MUSIC AUDIO GENERATION IN 2023: A SELECTIVE REVIEW

Jan Schlüter

54th Vienna Deep Learning Meetup

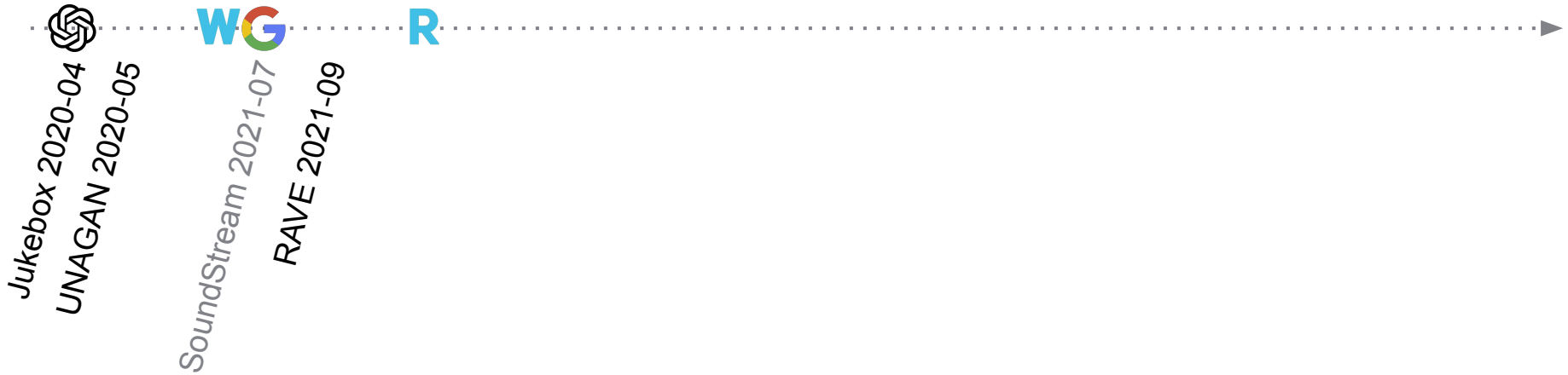
2023-10-18

JOHANNES KEPLER UNIVERSITY LINZ

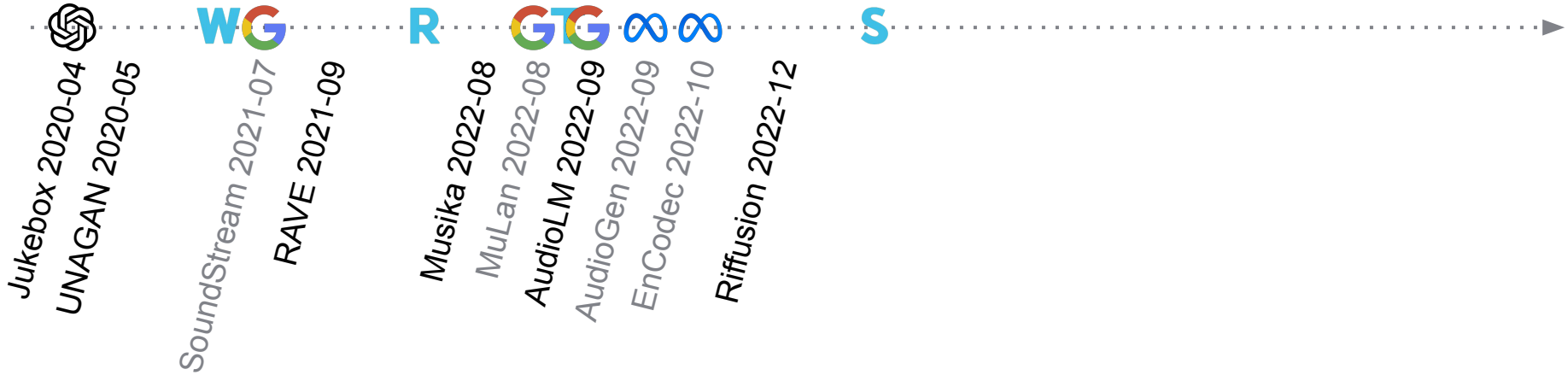
Institute of Computational Perception

(SELECTIVE) TIMELINE

(SELECTIVE) TIMELINE



(SELECTIVE) TIMELINE



(SELECTIVE) TIMELINE

(SELECTIVE) TIMELINE

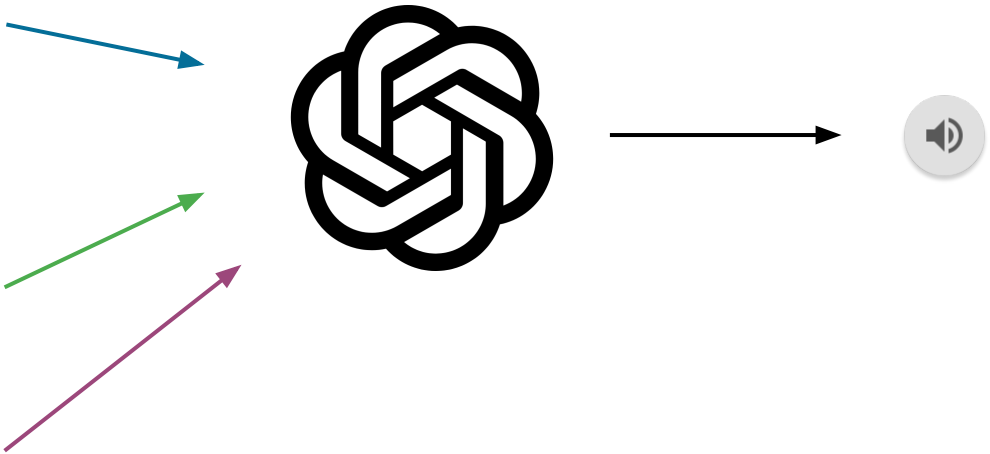
https://arxiv.org/abs/2005.00341

JUKEBOX

Lyrics:
It's Christmas time, and you know what that means,
Ohh, it's hot tub time!
As I light the tree, this year we'll be in a tub,
Ohh, it's hot tub time!

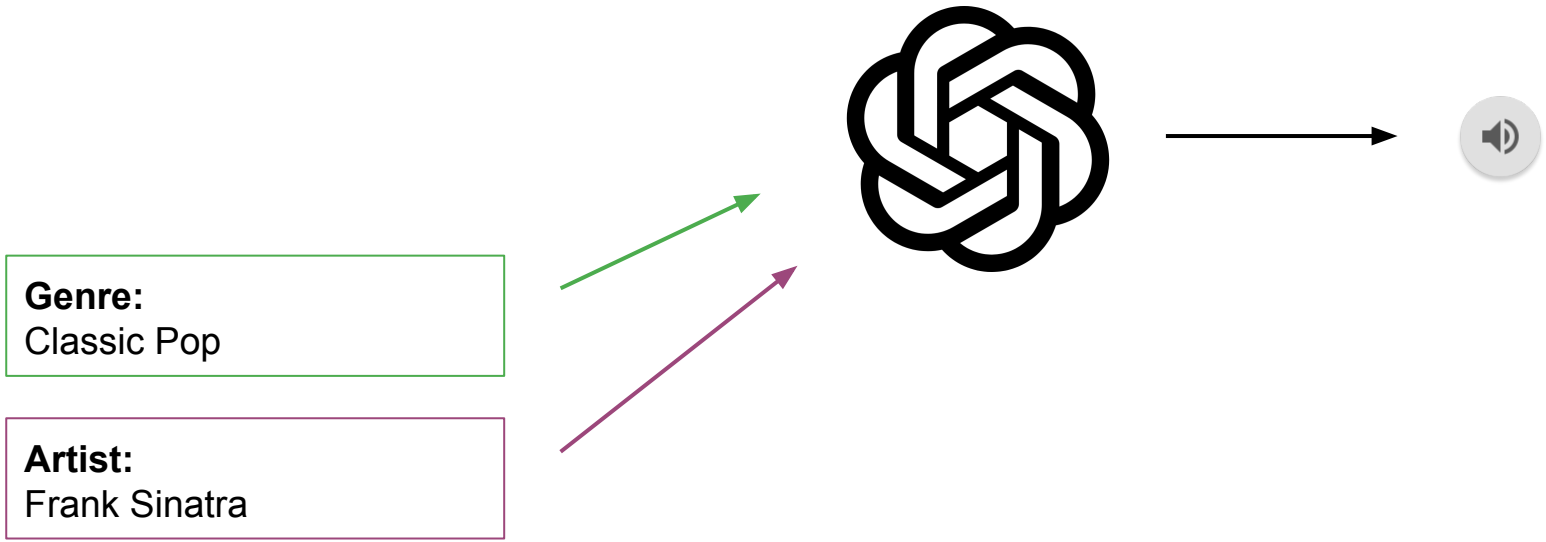
Genre:
Classic Pop

Artist:
(can you guess it?)



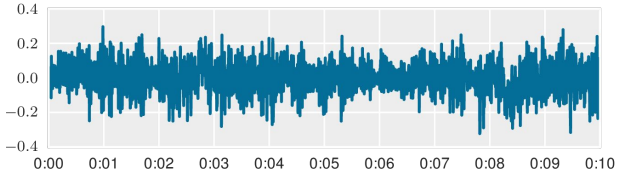
https://arxiv.org/abs/2005.00341

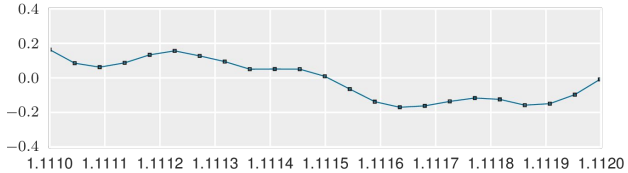
JUKEBOX



WHY IS THIS HARD?

Waveform: Sound represented as pressure variations in a medium measured at a regular time interval (e.g., 44100 times per second, 2^16 possible values)

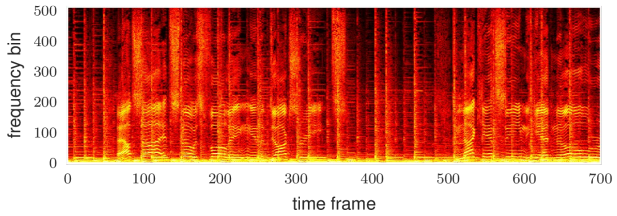




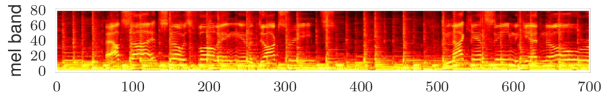
- Raw audio data is extremely high-dimensional compared to text or images (a 224x224 RGB image has as many bits as 1.7s of audio at 44.1 kHz, 16 bit)
- Music features long-range dependencies spanning up to several minutes (consistent key, tempo, rhythm, instrumentation, song structure, lyrics)

HANDLING AUDIO AS SPECTROGRAMS

Spectrogram: Sound represented as distribution of energy over frequencies at a regular time interval (e.g., 70 times per second, 513 frequency bands)



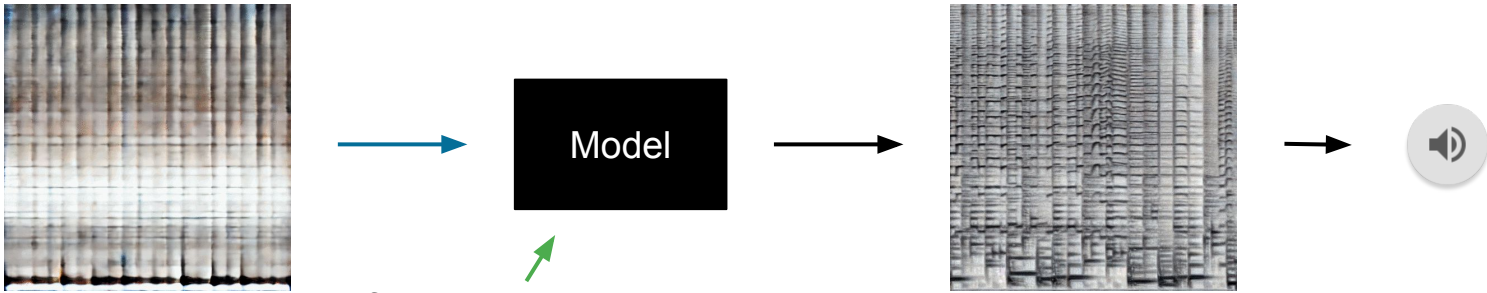
Mel spectrogram: Project to fewer bands (e.g., 80), condensing high frequencies



Mel spectrograms are more manageable (a 224x224 RGB image has as many pixel values as 26.88s of audio at 70 fps and 80 bands), but lossy

RIFFUSION

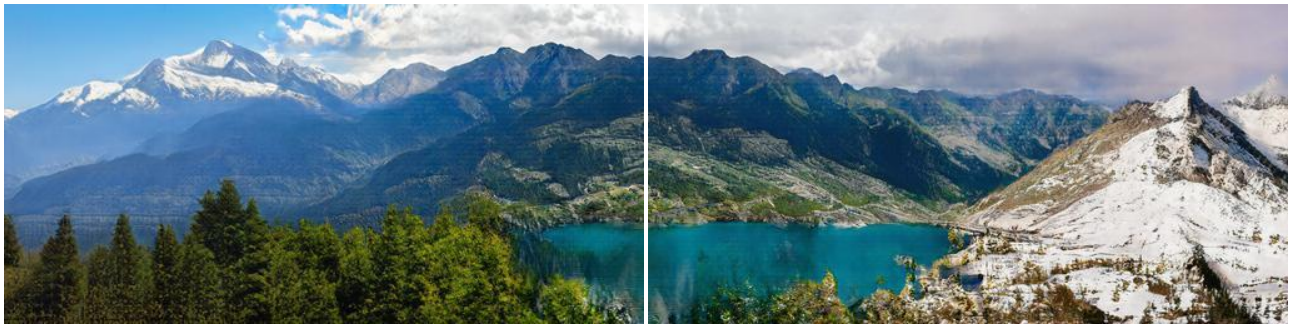
- Takes a pre-trained text-to-image model, Stable Diffusion 1.5
- Fine-tunes it on spectrogram images with text descriptions
- Seeds it with some music loop + noise, conditions reconstruction on text
- Produces audio from spectrograms with Griffin-Lim algorithm
- Commercialized since yesterday



https://arxiv.org/abs/2208.08706

MUSIKA

- Compresses spectrograms with two levels of adversarial autoencoders (from ~86 fps, 769 bands to latent vectors with ~24 fps, 64 dimensions)
- Trains a GAN on latent vectors, conditioned on a latent coordinate system, adopted from infinite panorama generation (https://arxiv.org/abs/2104.06954)



ALIS

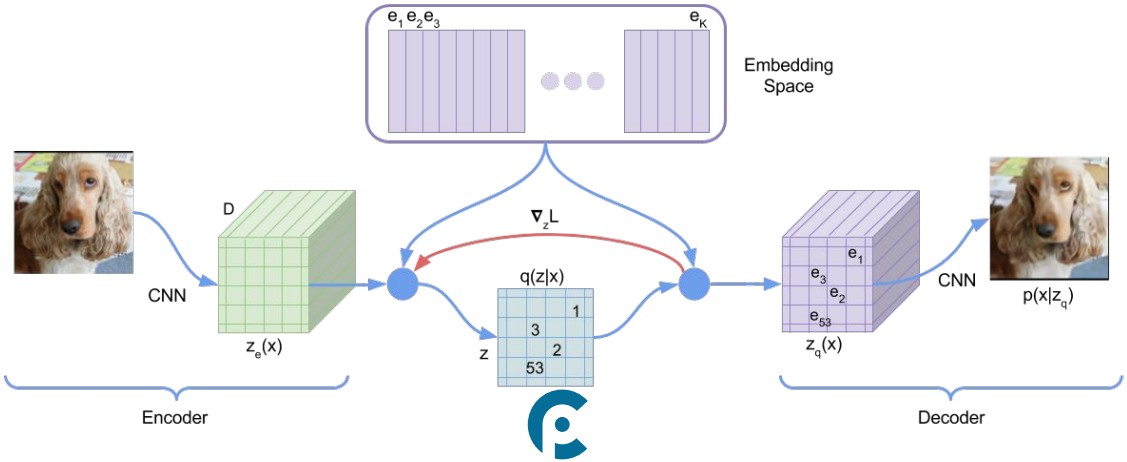
Musika

NEURAL AUDIO CODECS

- **Idea:** Compress audio waveform into a sequence of small discrete numbers (e.g., from 0 to 1023) that allows reconstructing the audio
- **Generation** can then be expressed as a **classification** task: Which number comes next?
- Classification is easier than regression when there are multiple plausible predictions (cf. WaveNet: casts audio sample prediction as classification)
- How do we learn such a compression?

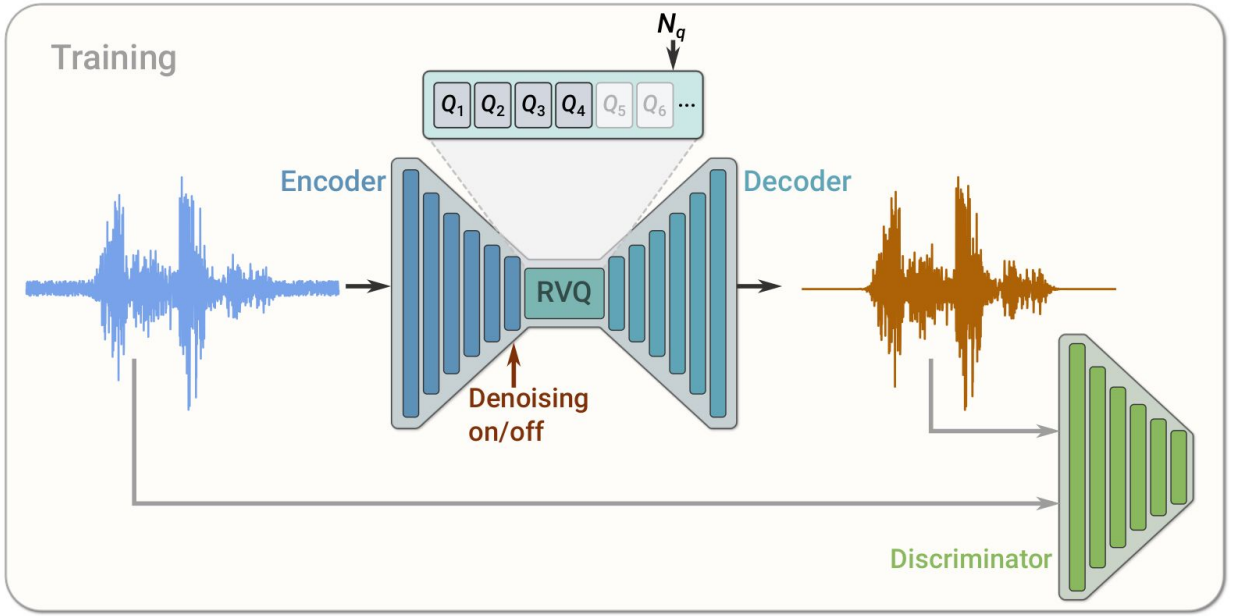
VQ-VAE

- A convolutional auto-encoder with a vector quantization bottleneck
- Forward pass: replace latent vector by closest dictionary entry
- Backward pass: copy gradient over this (non-differentiable) operation
- Loss: (1) minimize reconstruction error, (2) move embeddings towards associated latents, (3) keep latents close to their associated embeddings



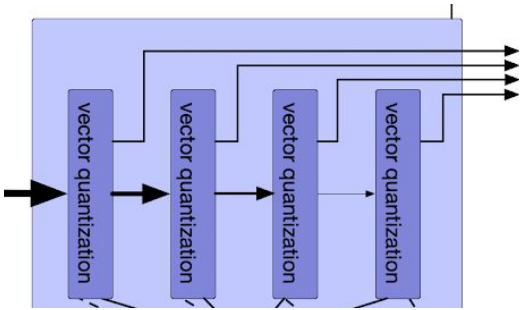
SOUNDSTREAM

- Temporal compression by convolution
- Discretization by **Residual Vector Quantization**
- Trained with reconstruction + adversarial loss



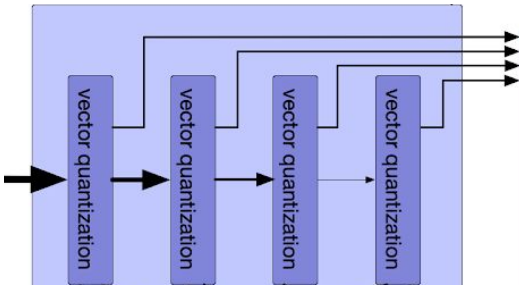
RESIDUAL VECTOR QUANTIZATION

- VQ: Learn codebook, replace latent vector by index of closest codebook entry
- RVQ: Compute difference between latent vector and closest codebook entry (= **residual**), encode this difference with another level of VQ, repeat
- In total, learns *k* codebooks, represents latent vector by *k* discrete numbers
- The first codebook (and the first discrete number) is the most important, the next ones model progressively finer details



RESIDUAL VECTOR QUANTIZATION

- VQ: Learn codebook, replace latent vector by index of closest codebook entry
- RVQ: Compute difference between latent vector and closest codebook entry (= **residual**), encode this difference with another level of VQ, repeat
- In total, learns *k* codebooks, represents latent vector by *k* discrete numbers
- The first codebook (and the first discrete number) is the most important, the next ones model progressively finer details
- **Quantizer dropout:** For each example, sample $n \leq k$, only use the first *n* codebooks. Effectively trains a codec that can run at different bitrates.
- **Training nuisance:** Codebook entries can get stuck and unused, need to be resampled during training.

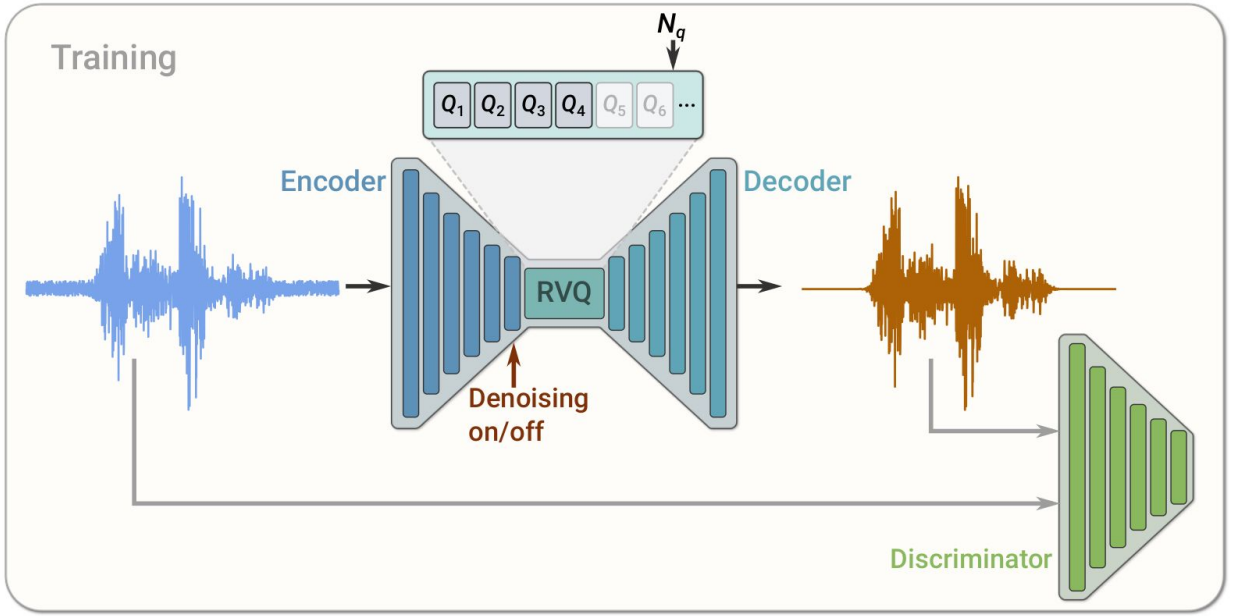


SOUNDSTREAM

- Frame rate: 75 Hz
- Latent size: ?
- RVQ levels: 8
- Codebook size: 1024

→ 75 x 8 x 10 = 6000 bits/sec

- Sadly, training code or trained models not published

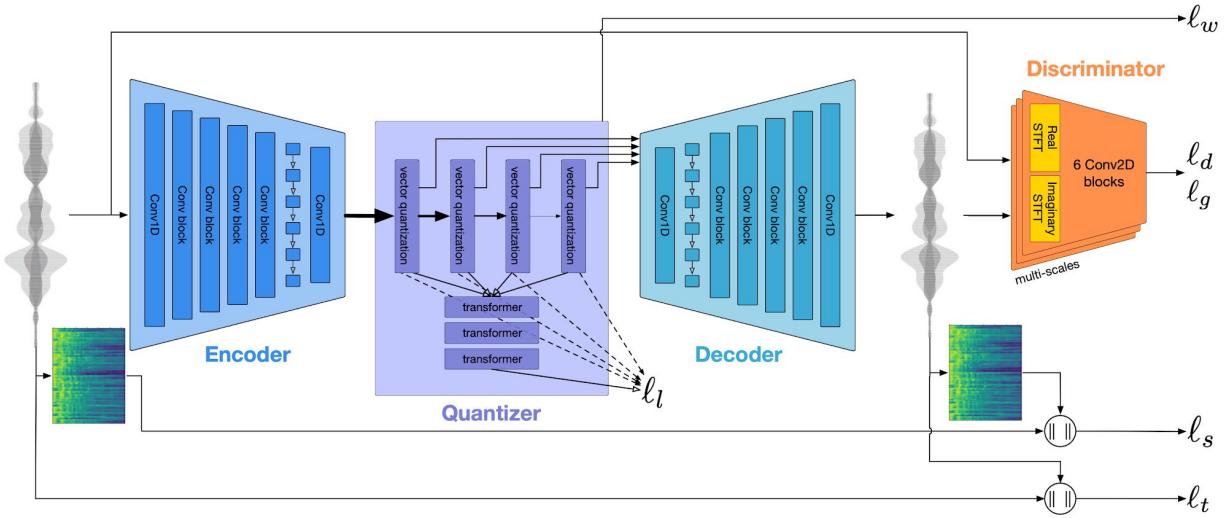


ENCODEC

- Frame rate: 75 Hz
- Latent size: 128
- RVQ levels: 32
- Codebook size: 1024

→ 75 x 32 x 10 = 24000 bits/sec

- Published two pretrained models (24kHz generic, 48kHz music)



DESCRIPT AUDIO CODEC

- Frame rate: ~86 Hz
- Latent size: 1024
- RVQ levels: 9
- Codebook size: 1024

→ ~86 x 9 x 10 = ~7752 bits/sec

- Published pretrained model and training code

- Uses snake activation to improve modeling audio
- Learns each codebook in an individual 32-dimensional L2-normalized space → leads to better use of codebook, no resampling needed
- Applies quantizer dropout only in 50% of cases → leads to much better quality when operating at full bandwidth, only little worse at low bandwidth

input

reconstruction

MORE ON VECTOR QUANTIZATION

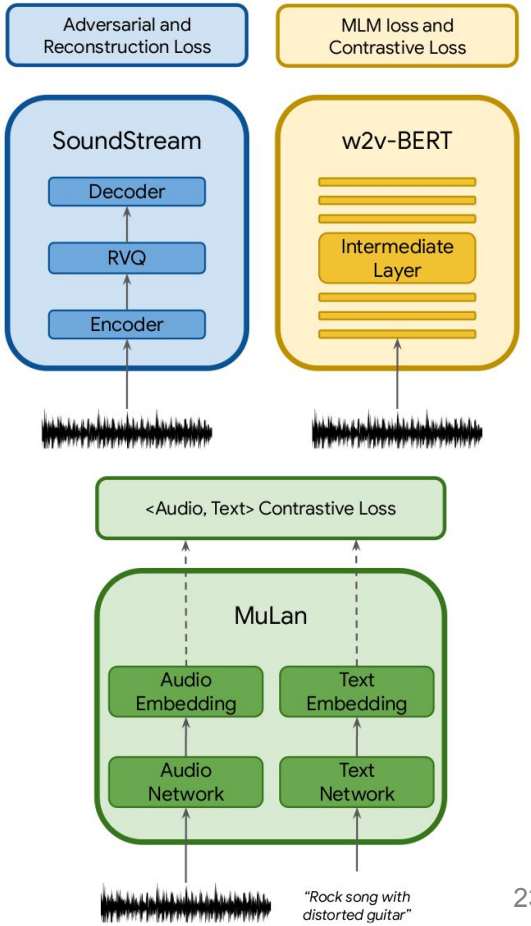
- For a great overview of more techniques and tricks (with descriptions, paper references and PyTorch implementations), see https://github.com/lucidrains/vector-quantize-pytorch

MUSICLM

Meditative song, calming and soothing, with flutes and guitars. The music is slow, with a focus on creating a sense of peace and tranquility.

MUSICLM

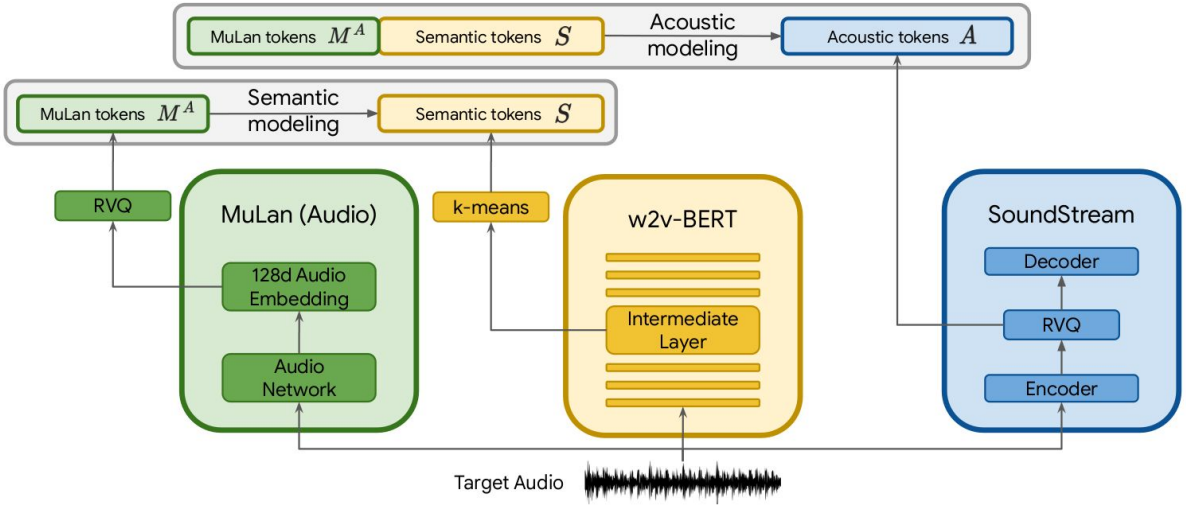
- Pre-trains three building blocks
- **SoundStream:** a version with 50Hz, 12 RVQ levels of size 1024
- **w2v-BERT:** a representation with 25Hz, quantized to 1024 codes, capturing higher-level information
- **MuLan:** a joint music/text embedding trained on 44M music videos with titles, tags, comments, descriptions, playlist titles; audio part pretrained on AudioSet, text part is a pretrained BERT; discretized with 12 RVQ levels of size 1024



MUSICLM

- Trains three autoregressive decoder transformers, only on audio data:

1. Given prefix of 12 MuLan tokens, continue with w2v-BERT tokens
2. Given MuLan and w2v-BERT tokens, continue with 4 SoundStream RVQ levels
3. Given 4 SoundStream RVQ levels, continue with 8 SoundStream RVQ levels

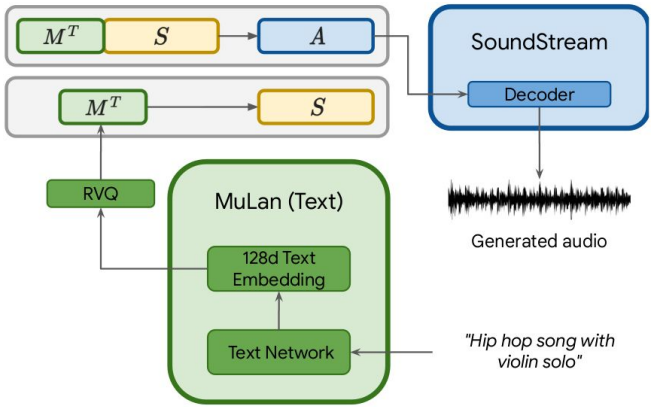


MUSICLM

- Trains three autoregressive decoder transformers, only on audio data:

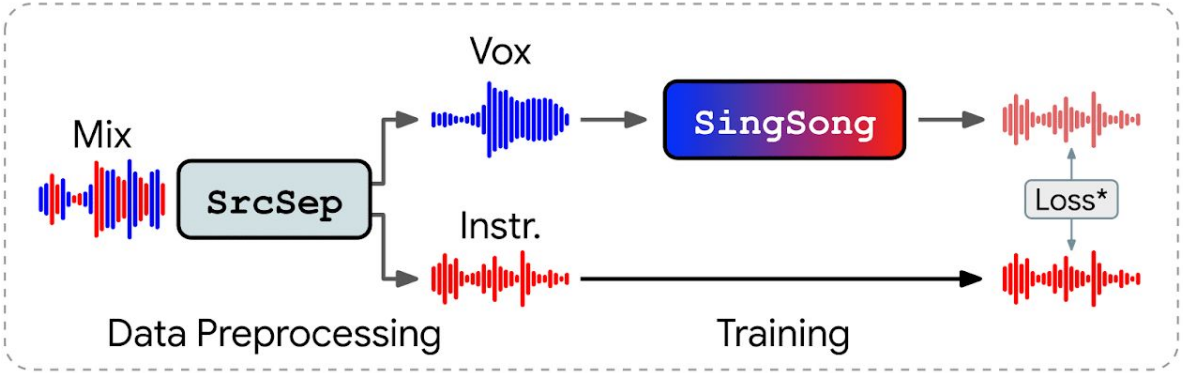
1. Given prefix of 12 MuLan tokens, continue with w2v-BERT tokens
2. Given MuLan and w2v-BERT tokens, continue with 4 SoundStream RVQ levels
3. Given 4 SoundStream RVQ levels, continue with 8 SoundStream RVQ levels

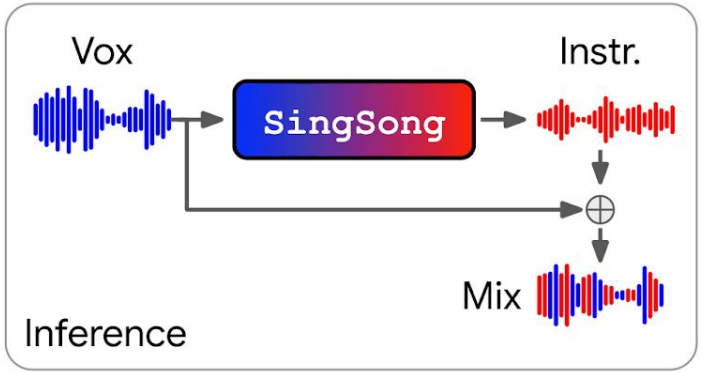
- For inference, compute prefix of 12 MuLan tokens from input text (these tokens are compatible with the audio tokens used for training), run the three models



SINGSONG

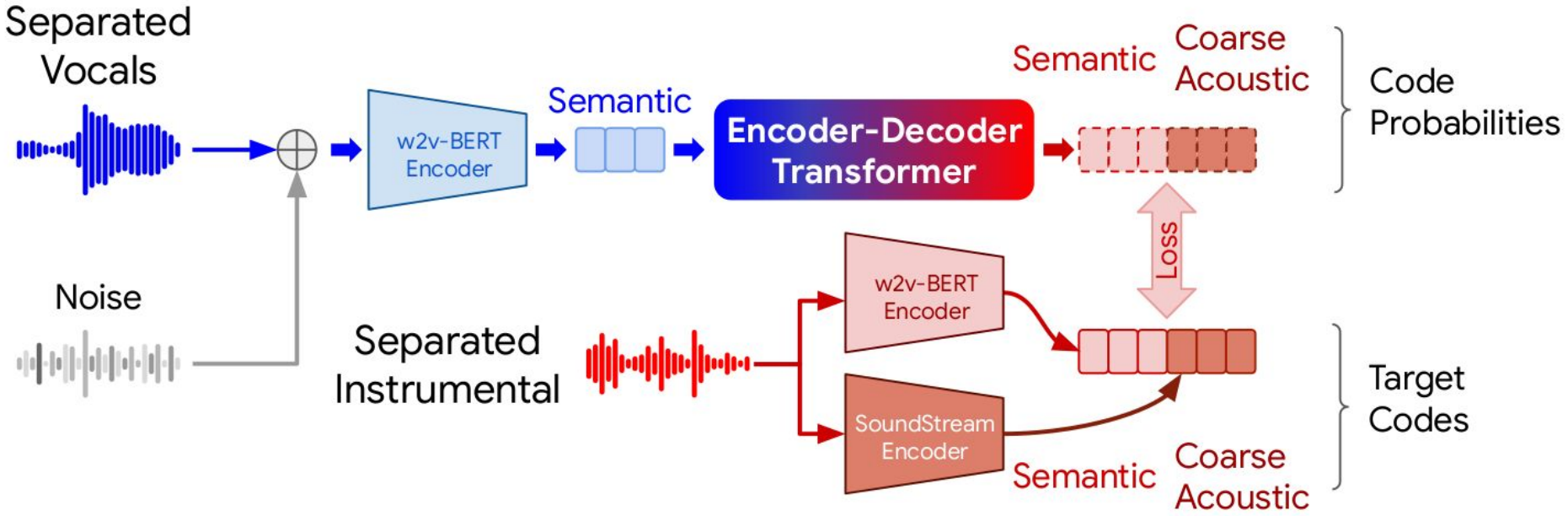
- Takes vocals, generates matching instrumentals





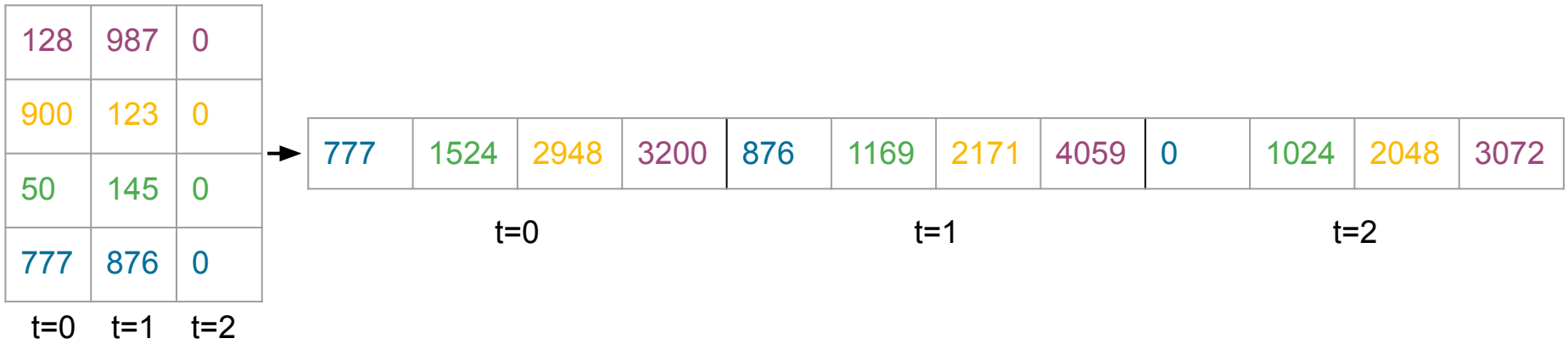
SINGSONG

- Trained on artificially source-separated data, otherwise similar to MusicLM



MUSICLM: HANDLING OF RVQ LEVELS

- SoundStream produces 12 tokens for each time frame
- The first levels are more important than the last ones
- MusicLM uses one model for 4 levels, a second one for the remaining 8
- RVQ levels are interleaved, both for input and targets:



MUSICLM: HANDLING OF RVQ LEVELS

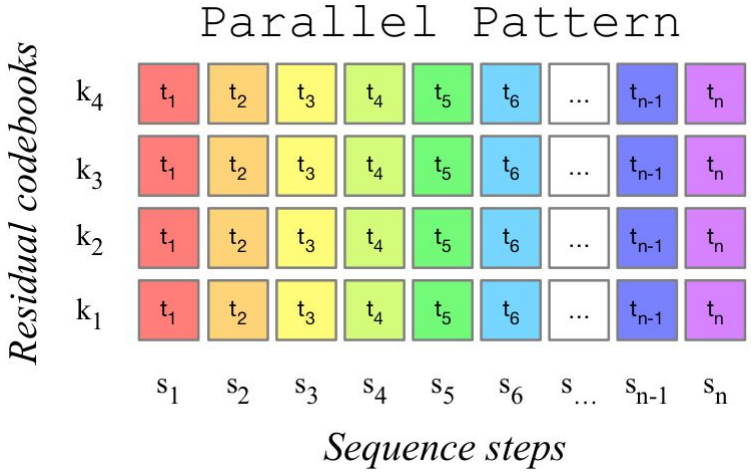
- SoundStream produces 12 tokens for each time frame
- The first levels are more important than the last ones
- MusicLM uses one model for 4 levels, a second one for the remaining 8
- RVQ levels are interleaved, both for input and targets
- This increases the sequence length by the number of RVQ levels!
- Transformer computation requirements scale quadratically with length
- First transformer trained on 30s clips, second one on 10s, third one on 3s

MUSICGEN

Edo25 major g melodies that sound triumphant and cinematic. Leading up to a crescendo that resolves in a 9th harmonic

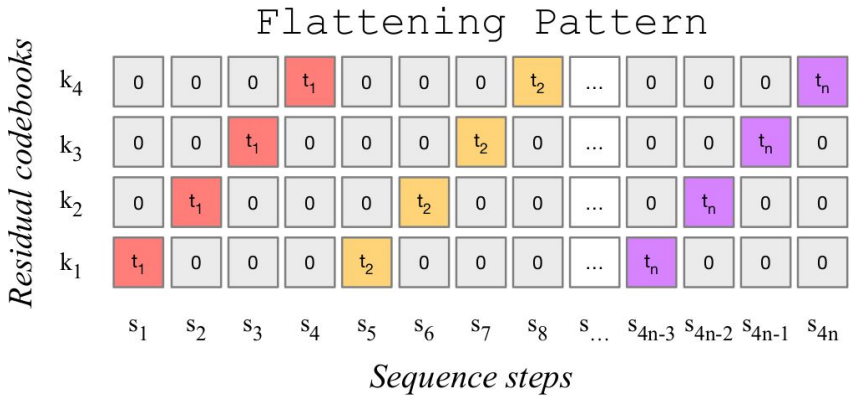
MUSICGEN

- Train a single autoregressive transformer to predict EnCodec tokens from text
- EnCodec version with 50 Hz, 4 RVQ levels of size 1024
- Key insight: RVQ levels do not need to be flattened
- (Too) Simple approach: read 4 levels in parallel, predict 4 levels in parallel
- Input: learn 4 embeddings, add up the embedding of each RVQ level
- Output: have 4 softmaxes à 1024 units
- Problem: assumes higher RVQ levels are conditionally independent of lower RVQ levels given the past
  → fast, but works poorly



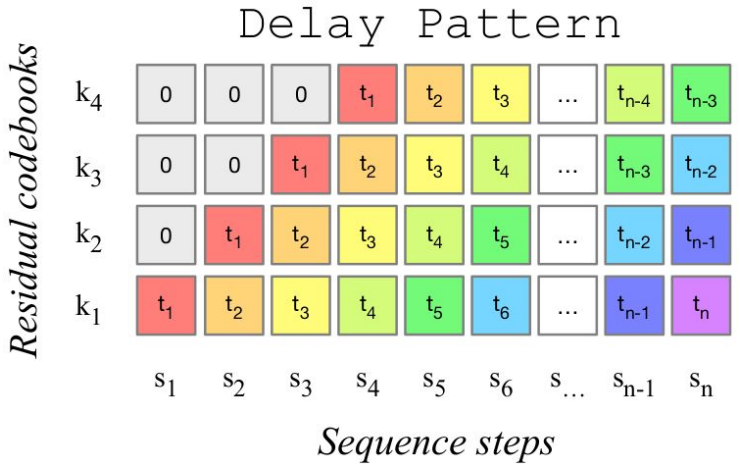
MUSICGEN

- Train a single autoregressive transformer to predict EnCodec tokens from text
- EnCodec version with 50 Hz, 4 RVQ levels of size 1024
- Key insight: RVQ levels do not need to be flattened
- MusicLM's approach: interleave the RVQ levels
- Input: learn 4 embeddings, only use one per time step
- Output: masked softmax of 4096 units
- Problem: sequence is 4 times longer, so requires 16 times the compute



MUSICGEN

- Train a single autoregressive transformer to predict EnCodec tokens from text
- EnCodec version with 50 Hz, 4 RVQ levels of size 1024
- Key insight: RVQ levels do not need to be flattened
- Compromise: predict in parallel, but delayed
- Input: learn 4 embeddings, add up the embedding of each RVQ level, have 4 special tokens for “not present”
- Output: have 4 softmaxes à 1024 units
- Allows higher RVQ levels to depend on lower levels for the same time step
- Works almost as well as flattening



VAMPNET

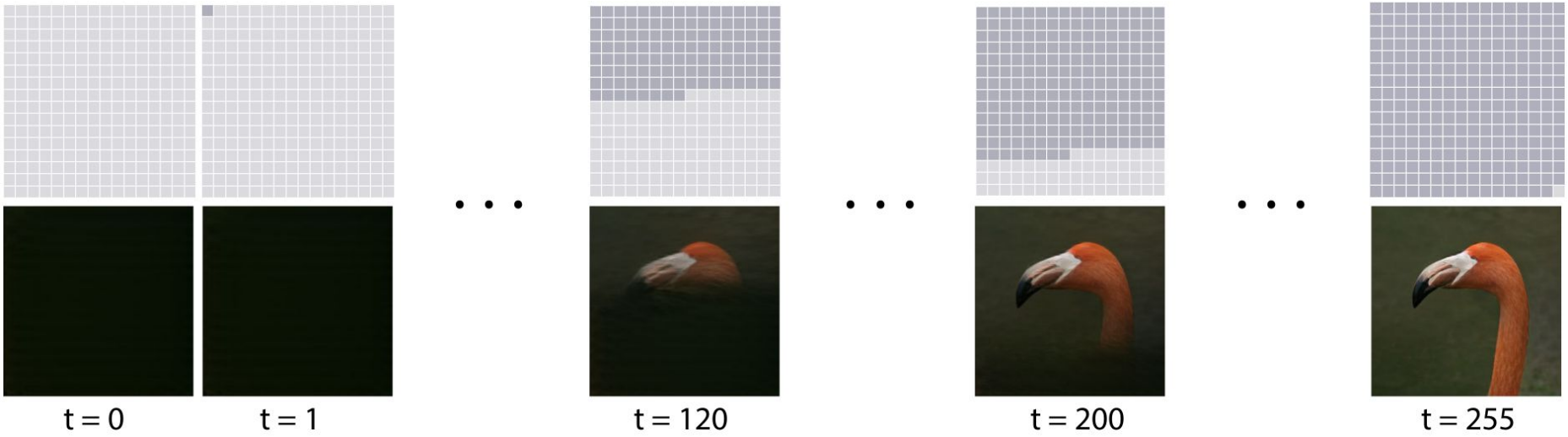
- Trains **MaskGIT**-like transformer on **DAC** tokens to avoid autoregression

VAMPNET

- Trains **MaskGIT**-like transformer on **DAC** tokens to avoid autoregression 🤔

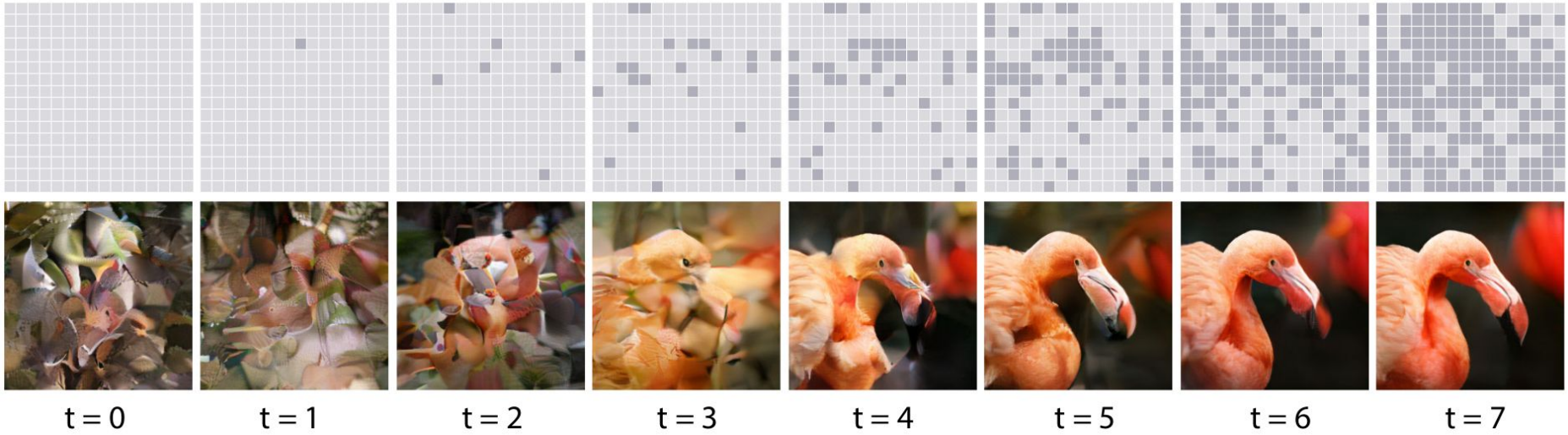
https://arxiv.org/abs/2202.04200

AUTOREGRESSIVE INFERENCE EXAMPLE



https://arxiv.org/abs/2202.04200

MASKGIT INFERENCE



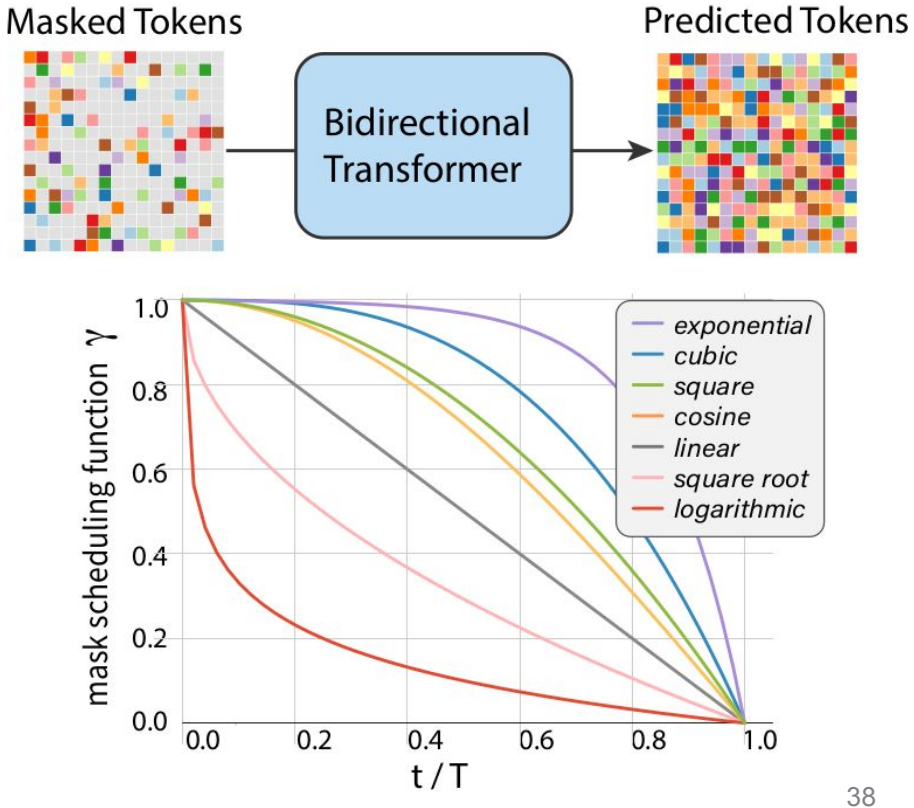
https://arxiv.org/abs/2202.04200

MASKGIT: TRAINING

- Like BERT, trained to predict masked tokens (in images, though)
- Unlike BERT, has a *random amount* of tokens masked, including very few and all tokens
- The random amount is not uniform, but chosen according to a carefully selected scheme:

  $\gamma(r) = \cos(r \cdot \pi / 2)$

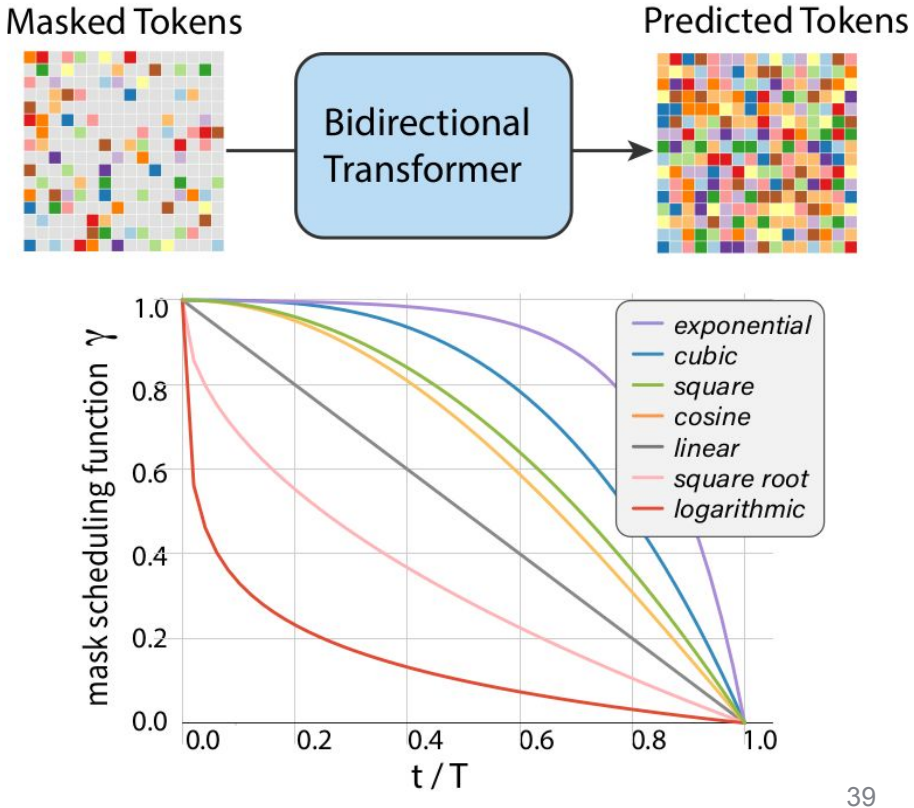
  with r ~ randomly uniform in [0, 1)



https://arxiv.org/abs/2202.04200

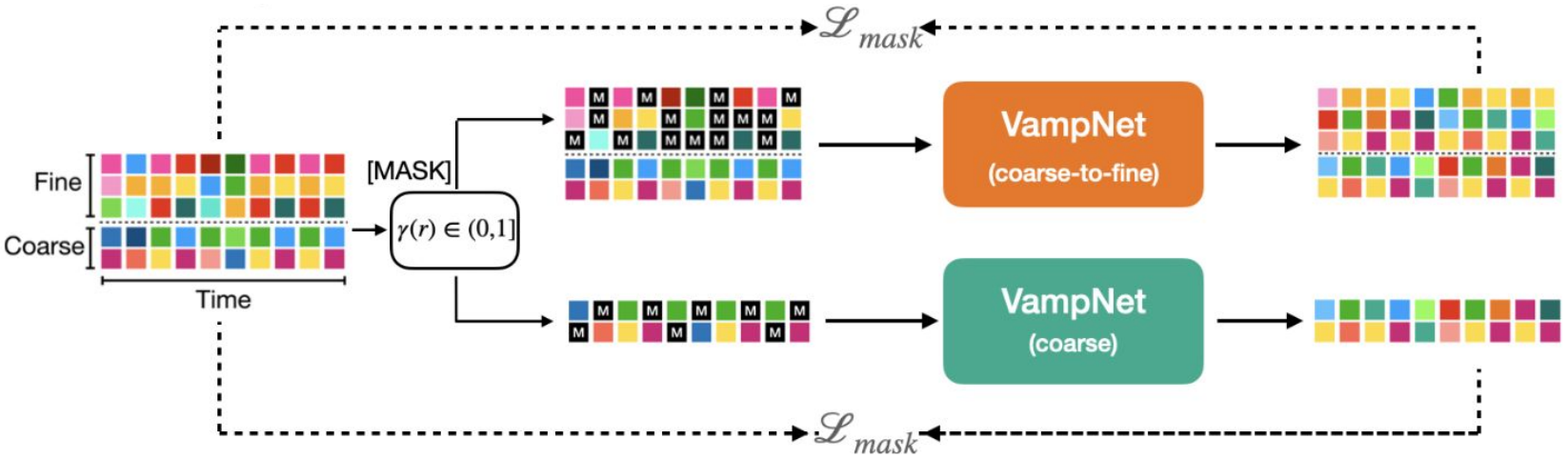
MASKGIT: INFERENCE

1. Start with input of all <MASK>, t=0
2. Predict: we get a softmax over the token dictionary at each position
3. Sample from the softmax for each masked position, remember the token's probability as a confidence
4. Keep unmasked tokens, set their confidence to 1.0
5. Add noise to all confidences
6. t ← t+1; if t < T, mask the least-confident $\gamma(t/T)$ tokens and goto 2



https://arxiv.org/abs/2307.04686

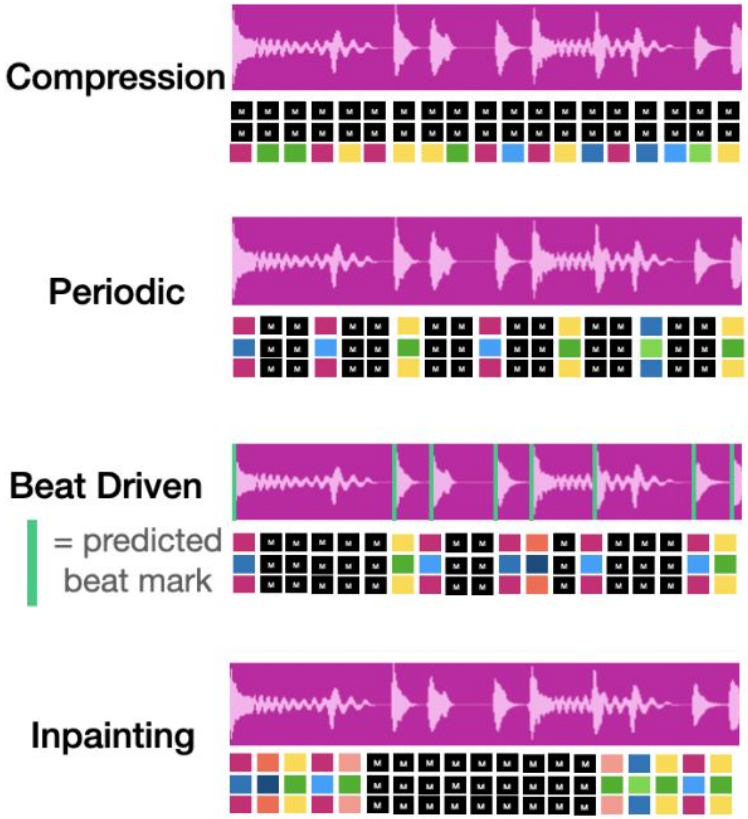
VAMPNET: TRAINING



https://arxiv.org/abs/2307.04686

VAMPNET: INFERENCE

- Same iterative algorithm as MaskGIT
- Starting from completely blank input probably did not work well (?), they tried different structured schemes
- **Compression**: keep only 1 codebook
- **Periodic**: keep every 8th, every 16th, or every 32nd frame
- **Beat Driven**: keep 4 frames around beats (as detected by WaveBeat)
- **Inpainting**: keep first and last 1s or 2s

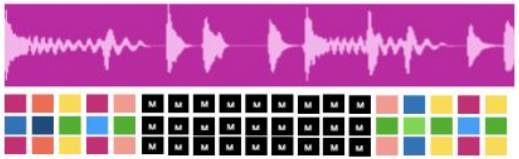


https://arxiv.org/abs/2307.04686

VAMPNET: EXAMPLES

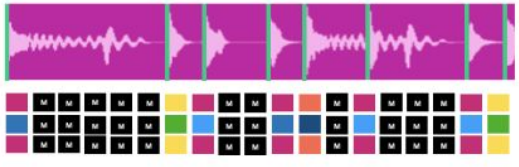
original

Inpainting



prompt result

Beat Driven



prompt result

LoRA

sample

EVALUATION

- **Frechét Audio Distance (FAD):** tests whether 1s snippets of generated audio are statistically indistinguishable from 1s snippets of original audio → measures (short-term) audio quality
- **Classifier KL Divergence (KLD):** tests whether a pretrained AudioSet classifier produces similar predictions for original and generated audio of the same prompt → measures coherence with prompt
- **Mean Opinion Score (MOS):** asks human raters, often via Amazon Turk, with smart filtering of unreliable annotations (CrowdMOS software)

DATA USE

- **MuLan:** 44M tracks with noisy text descriptions
- **MusicLM:** SoundStream and w2v-BERT on Free Music Archive, all three transformers on 5M tracks
- **SingSong:** 1M tracks
- **MusicGen:** 400k tracks, 390k of which is instrument-only stock music
- **VampNet:** 797k tracks

CAVEATS

- Plagued by all problems of image generation
- Biases in training data will result in biases in samples
- Models are trained on copyrighted data → what does it mean for samples?
- Models can reproduce training samples → 1% of MusicLM's first transformer samples approximately match training data given the same text prompt, rises to 10% when also giving 1s of training data prefix
- Music generation threatens human musician's income – not for pop music, but for utility music (Gebrauchsmusik)