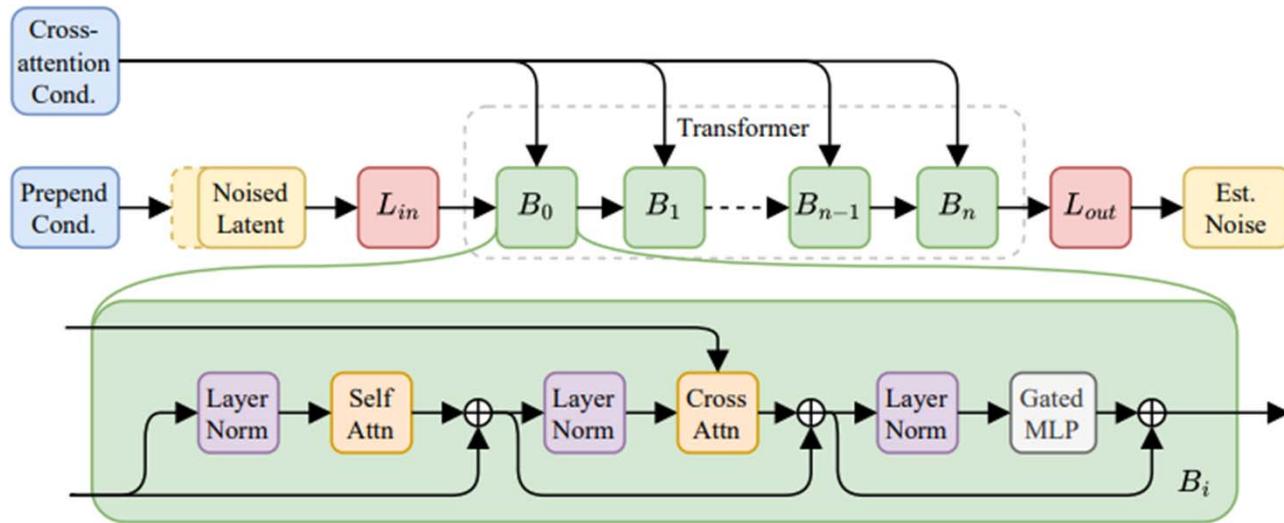
- Not open source



# Stable Audio 2

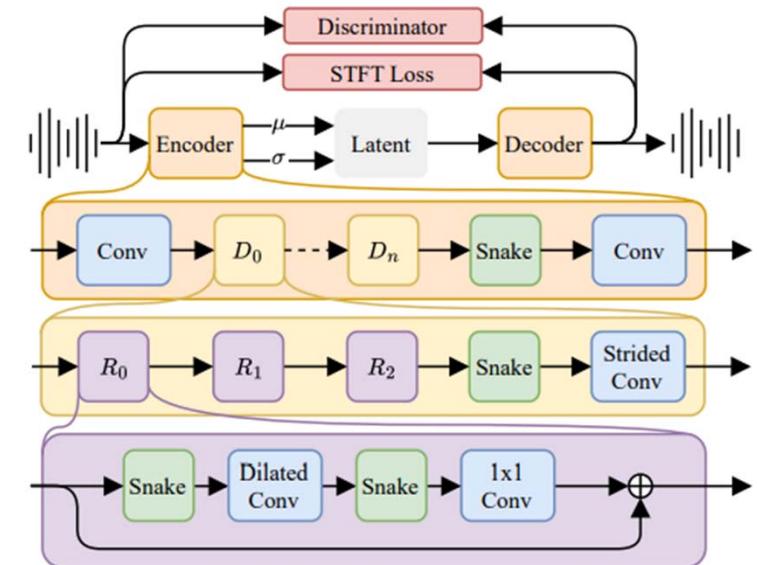
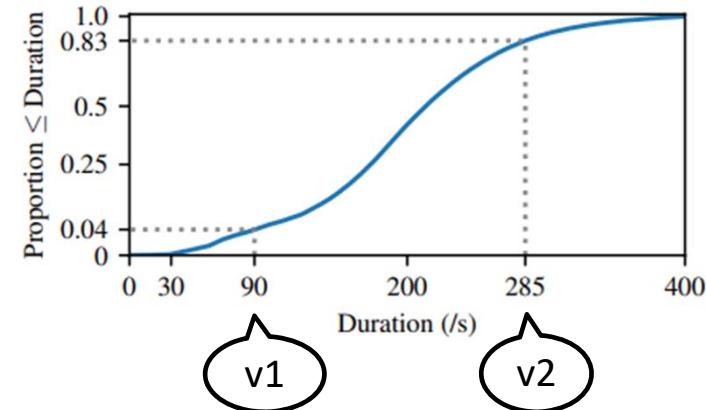
- Not open source

|            | DiT  | AE   | CLAP |  | Total |
|------------|------|------|------|--|-------|
| Parameters | 1.1B | 157M | 125M |  | 1.3B  |



**Figure 2:** Architecture of the diffusion-transformer (DiT).

Ref: Evans et al, "Long-form music generation with latent diffusion," ISMIR 2024



**Figure 3:** Architecture of the autoencoder.

# Stable Audio 2 vs MusicGen-large

|                           | channels/sr | output length   | $FD_{openl3} \downarrow$ | $KL_{passt} \downarrow$ | $CLAP_{score} \uparrow$ | inference time |
|---------------------------|-------------|-----------------|--------------------------|-------------------------|-------------------------|----------------|
| MusicGen-large-stereo [8] | 2/32kHz     | 2m              | 204.03                   | 0.49                    | 0.28                    | 6m 38s         |
| Ours (fully-trained)      | 2/44.1kHz   | 2m <sup>†</sup> | 79.09                    | 0.35                    | 0.40                    | 13s            |
| MusicGen-large-stereo [8] | 2/32kHz     | 4m 45s          | 218.02                   | 0.50                    | 0.27                    | 12m 53s        |
| Ours (fully-trained)      | 2/44.1kHz   | 4m 45s          | 81.96                    | 0.34                    | 0.39                    | 13s            |

| Results with the fully-trained model: | 2m long           |                           |                 | 4m 45s long       |                           |
|---------------------------------------|-------------------|---------------------------|-----------------|-------------------|---------------------------|
|                                       | Stable<br>Audio 2 | MusicGen-<br>large-stereo | ground<br>truth | Stable<br>Audio 2 | MusicGen-<br>large-stereo |
| Audio Quality                         | 4.0±0.6           | 2.8±0.8                   | 4.6±0.4         | 4.5±0.4           | 2.8±0.8                   |
| Text Alignment                        | 4.3±0.7           | 3.1±0.8                   | 4.6±0.5         | 4.6±0.4           | 2.9±1.0                   |
| Structure                             | 3.5±1.3           | 2.4±0.7                   | 4.3±0.8         | 4.0±1.0           | 2.1±0.7                   |
| Musicality                            | 4.0±0.8           | 2.7±0.9                   | 4.6±0.5         | 4.3±0.7           | 2.6±0.7                   |
| Stereo correctness                    | 96%               | 61%                       | 96%             | 100%              | 57%                       |

# Stable Audio Open

- **Open source!**
  - trained on data from Freesound and FMA (about 500K recordings; or 7,300 h)
  - all licensed under CC-0, CC-BY, or CC-Sampling+

|                           | channels/sr | output length        | $FD_{openl3} \downarrow$ | $KL_{passt} \downarrow$ | $CLAP_{score} \uparrow$ |
|---------------------------|-------------|----------------------|--------------------------|-------------------------|-------------------------|
| MusicGen-large-stereo [8] | 2/32kHz     | 47                   | 190.47                   | 0.52                    | 0.31                    |
| Stable Audio 1.0 [5]      | 2/44.1kHz   | 95 sec <sup>†</sup>  | 142.50                   | 0.40                    | 0.38                    |
| Stable Audio 2.0 [6]      | 2/44.1kHz   | 190 sec <sup>†</sup> | <b>71.25</b>             | <b>0.37</b>             | <b>0.42</b>             |
| Stable Audio 2.0 [6]      | 2/44.1kHz   | 285 sec <sup>†</sup> | 81.05                    | 0.39                    | 0.42                    |
| Stable Audio Open         | 2/44.1kHz   | 47                   | 96.51                    | 0.55                    | 0.41                    |

**Table 2.** *Song Desciber Dataset (instrumental music, not sounds).*

# 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