LLMs for Audio Applications

FOCI GENAI/LLM USERS GROUP EPISODE #6

MAY 1ST, 2024

Agenda

- Introduction
- Background: Language Modeling
- Audio Tokenization
- Audio Language Modeling
- Application: ASR, TTS, S2S Tasks
- Application: Sound & Music Generation
- Application: Full-Duplex Dialogue Agents
- Discussion

Introduction

Natural language is first acquired in the audio modality beginning in infancy.

Children are typically fluent communicators years before they read or write their first word of text.

Can Large Language Models (LLMs) learn to model language without text?

- **Can LLMs directly understand and generate audio?**

Introduction

Audio enables more natural human-computer interaction by interacting through speech. Applications include:

- Digital Assistants (e.g., Alexa)
- Accessibility Aids
- Customer Service Automation
- Transcription & Translation
- More engaging chat agents
- Smart NPCs for video games & immersive worlds

Immersive SDS for NPC concept:
https://www.youtube.com/watch?v=FzSIJ7d3lt0

Agenda

- Introduction
- Background: Language Modeling
- Audio Tokenization
- Audio Language Modeling
- Application: ASR, TTS, S2S Tasks
- Application: Sound & Music Generation
- Application: Full-Duplex Dialogue Agents
- Discussion

Background: Language Modeling

In text generation, we feed tokens in and predict the next ones **autoregressively.**

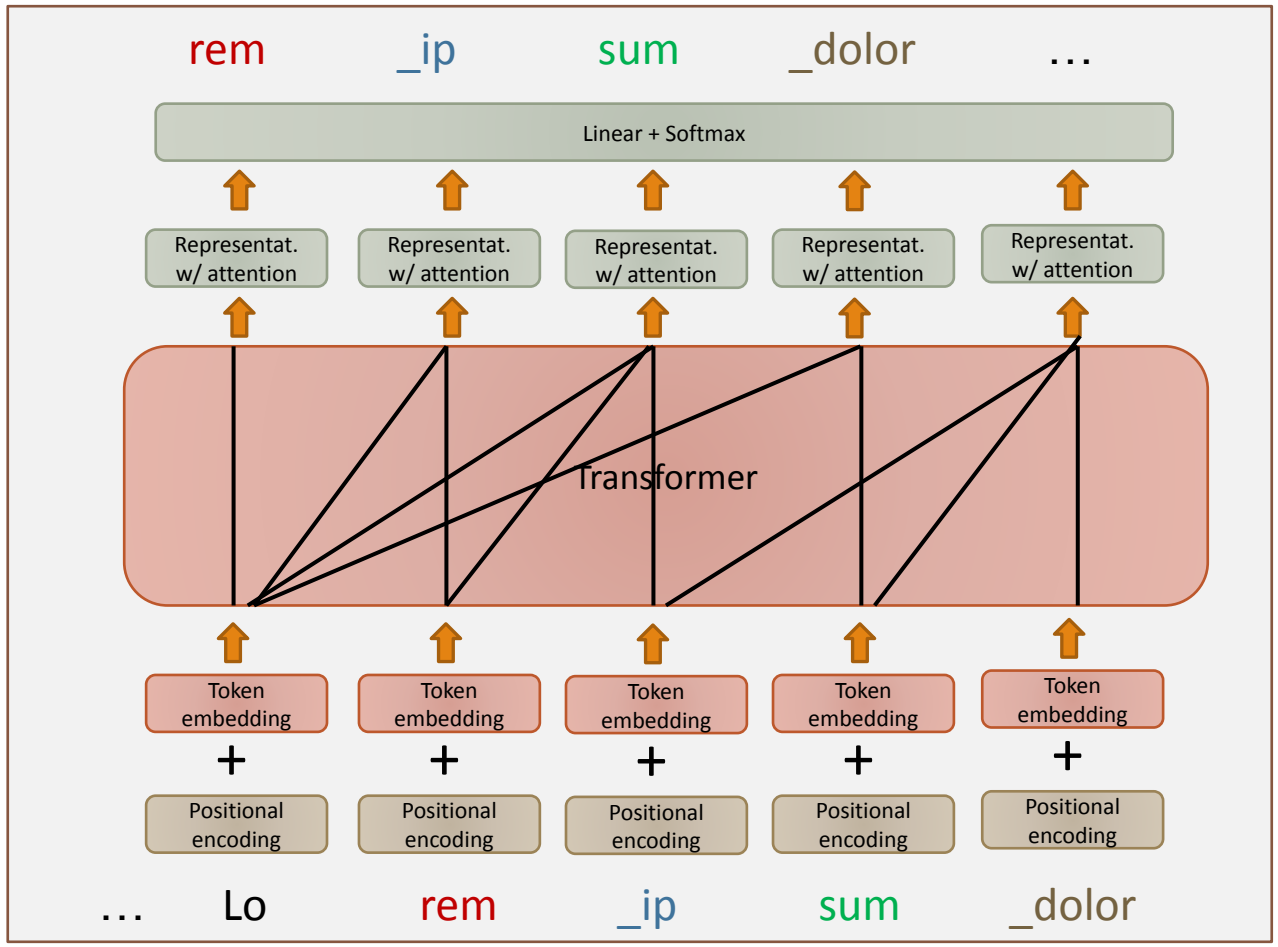
(e.g., GPT 2-4, Llama 1-3, Mixtral, etc.)

Input text is first preprocessed by tokenization into words or subwords:

”Lorem ipsum dolor sit amet”

[”Lo”, ”rem”, ”_ip”, ”sum”, ”_dolor”, ”_sit”, ”_a”, ”met”]

[5643, 6568, 332, 2224, 99, 129, 22931, 2321]



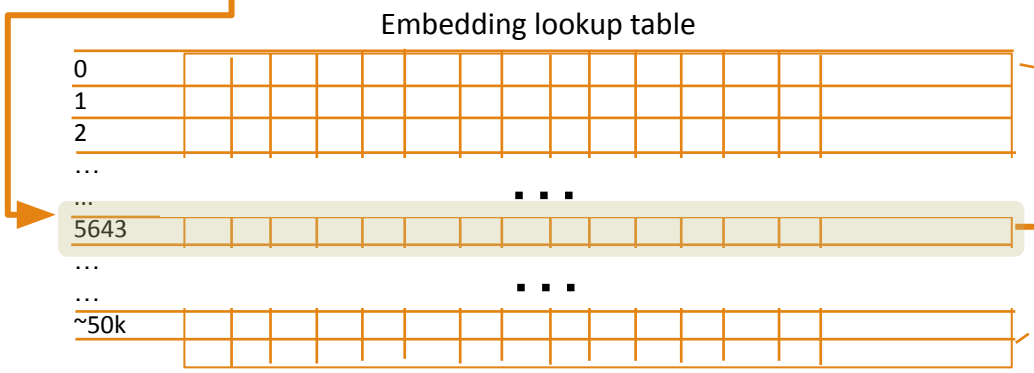
Background: Language Modeling

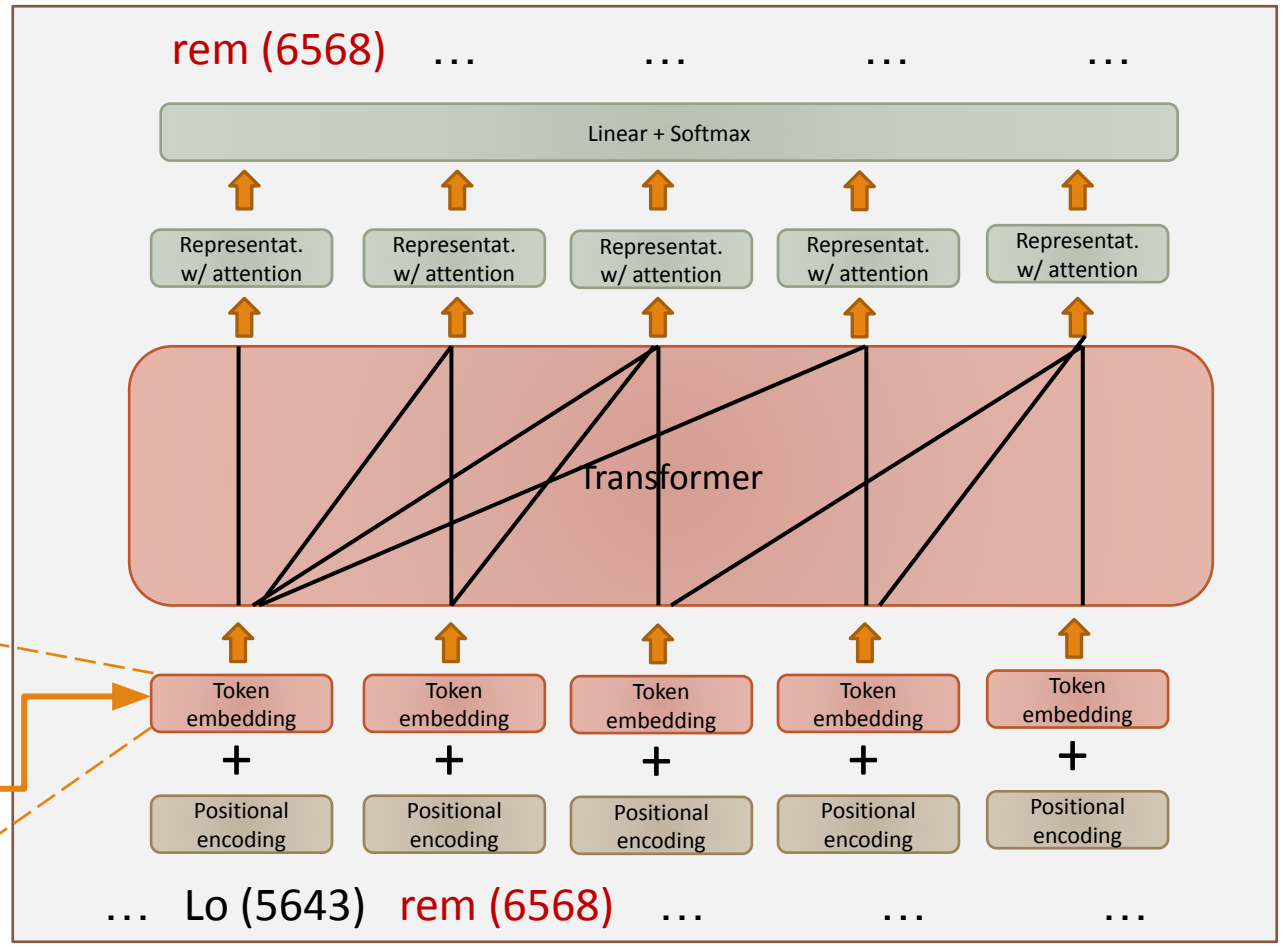
Input text is first preprocessed by tokenization into words or subwords:

”Lorem ipsum dolor sit amet”

[”Lo”, ”rem”, ”_ip”, ”sum”, ”_dolor”, ”_sit”, ”_a”, ”met”]

[5643, 6568, 332, 2224, 99, 129, 22931, 2321]





Agenda

- Introduction
- Background: Language Modeling
- **Audio Tokenization**
- Audio Language Modeling
- Application: ASR, TTS, S2S Tasks
- Application: Sound & Music Generation
- Application: Full-Duplex Dialogue Agents
- Discussion

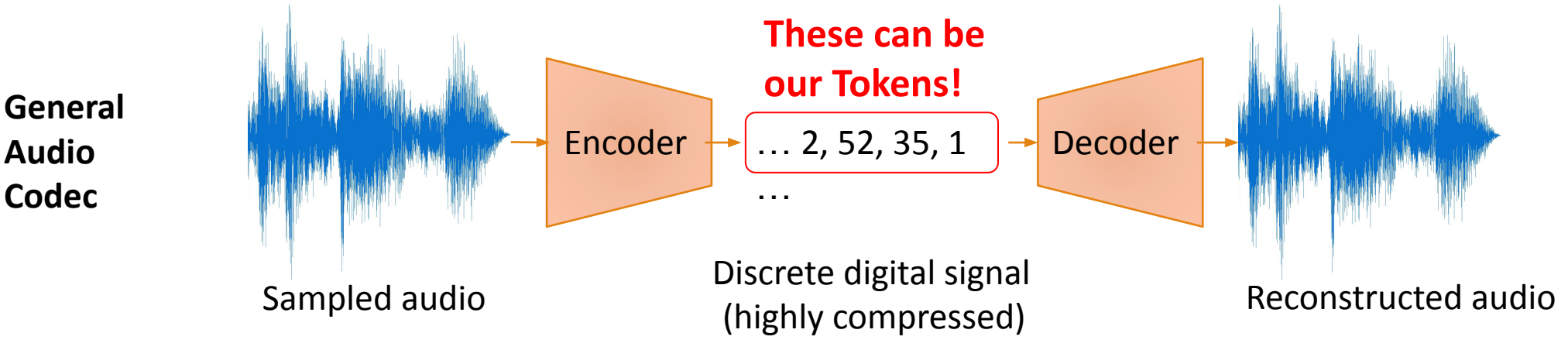
Audio Tokenization

The immediate question:

When building an audio language model, what are the tokens?

- "Acoustic Tokens" need to represent units of sound rather than units of language.

For this, we turn to the Audio Codec (e.g., MP3, AAC, FLAC, WAV, etc…):



Audio Tokenization

The immediate question:

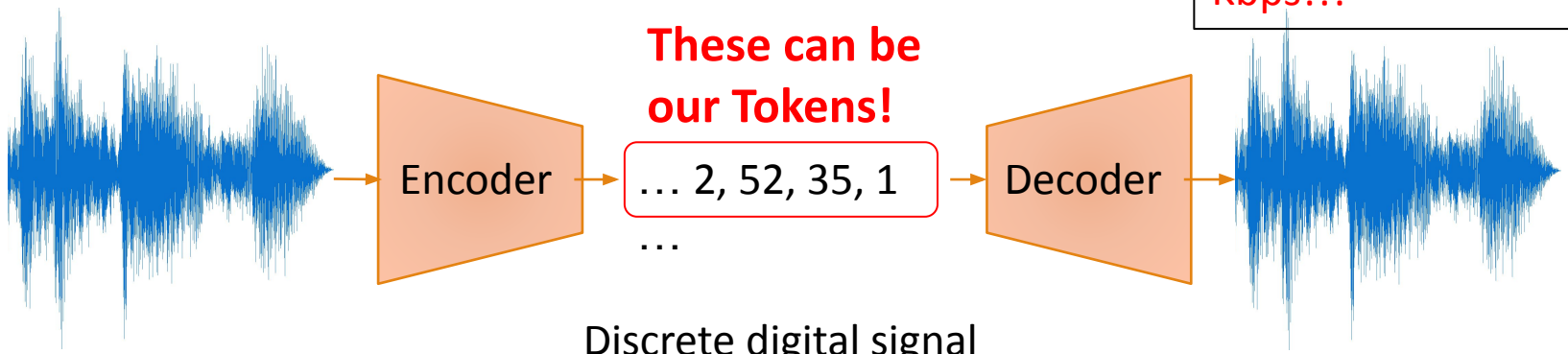
When building an audio language model, what are the tokens?

- "Acoustic Tokens" need to represent units of sound rather than units of language.

For this, we turn to the Audio Codec ~~(e.g., MP3, AAC, FLAC, WAV, etc…)~~:

For 44.1 kHz sampling rate:
MP3, AAC: 96-320 Kbps
Neural Audio Codec: 1.5-24 Kbps!!!

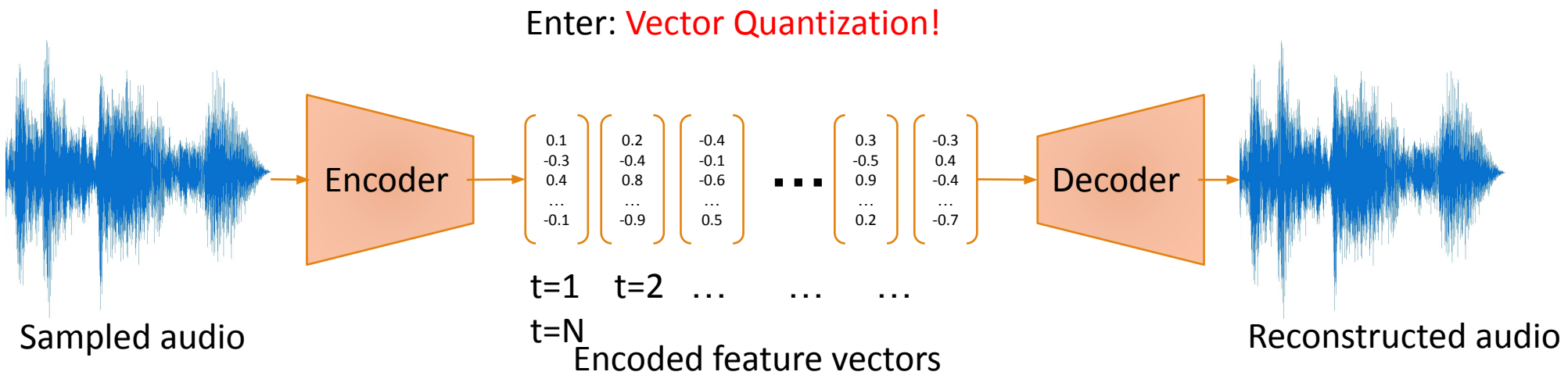
General Audio Codec



Audio Tokenization

A neural audio codec uses a neural encoder to encode a sequence of continuous vector representations of the sampled audio over time, usually at a much lower frequency:

- E.g., raw audio sampled at 24kHz, vectors produced at 75Hz
- But…. this is even more information than the raw audio waveform!

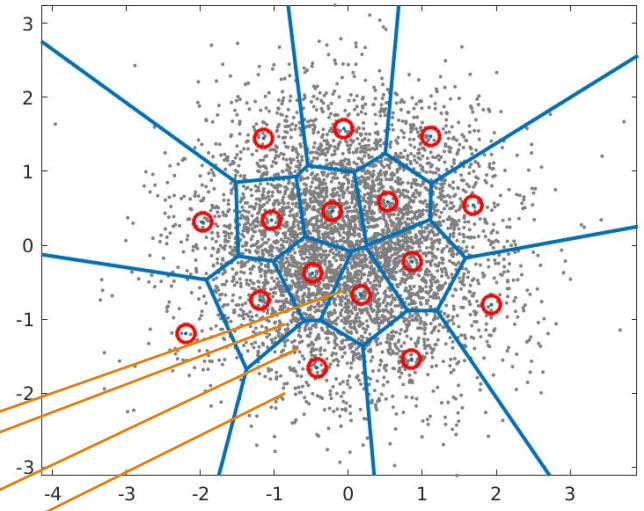


Audio Tokenization

Vector Quantization (VQ) quantizes (discretizes) continuous vectors by mapping each of them to the centroids of their respective Voronoi regions in the space.

It's basically just k-means!*

*K-means usually refers to Lloyd's algorithm, but that is just one of many ways to construct this cluster space. Others include Kohonen's SOM and VQ-VAE.



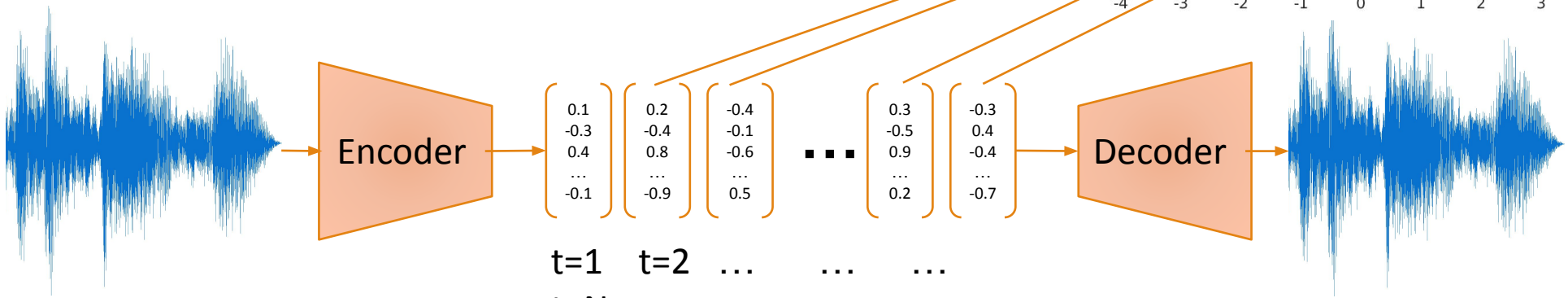


Image source: https://wiki.aalto.fi/pages/viewpage.action?pageId=149883153

Audio Tokenization

The centroid vectors in the space are called Codewords!

Together they create an indexable Codebook!

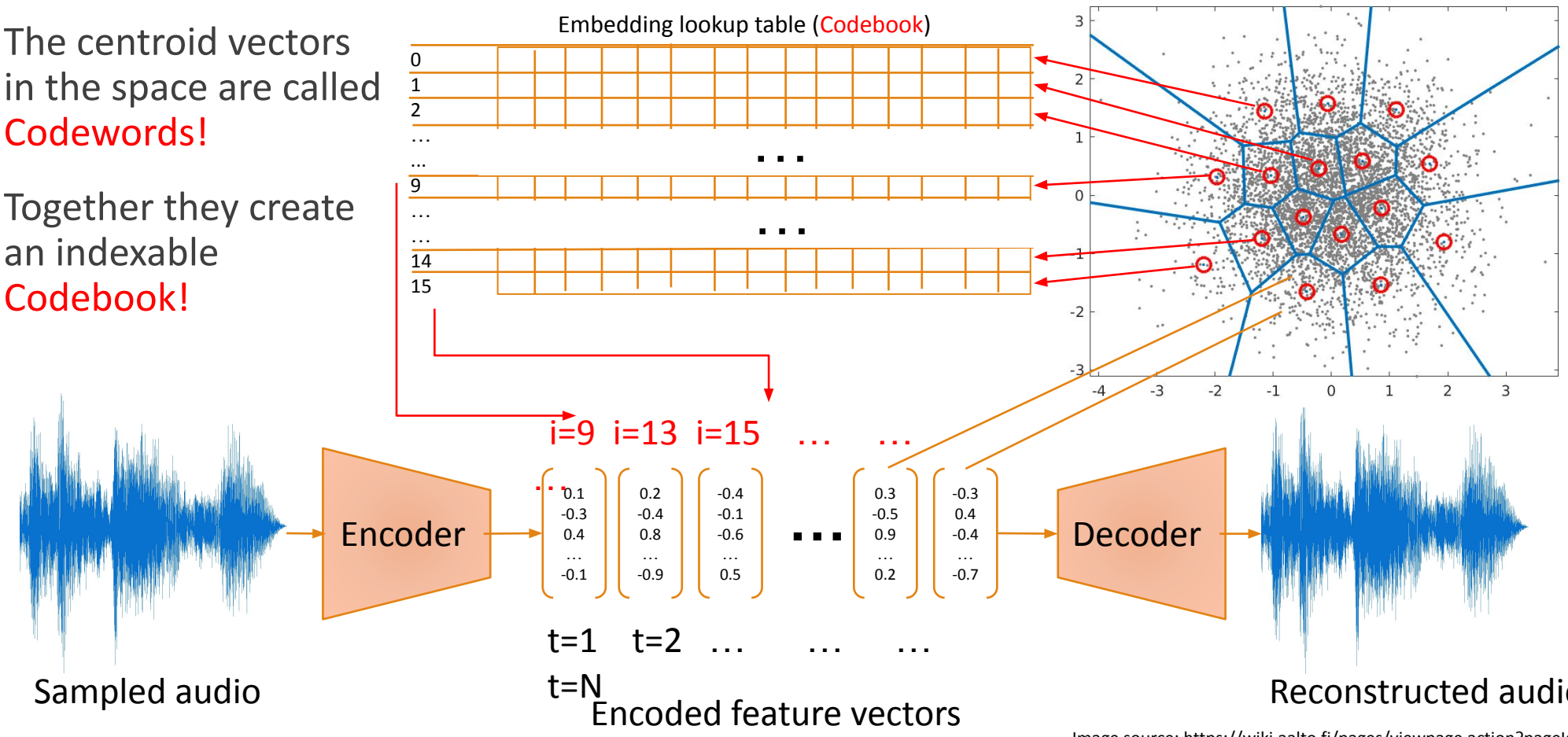
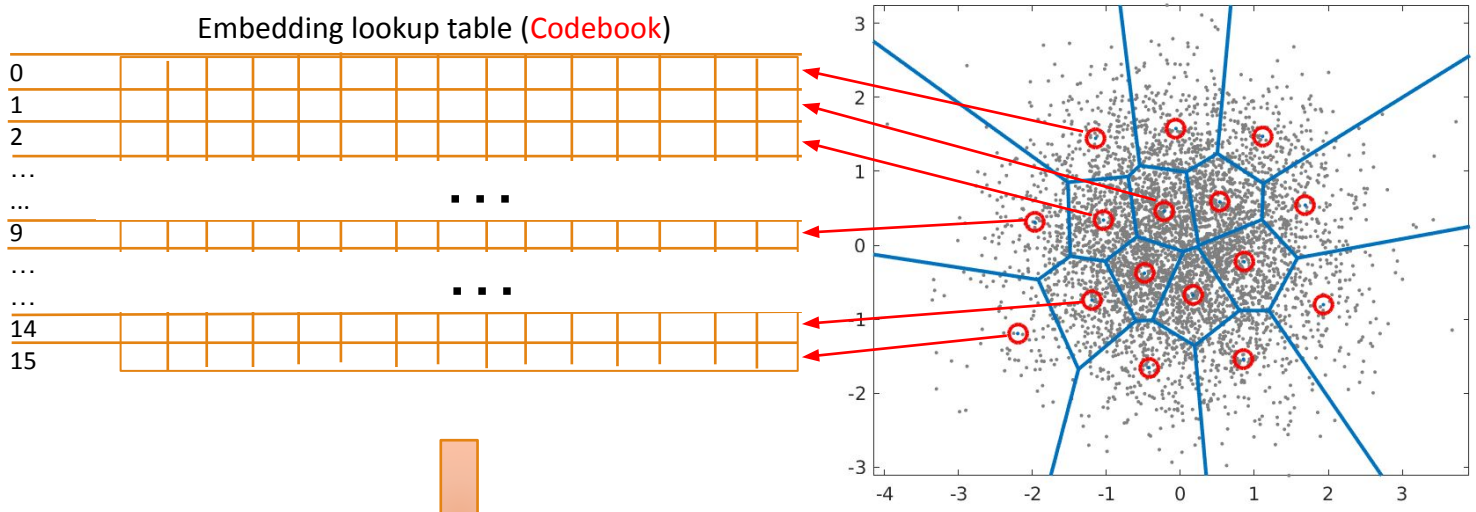


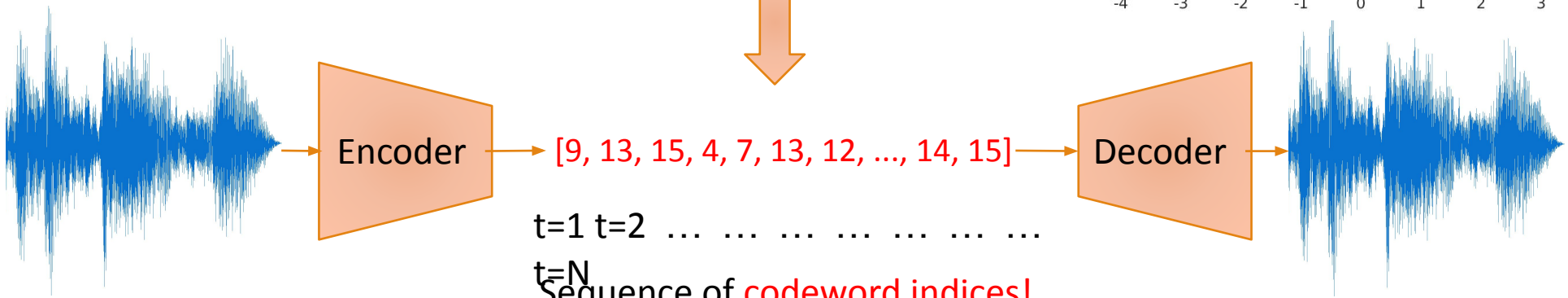
Image source: https://wiki.aalto.fi/pages/viewpage.action?pageId=149883153

Audio Tokenization

The centroid vectors in the space are called Codewords!

Together they create an indexable Codebook!





Sampled audio

Sequence of codeword indices!
(discrete & highly compressed)

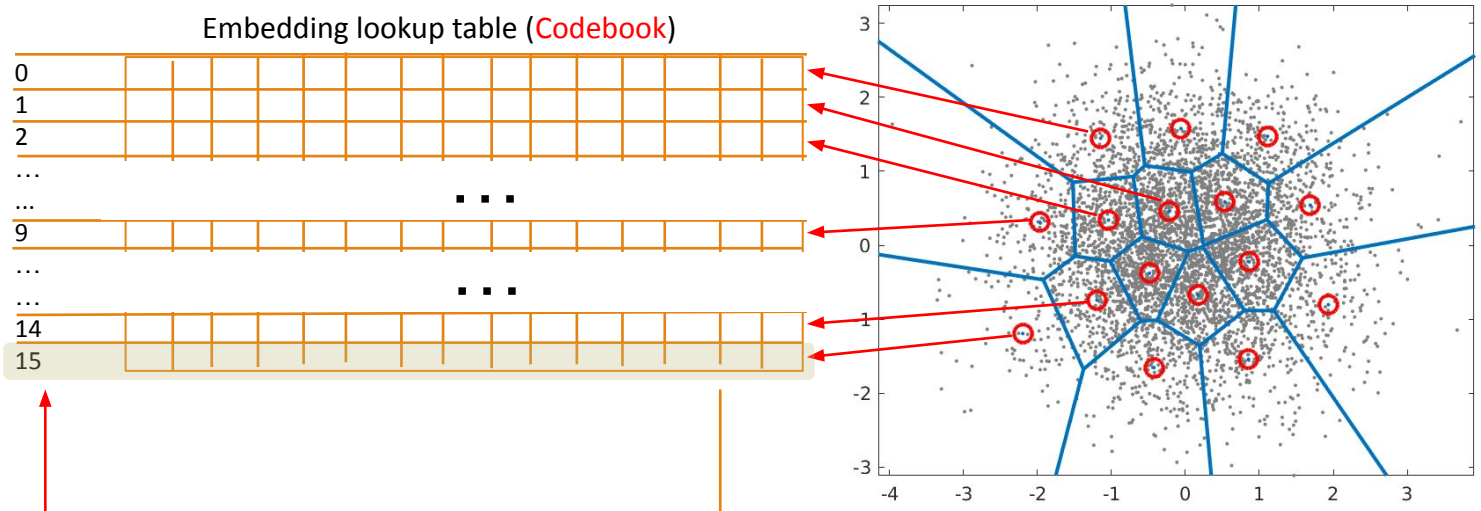
Reconstructed audio

Image source: https://wiki.aalto.fi/pages/viewpage.action?pageId=149883153

Audio Tokenization

Text NLP Analogies:

- Codeword Index: Token Id!
- Codeword: Token embedding!
- Codebook: Token embedding matrix! (lookup table)



The decoder can simply look up each centroid (codeword) vector by its index as a good approximation of the original input vector that it represents.

[9, 13, 15, 4, …, 15]

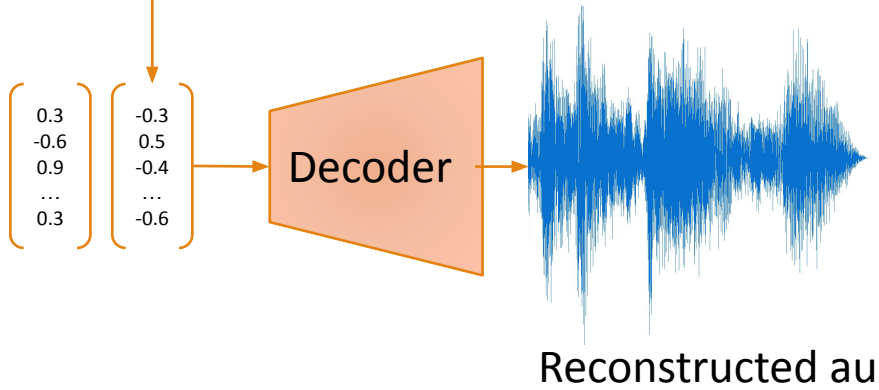
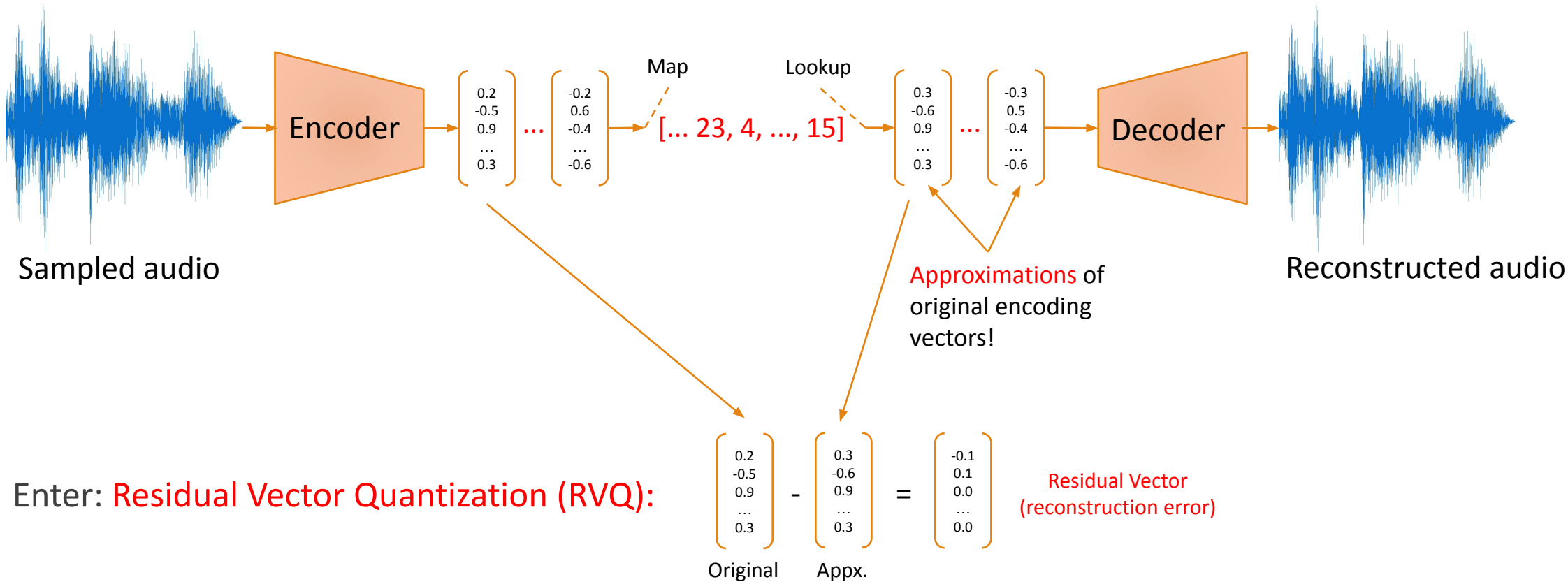


Image source: https://wiki.aalto.fi/pages/viewpage.action?pageId=149883153

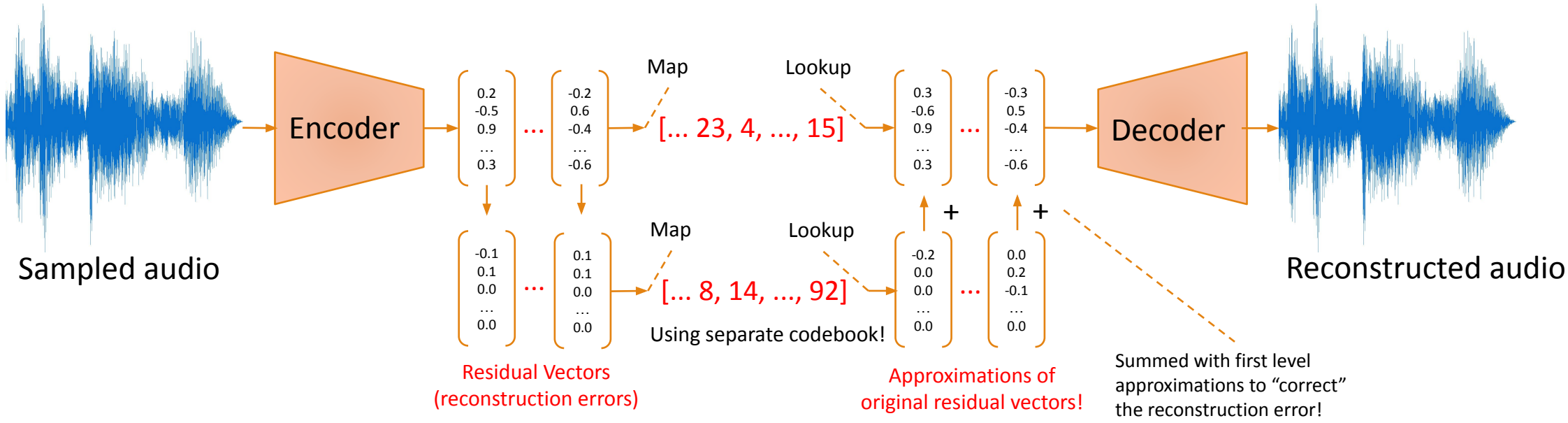
Audio Tokenization

However - the reconstruction will be lossy:



Audio Tokenization

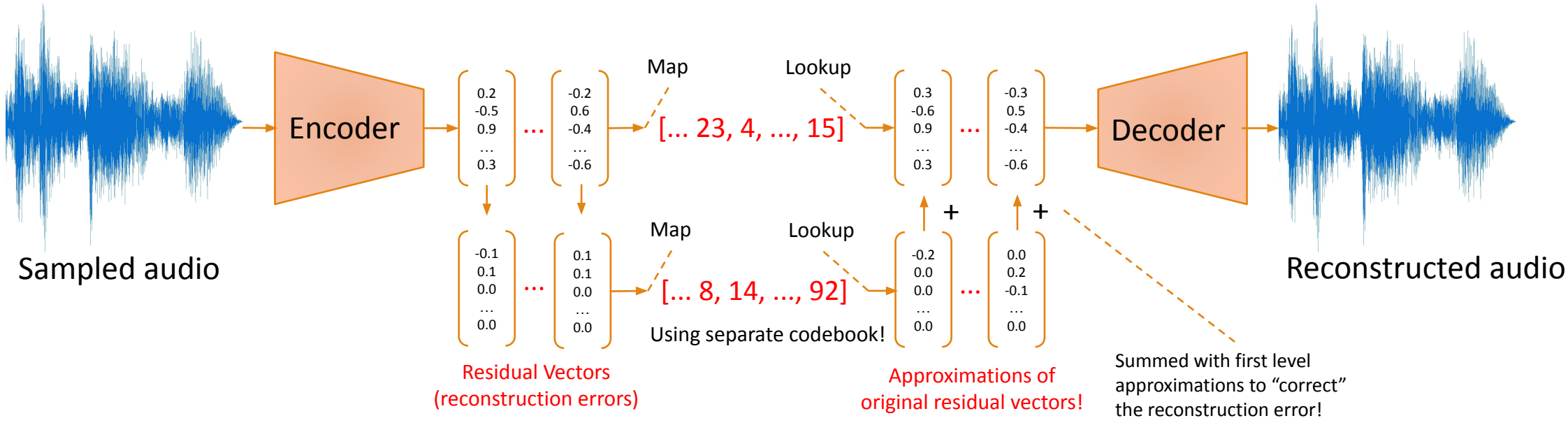
However - the reconstruction will be lossy:



Enter: Residual Vector Quantization (RVQ):

Audio Tokenization

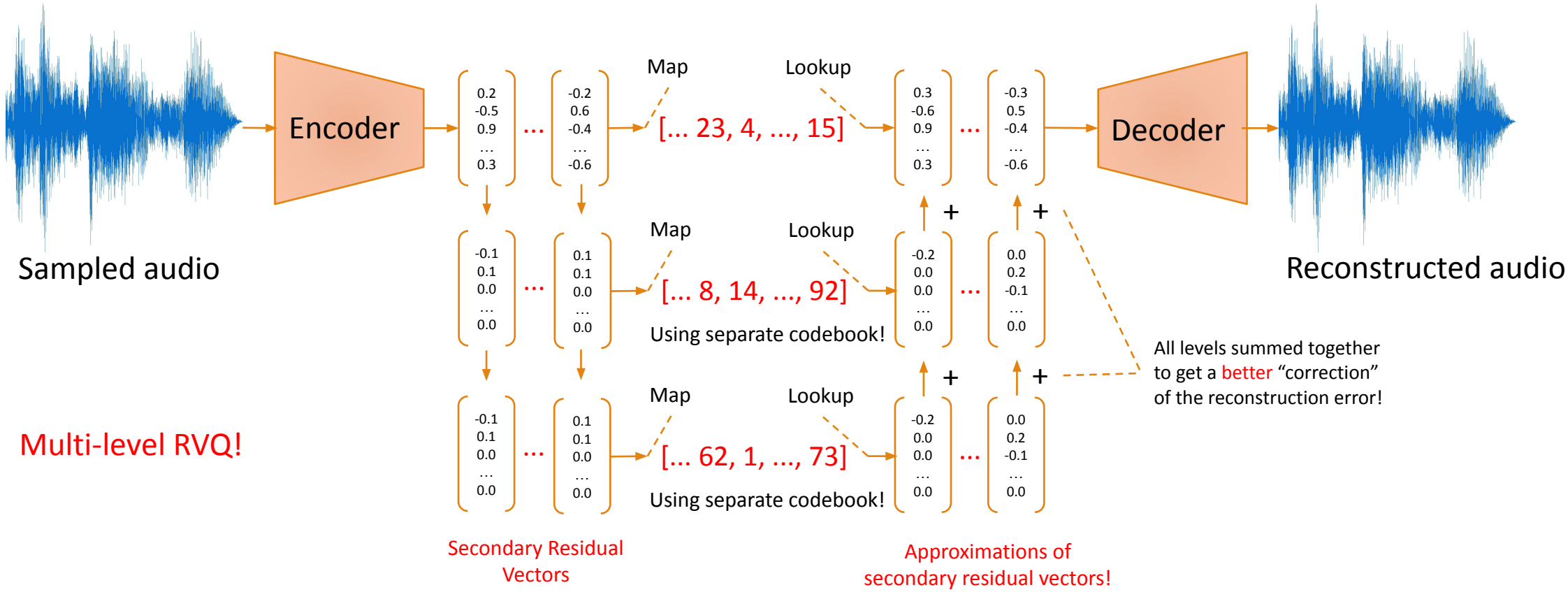
However - the reconstruction will be lossy:



But… won’t the residual approximations have their own reconstruction error? No problem!

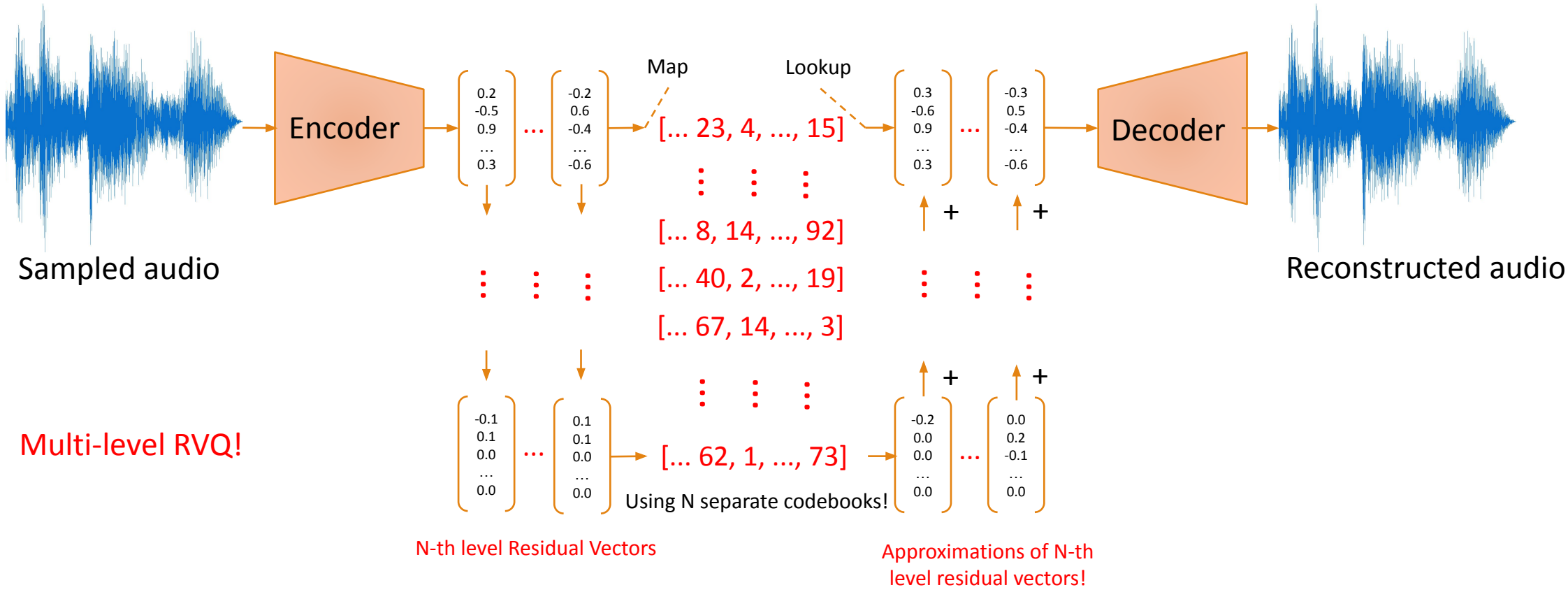
Audio Tokenization

However - the reconstruction will be lossy:



Audio Tokenization

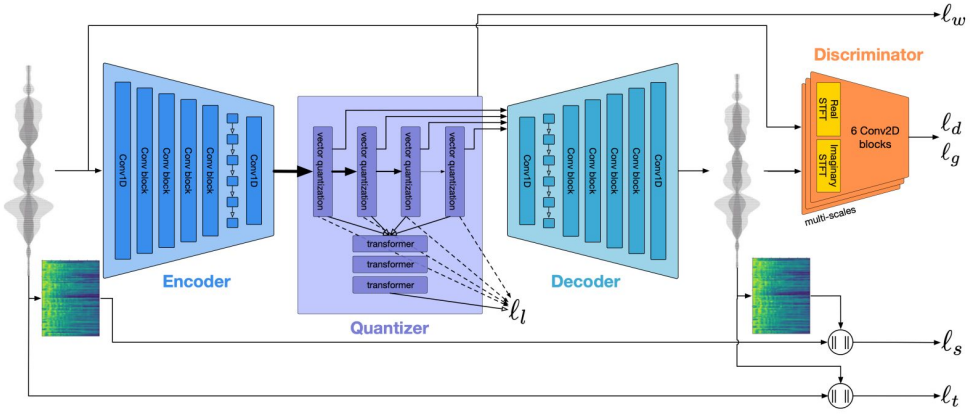
This process can be repeated N times to get progressively better reconstruction:



Audio Tokenization

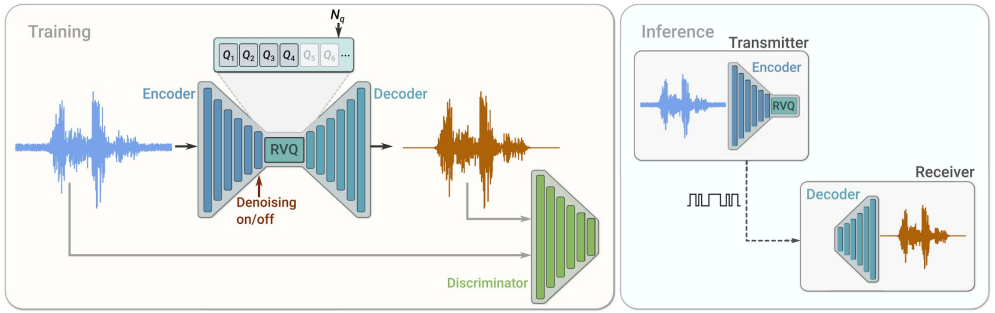
We now have powerful, general Neural Audio Codecs for discrete representation of audio!

EnCodec (Defossez et al., 2022)

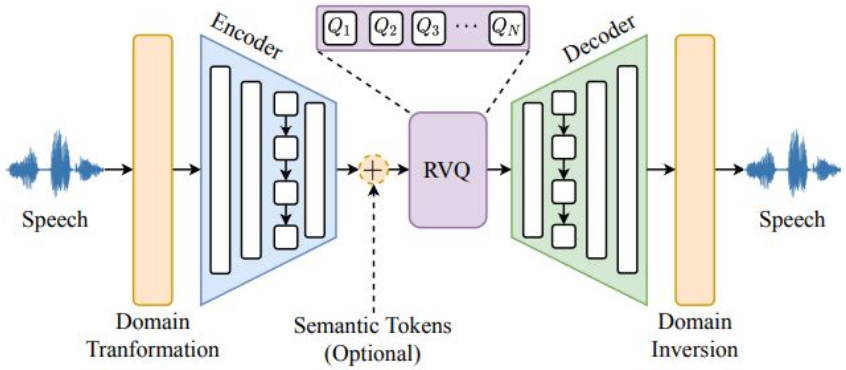


And several others!

SoundStream (Zeghidour et al., 2021)



FunCodec (Du et al., 2023)



By the way:

Vector Quantization is not just for audio: **it can be used anywhere there is a vector space!**

- E.g., VQ-VAE (Van Den Oord et al., 2017) uses VQ to construct a discrete image embedding space
  - **Here, the codebook is learned by gradient descent jointly with the rest of the model.**
  - **This goes on to be a major component in DALL·E!**

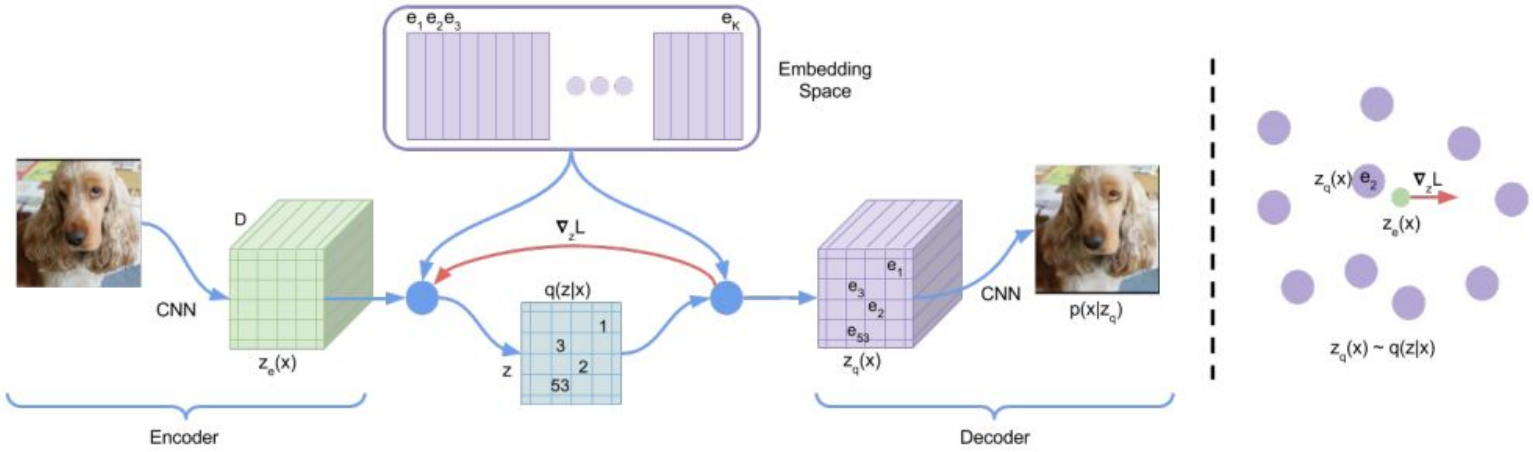


Figure 1: Left: A figure describing the VQ-VAE. Right: Visualisation of the embedding space. The output of the encoder $z(x)$ is mapped to the nearest point e_2. The gradient $\nabla_z L$ (in red) will push the encoder to change its output, which could alter the configuration in the next forward pass.

Agenda

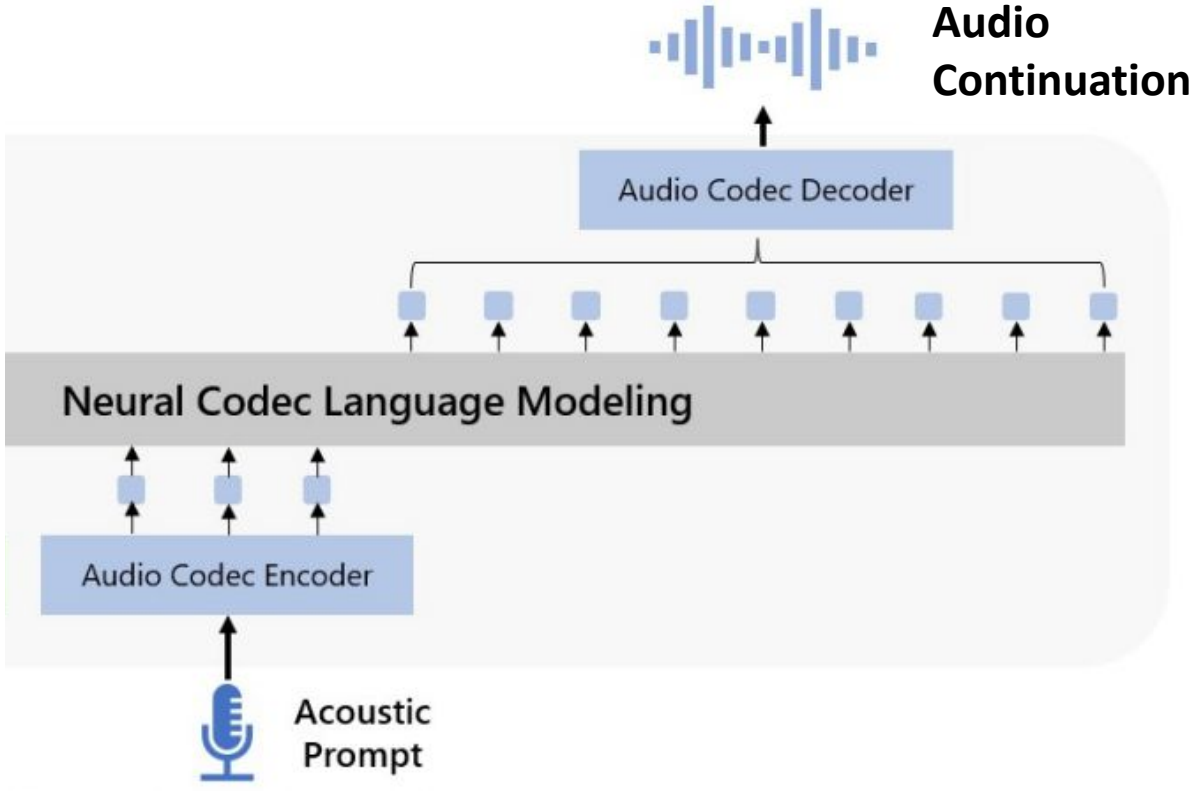
- Introduction
- Background: Language Modeling
- Audio Tokenization
- **Audio Language Modeling**
- Application: ASR, TTS, S2S Tasks
- Application: Sound & Music Generation
- Application: Full-Duplex Dialogue Agents
- Discussion

Audio Language Modeling

Now that we can tokenize audio into a discrete “vocabulary”:

Proceed exactly as we do with text language modeling!

Yes, the concept is that simple.

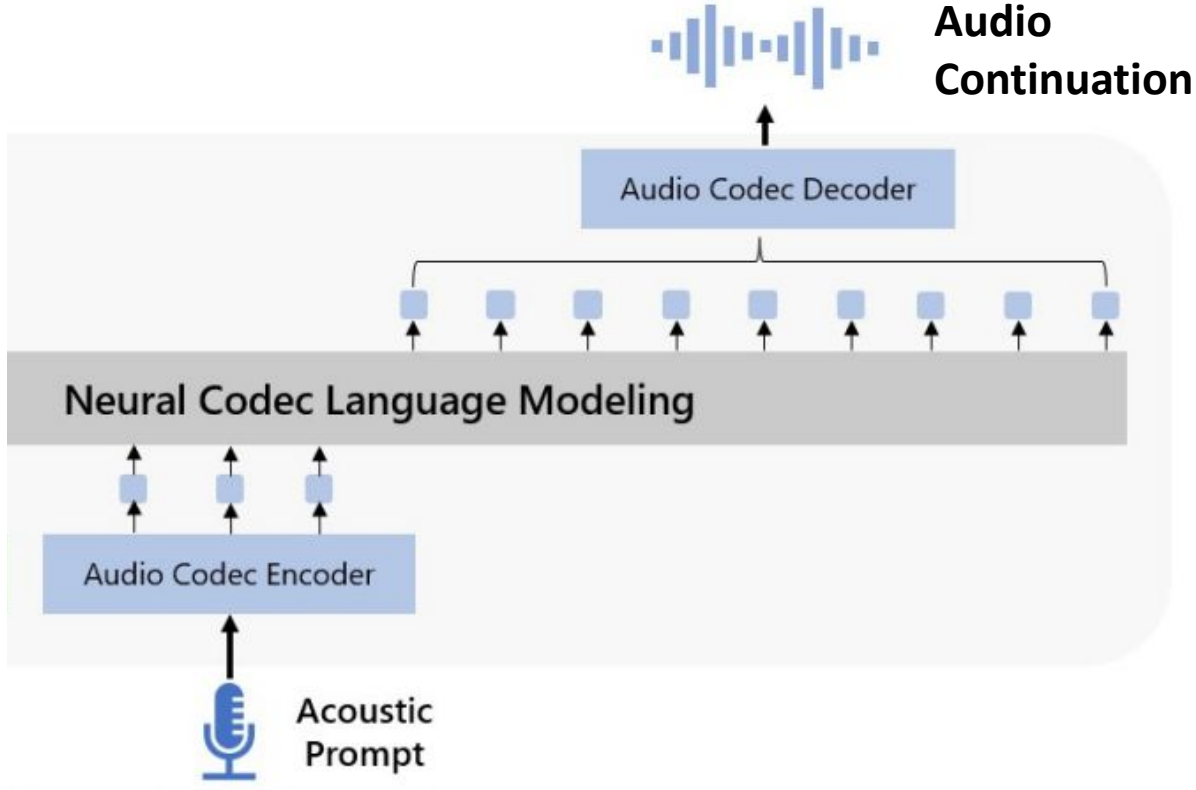


Audio Language Modeling

Yes, the concept is that simple…

… but there arise some practical challenges:

1. Increased sequence length

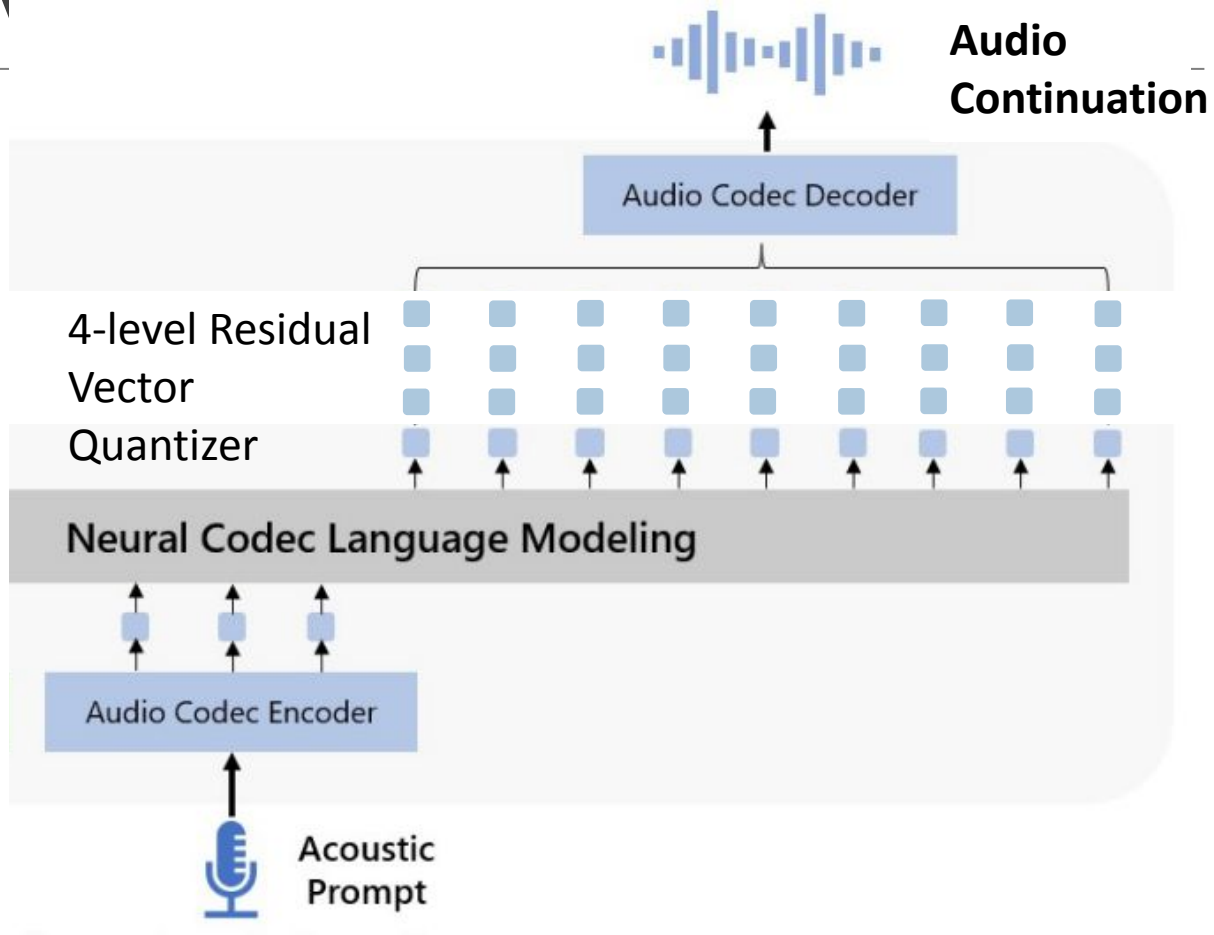


Audio Language Modeling

Yes, the concept is that simple…

… but there arise some practical challenges:

1. Increased sequence length
2. Dealing with the multi-level RVQ
   - Predict multiple tokens per timestep?
     - Non-standard implementation incompatible with LLM ecosystem
     - Poor compatibility with existing sampling methods*
     - Lose causal bias between levels
   - Flatten the codebook and stretch each timestep into N (e.g., 4 for a 4-level RVQ)?
     - Exacerbates issue #1
     - Slower inference
     - Slower to train
     - Requires more GPU memory



Audio Language Modeling

Yes, the concept is that simple…

… but there arise some practical challenges:

1. Increased sequence length
2. Dealing with the multi-level RVQ

These methods are formalized into common “codebook interleaving patterns” (Copet et al., 2023)

All of these patterns can be used successfully with varying results. The “**Delay**” pattern is popular in very recent work (MusicGen, VoiceCraft, Parler-TTS).

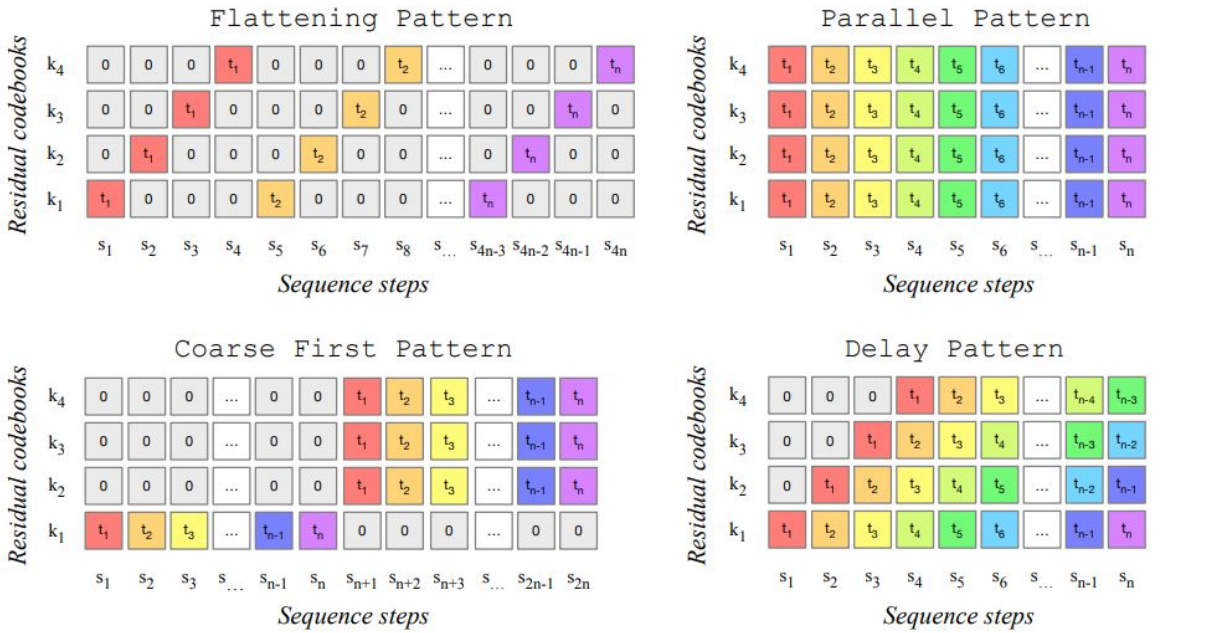


Figure 1: Codebook interleaving patterns presented in Section 2.2. Each time step $t_1, t_2, \ldots, t_n$ is composed of 4 quantized values (corresponding to $k_1, \ldots, k_4$). When doing autoregressive modelling, we can flatten or interleave them in various ways, resulting in a new sequence with 4 parallel streams and steps $s_1, s_2, \ldots, s_m$. The total number of sequence steps S depends on the pattern and original number of steps T. 0 is a special token indicating empty positions in the pattern.

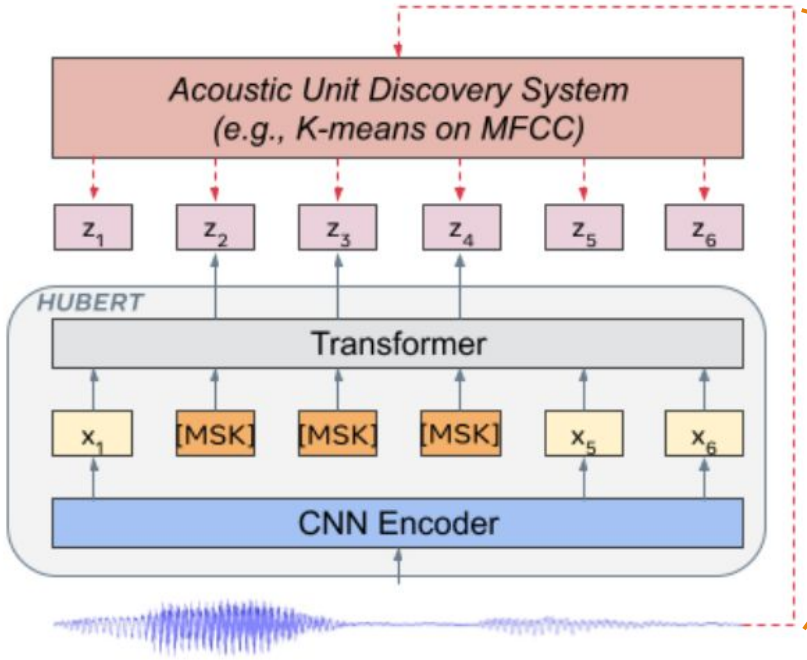
Audio Language Modeling

Early examples of Audio Language Models… (2021 – early 2023)

Audio Language Modeling

Unit = “Codeword”
S2u = “Speech-to-Unit”
uLM = “Unit Language Model”
u2S = “Unit-to-Speech”

Proposed by Lakhotia et al. (2021) in “On Generative Spoken Language Modeling from Raw Audio” (TACL)



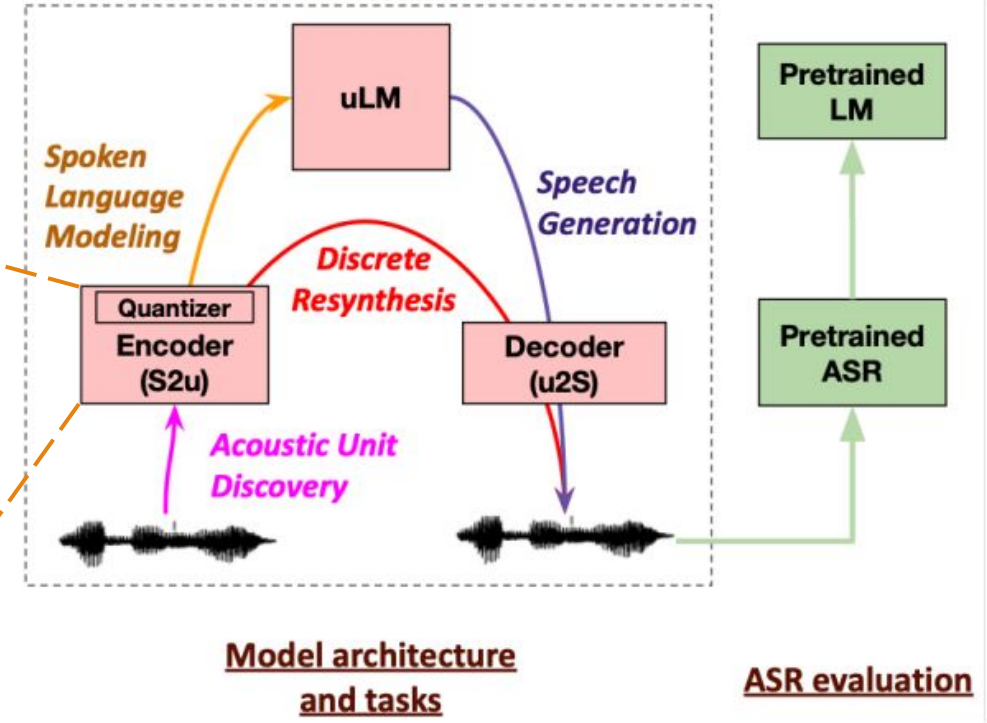


Figure 1: **Setup of the baseline model architecture, tasks and metrics.**

Audio Language Modeling

Unit = “Codeword”
S2u = “Speech-to-Unit”
uLM = “Unit Language Model”
u2S = “Unit-to-Speech”

Proposed by Lakhotia et al. (2021) in “On Generative Spoken Language Modeling from Raw Audio” (TACL)

Standard decoder-only LLM (e.g. GPT-3, etc.)

Layers: 12
Attn Heads: 16
d_model: 1024
d_ffn: 4096
Context size: 3072

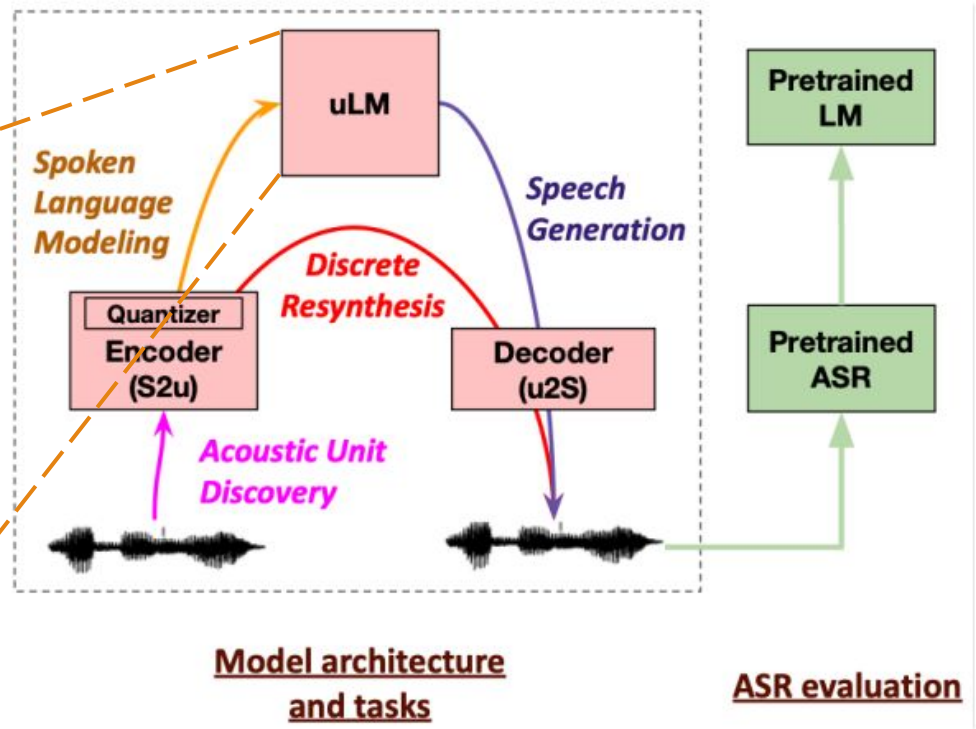


Figure 1: **Setup of the baseline model architecture, tasks and metrics.**

Audio Language Modeling

Unit = “Codeword”
S2u = “Speech-to-Unit”
uLM = “Unit Language Model”
u2S = “Unit-to-Speech”

Proposed by Lakhotia et al. (2021) in “On Generative Spoken Language Modeling from Raw Audio” (TACL)

Modified Tacotron2 (Shen et al., 2018)

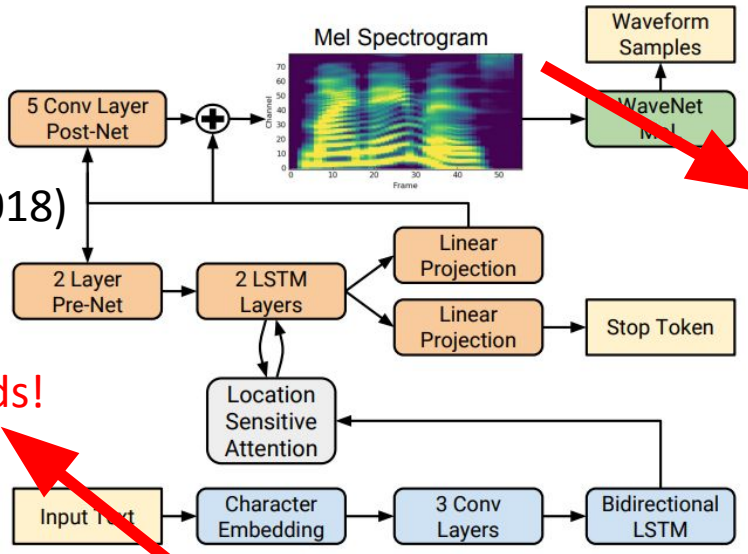


Fig. 1. Block diagram of the Tacotron 2 system architecture.

Codewords!

WaveGlow Vocoder (Prenger et al., 2019)

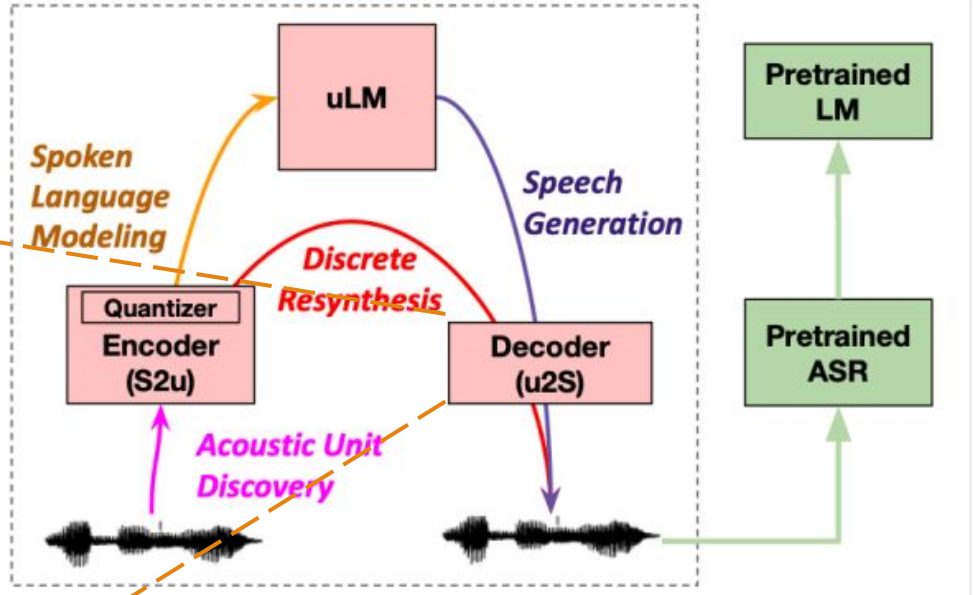


Figure 1: **Setup of the baseline model architecture, tasks and metrics.**

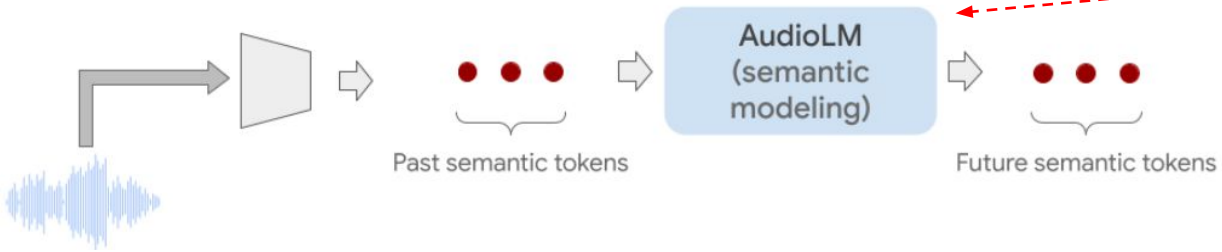
Audio Language Modeling

Let’s listen - https://speechbot.github.io/gslm/

Audio Language Modeling

Decoder-only
Autoregressive Generation

AudioLM (Borsos et al., 2022)



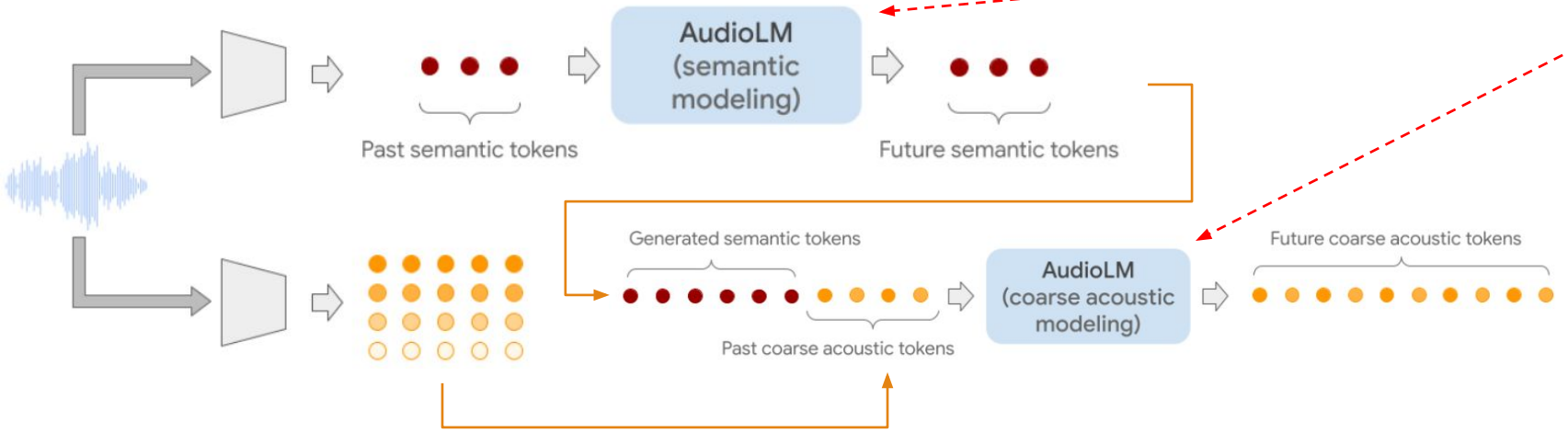
Stage 1: Semantic Modeling

Image sources: https://ai.googleblog.com/2022/10/audiolm-language-modeling-approach-to.html

Audio Language Modeling

Decoder-only
Autoregressive Generation

AudioLM (Borsos et al., 2022)

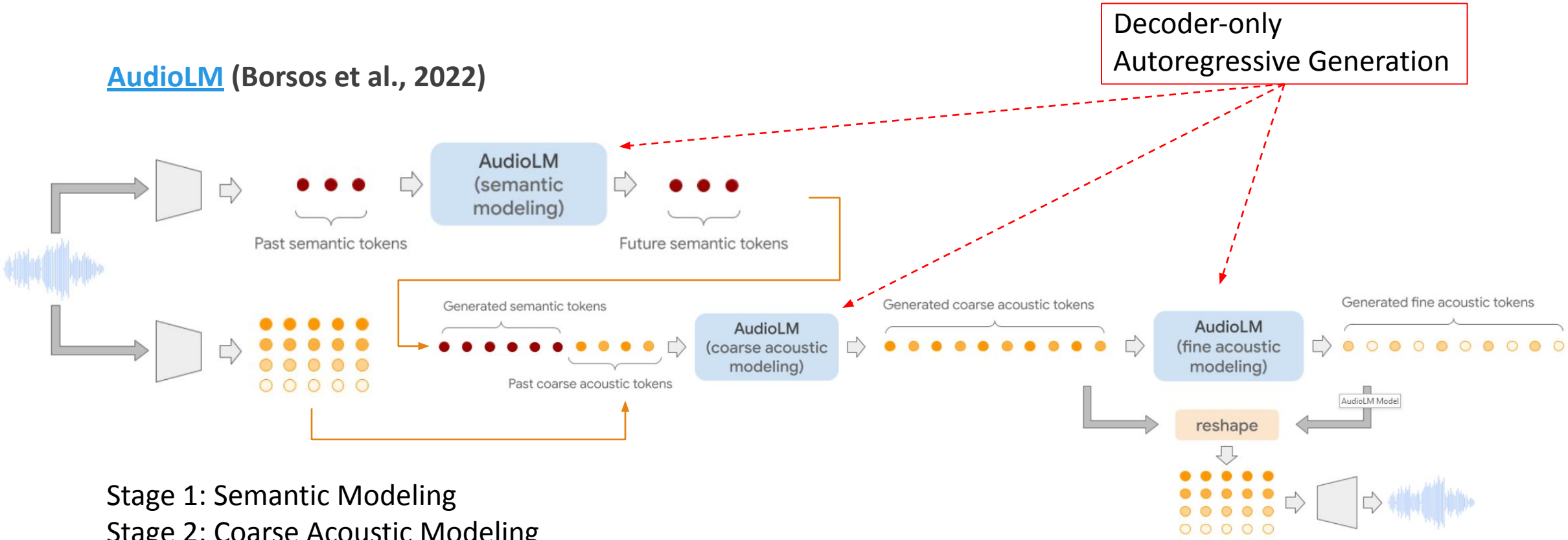


Stage 1: Semantic Modeling
Stage 2: Coarse Acoustic Modeling

Image sources: https://ai.googleblog.com/2022/10/audiolm-language-modeling-approach-to.html

Audio Language Modeling

AudioLM (Borsos et al., 2022)



Stage 1: Semantic Modeling
Stage 2: Coarse Acoustic Modeling
Stage 3: Fine Acoustic Modeling

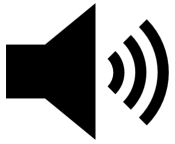
Image sources: https://ai.googleblog.com/2022/10/audiolm-language-modeling-approach-to.html

Audio Language Modeling

AudioLM is a significant step in this direction – textless, audio-only language generation!

AudioLM is a general framework that applies autoregressive language generation principals to any type of audio:

- Speech, Music, including background recording conditions



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

Audio Language Modeling

Modeling raw speech audio is hard – it requires learning the *semantics and pragmatics* of language from audio.

For example:

- Learning what words and sentences mean in context
- Predicting the underlying intent of the speaker
- Remaining coherent beyond a few words at a time

Text LLMs are VERY good at this...

If we condition audio generation on text, the *language modeling* problem reduces to a much easier *translation* problem…

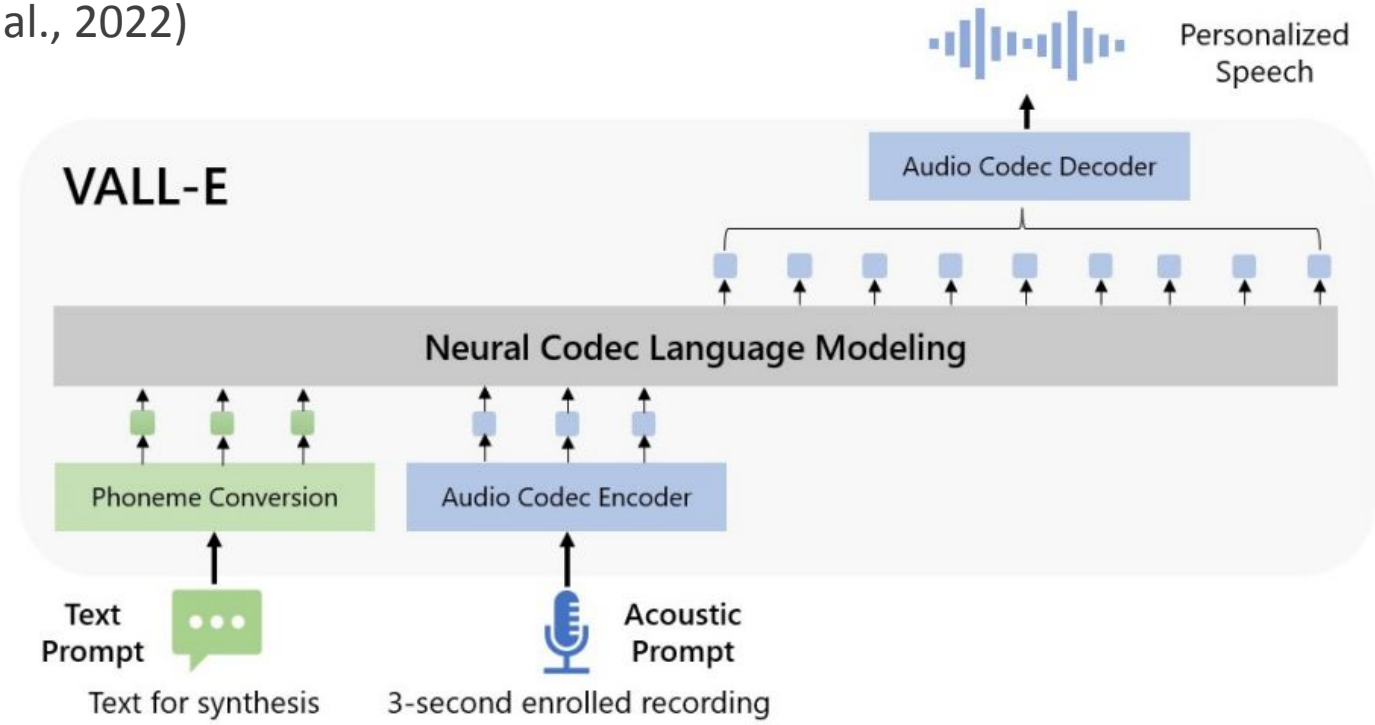
Audio Language Modeling

VALL-E is a Decoder-only Transformer LM that predicts the discrete codes from a pre-trained neural audio codec (EnCodec; Defossez et al., 2022)

https://www.microsoft.com/en-us/research/project/vall-e-x/vall-e/

VALL-E's prompt contains:

1. Text Portion
   1. Phoneme tokens representing ~3 seconds of sample speech from the desired speaker
   2. Phoneme tokens to be synthesized
2. Acoustic Portion
   1. Acoustic tokens (EnCodec codes) for the ~3 seconds of sample speech



Agenda

- Introduction
- Background: Language Modeling
- Audio Tokenization
- Audio Language Modeling
- **Application: ASR, TTS, S2S Tasks**
- Application: Sound & Music Generation
- Application: Full-Duplex Dialogue Agents
- Discussion

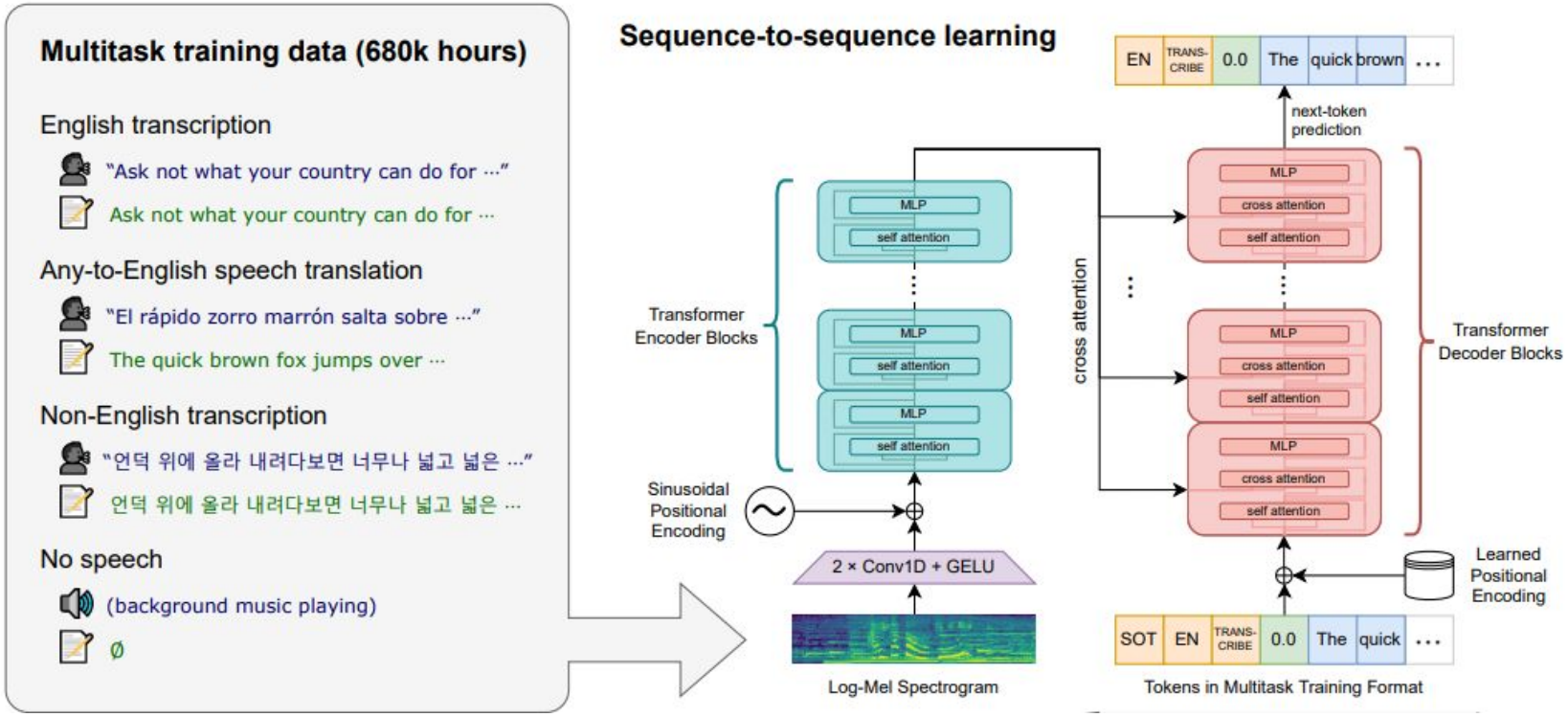
Application: ASR, TTS, S2S Tasks

Automatic Speech Recognition (ASR)

OpenAI's Whisper is a very well-known state-of-the-art ASR model.

This is NOT an audio language model, but rather a text LM with cross-attention to a Log-Mel Spectrogram encoder:

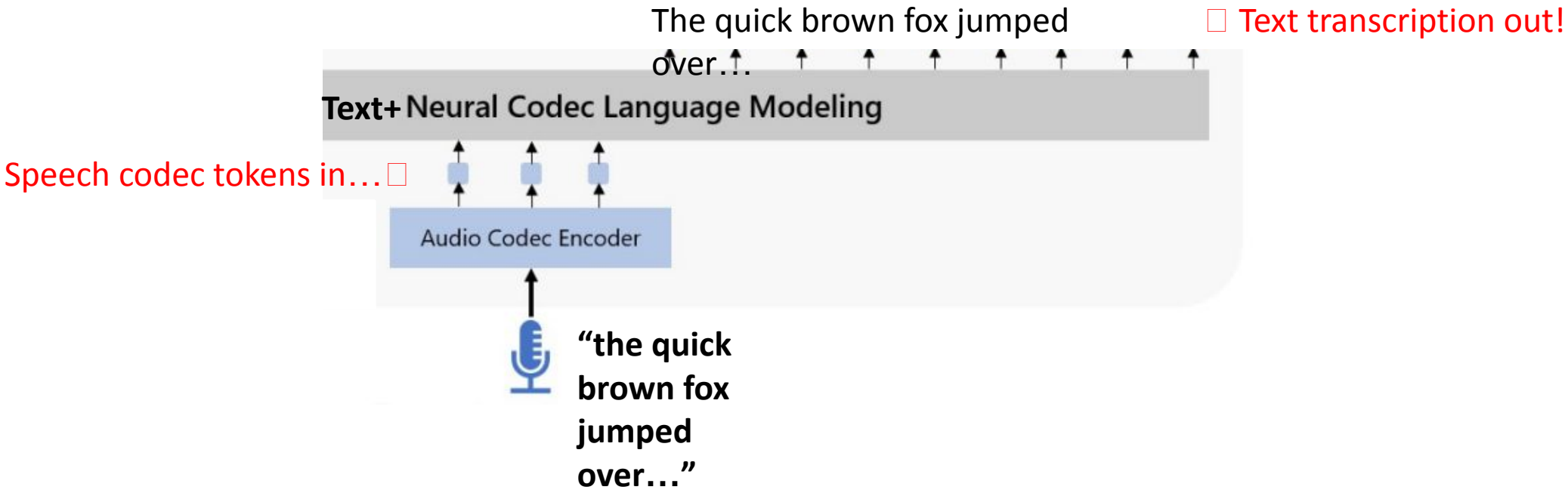
Although not an audio LM, It is very powerful and easy to use so worth a mention here!



Application: ASR, TTS, S2S Tasks

Automatic Speech Recognition (ASR) - Ok, back to Audio Language Models.

ASR in an audio LM can be defined as:



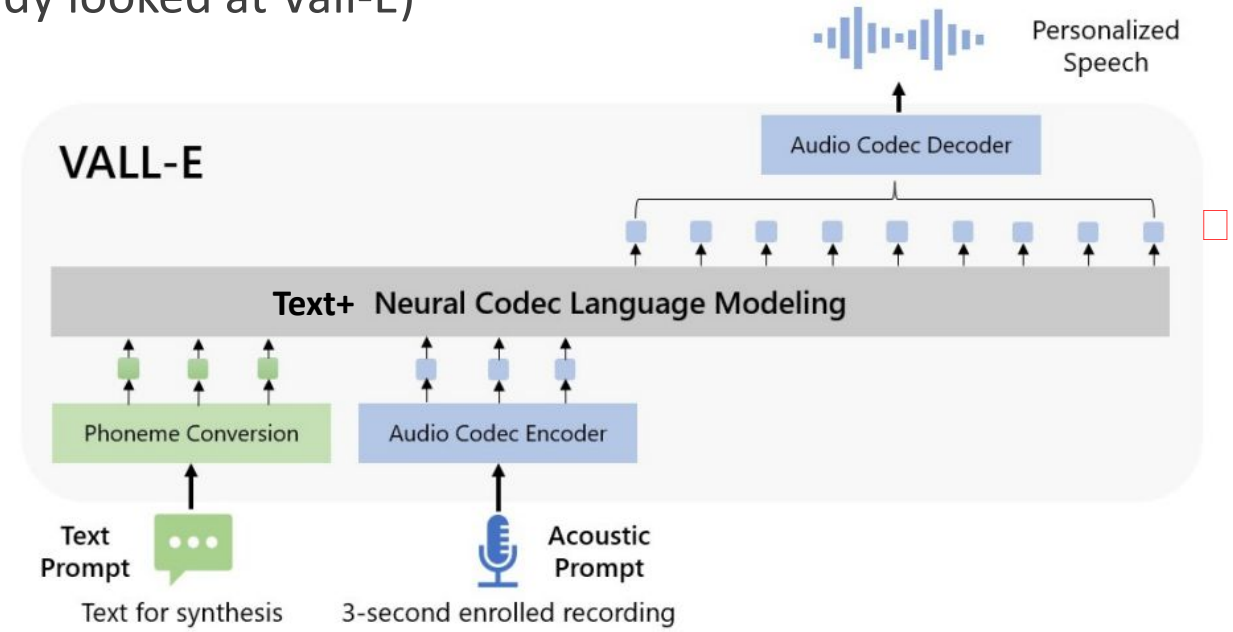
Application: ASR, TTS, S2S Tasks

Text-To-Speech (TTS)

TTS in an audio LM can be defined as the opposite:

(e.g., we already looked at Vall-E)

Text in…



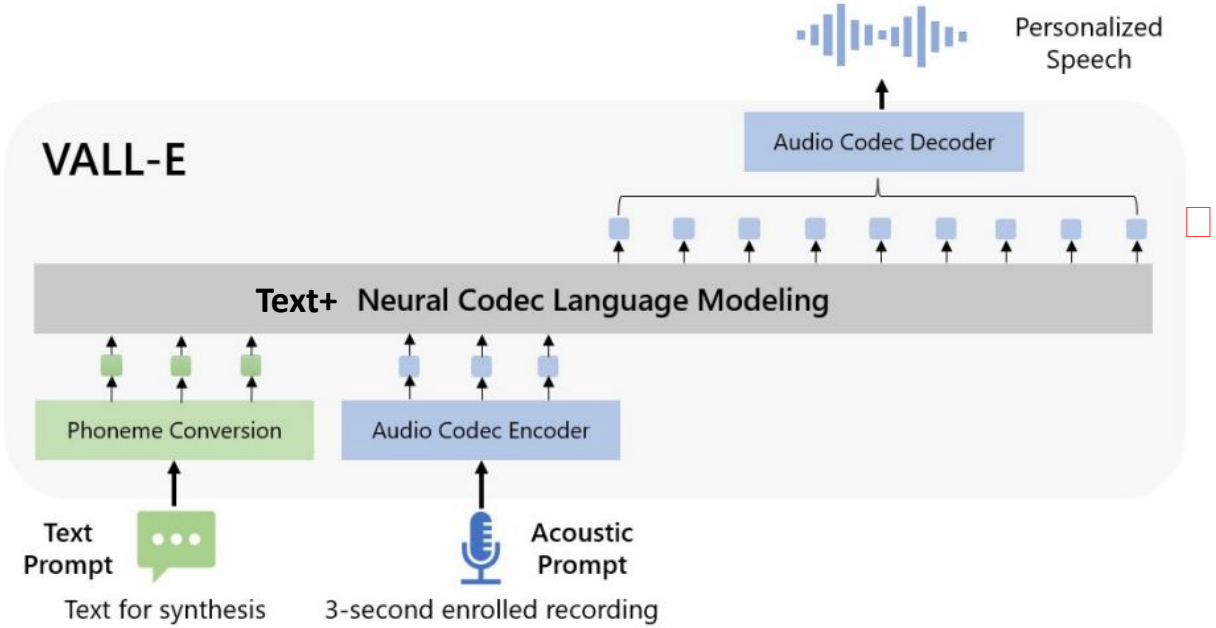
Speech codec tokens out!

Application: ASR, TTS, S2S Tasks

Text-To-Speech (TTS)

TTS models like Vall-E that accept an audio speaker sample along with the text are sometimes called “Voice Cloning” models.

Text in…

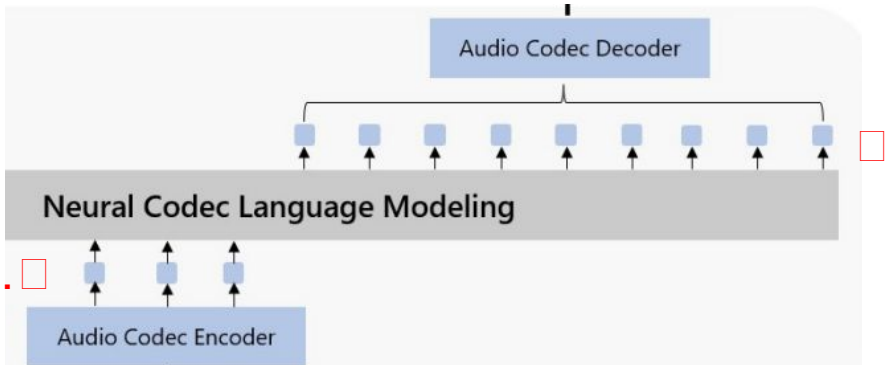


Speech codec tokens out!

Application: ASR, TTS, S2S Tasks

Speech-to-Speech (S2S)

S2S in an audio LM can be defined as:



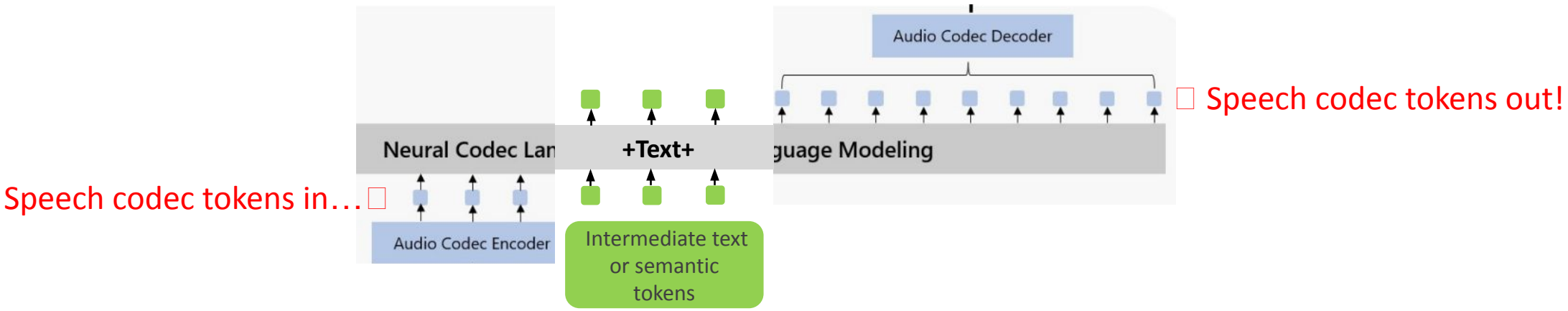
Speech codec tokens in…

Speech codec tokens out!

Application: ASR, TTS, S2S Tasks

Speech-to-Speech (S2S)

Multimodal S2S models can interleave audio and text (or audio semantic tokens) to do the core NLP processing in the text domain before translating back to audio:



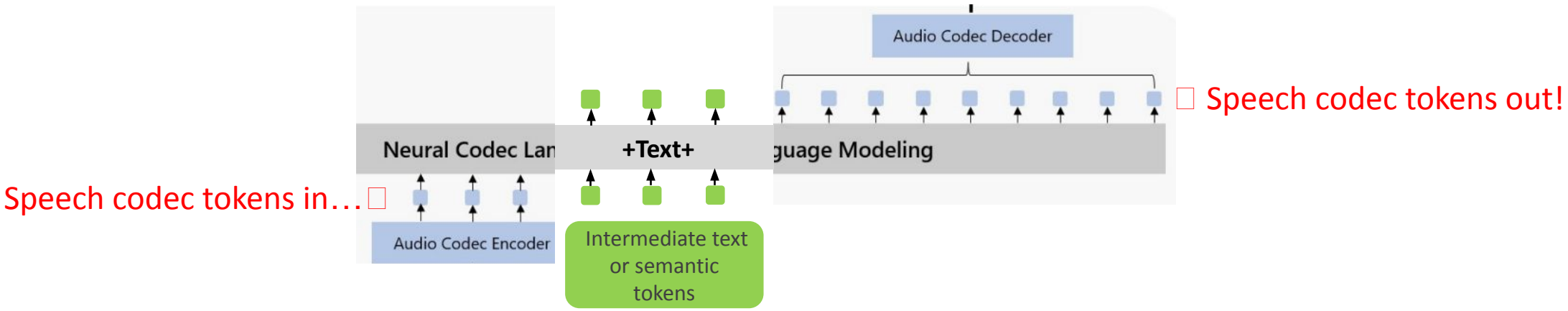
Speech codec tokens in…

Speech codec tokens out!

Application: ASR, TTS, S2S Tasks

Speech-to-Speech (S2S)

Common S2S tasks include Chat, Q&A, Machine Translation, and Voice Style Transfer.



Speech codec tokens in…

Speech codec tokens out!

Application: ASR, TTS, S2S Tasks

Recent examples of Multi-task Audio LMs for ASR, TTS, and S2S

(2023-2024)

Application: ASR, TTS, S2S Tasks

VioLA: Unified Codec Language Models for Speech Recognition, Synthesis, and Translation

(Wang et al., 2023)

Microsoft

Details:

- **Architecture:** Transformer decoder + LSTM
- **Codec:** EnCodec
- **RVQ Handling:** Flatten pattern (8 codebooks)
- **Demo:** Unavailable

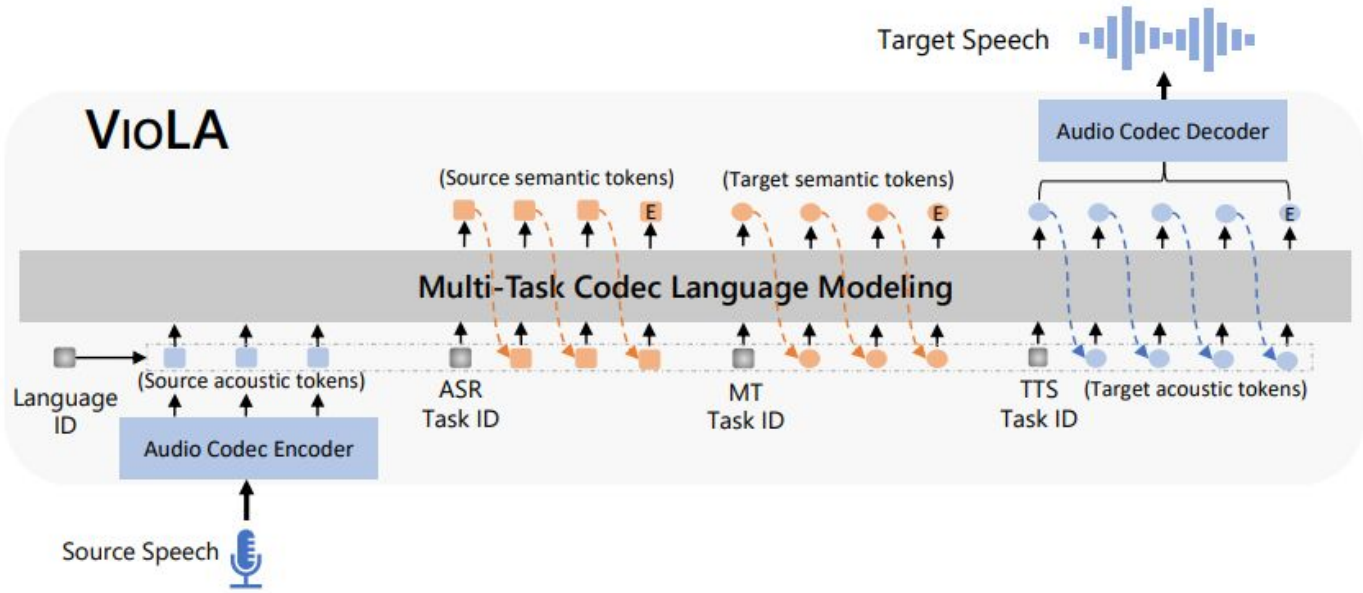


Figure 1: The overall framework of VIOLA, which regards various speech processing tasks as a conditional codec language model task. The model training is conducted on a multi-task learning framework with ASR, MT, and TTS tasks, and the model is capable of performing speech-to-text recognition and translation, text-to-text translation, text-to-speech synthesis, and speech-to-speech translation tasks.

Application: ASR, TTS, S2S Tasks

LauraGPT: Listen, Attend, Understand, and Regenerate Audio with GPT

(Wang et al., 2023)

Alibaba

Details:

- **Architecture:** Transformer decoder
- **Codec:** FunCodec + Vocoder
- **RVQ Handling:** N/A – only one codebook used
- **Demo:** https://lauragpt.github.io/

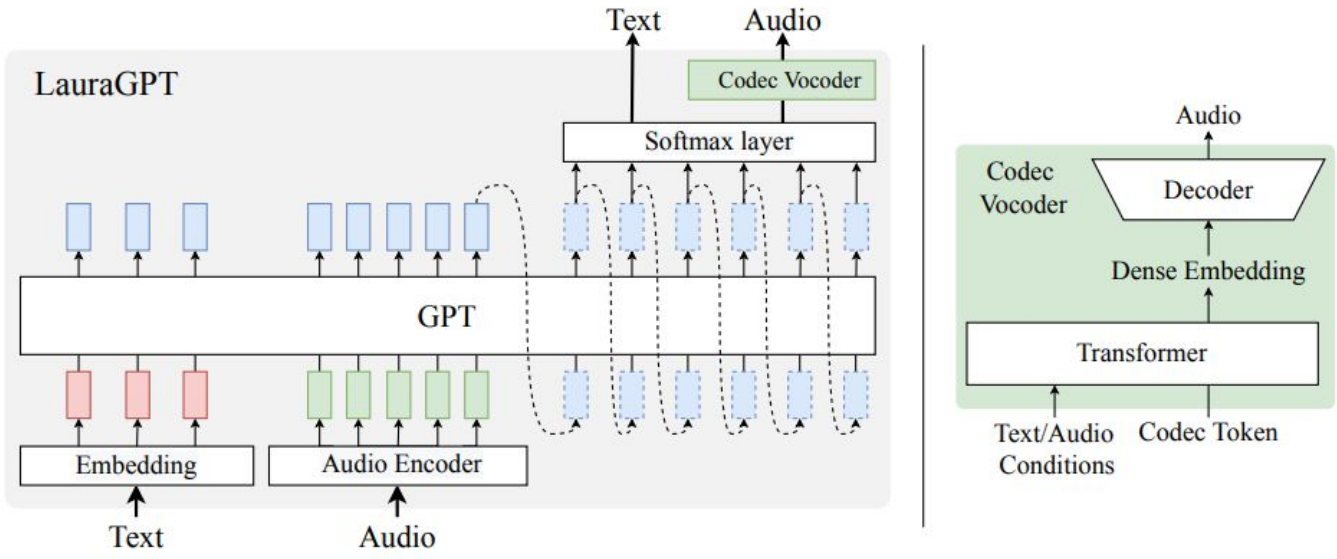


Figure 1: An overview of the proposed LauraGPT model. The right part provides an enlarged view of the Codec Vocoder in LauraGPT. We omit the text tokenizer for simplicity.

Application: ASR, TTS, S2S Tasks

SpiRit-LM: Interleaved Spoken and Written Language Model

(Nguyen et al., 2024)

Meta, Inria Paris, EHESS, ENS-PSL, CNRS Paris

Details:

- **Architecture:** Transformer decoder
- **Codec:** HuBERT Encoder, k-means quantizer, HifiGAN Decoder
- **RVQ Handling:** N/A – no RVQ
- **Demo:** https://speechbot.github.io/spiritlm/

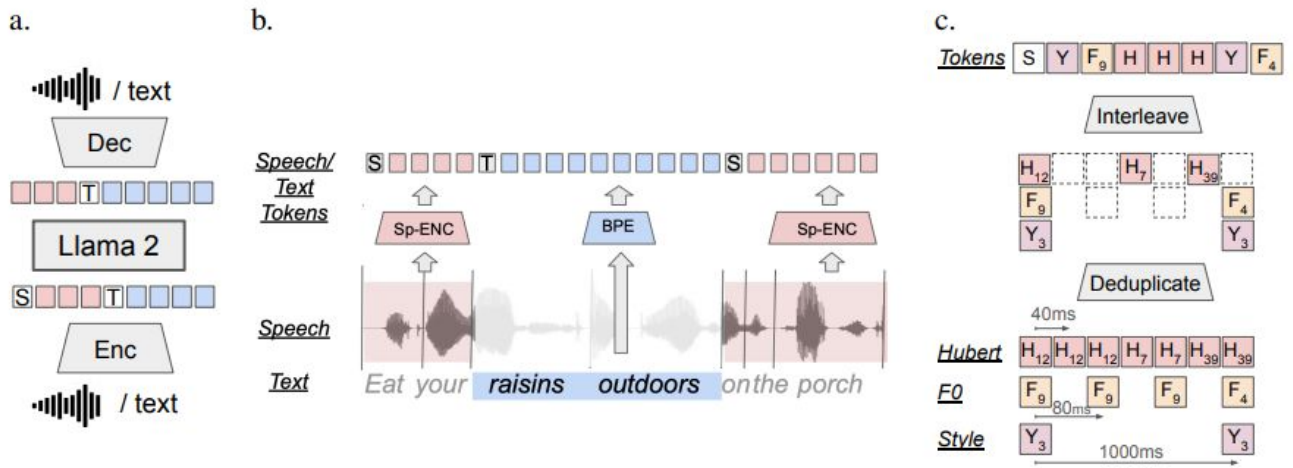


Figure 1: **a. The SPIRIT-LM architecture.** A language model trained with next token prediction; tokens are derived from speech or text with an encoder, and rendered back in their original modality with a decoder. SPIRIT-LM models are trained on a mix of text-only sequences, speech-only sequences, and *interleaved* speech-text sequences. **b. Speech-text interleaving scheme.** Speech is encoded into tokens (pink) using clusterized speech units (Hubert, Pitch, or Style tokens), and text (blue) using BPE. We use special tokens [TEXT] to prefix text and [SPEECH] for speech tokens. During training, a change of modality is randomly triggered at word boundaries in aligned speech-text corpora. Speech tokens are deduplicated and interleaved with text tokens at the modality change boundary. **c. Expressive Speech tokens.** For SPIRIT-LM-EXPRESSIVE, pitch tokens and style tokens are interleaved after deduplication.

Application: ASR, TTS, S2S Tasks

VoiceCraft: Zero-Shot Speech Editing and Text-to-Speech in the Wild

(Peng et al., 2024)

University of Texas at Austin, Rembrand

Details:

- **Architecture:** Transformer decoder
- **Codec:** EnCodec
- **RVQ Handling:** Delay pattern (4 codebooks)
- **Demo:** https://jasonppy.github.io/VoiceCraft_web/

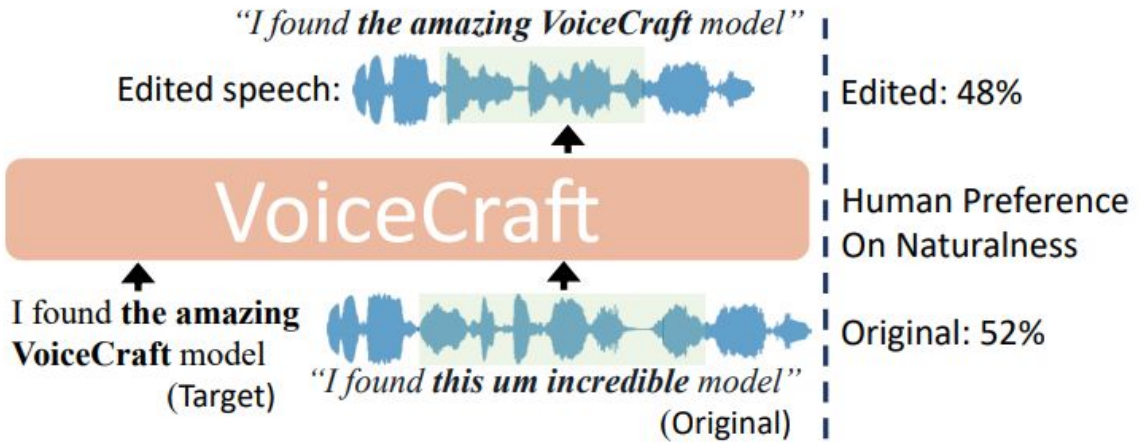


Figure 1: Speech editing with VOICECRAFT. Human listeners prefer VOICECRAFT edited speech over the original real recording 48% of the time in side-by-side naturalness comparison (details in §5.3)

Application: ASR, TTS, S2S Tasks

Natural language guidance of high-fidelity text-to-speech with synthetic

(Lyth & King, 2024)

Stability AI, University of Edinburgh, UK

Details:

- **Architecture:** Transformer decoder with cross-attention to T5 text encodings
- **Codec:** DAC (Kumar et al., 2024)
- **RVQ Handling:** Delay pattern (9 codebooks)
- **Demo:** https://www.text-description-to-speech.com/
- **Open-source reproduction:** "Parler-TTS" by HuggingFace: https://github.com/huggingface/parler-tts

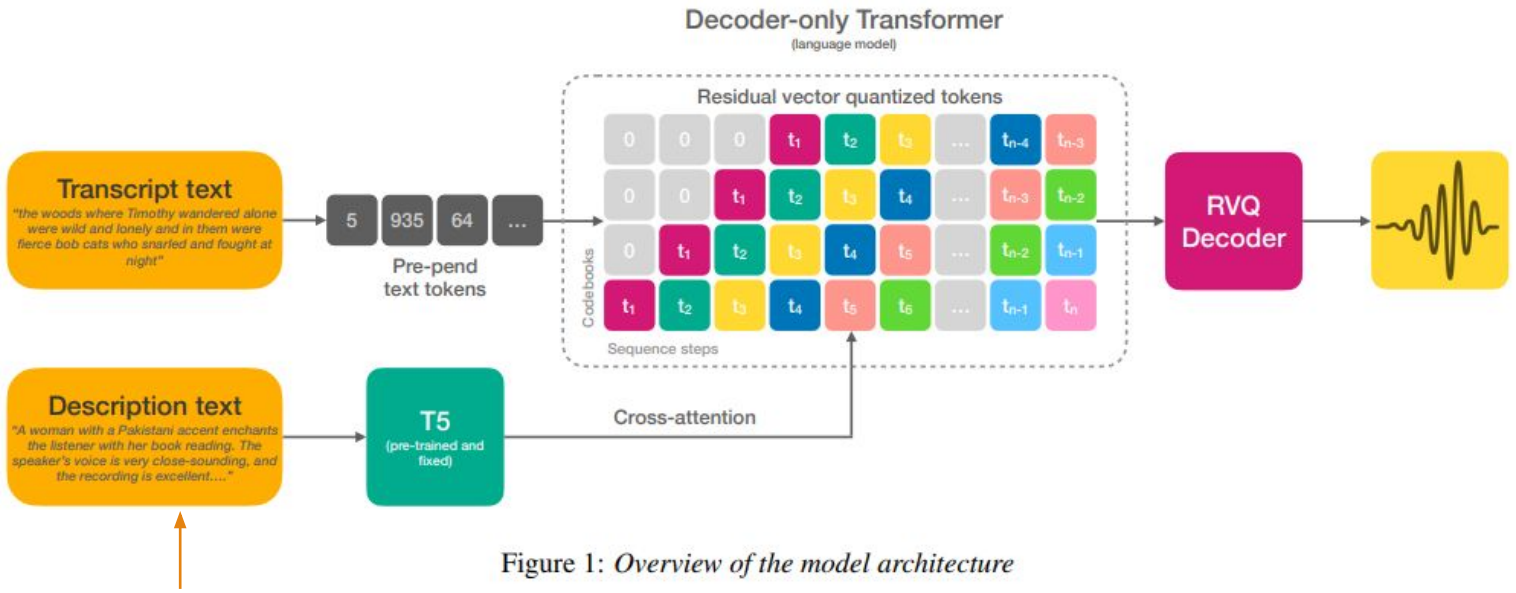


Figure 1: *Overview of the model architecture*

Vocal properties are controllable via a separate speaker description prompt

Application: ASR, TTS, S2S Tasks

Additional Resources:

Other recent ASR / TTS / S2S work from Meta, some of which uses Codecs:

https://ai.meta.com/blog/multilingual-model-speech-recognition/

https://ai.meta.com/blog/seamless-m4t/

https://ai.meta.com/blog/seamless-communication/

An EnCodec-based TTS audio LM from Suno AI:

https://github.com/suno-ai/bark

Agenda

- Introduction
- Background: Language Modeling
- Audio Tokenization
- Audio Language Modeling
- Application: ASR, TTS, S2S Tasks
- **Application: Sound & Music Generation**
- Application: Full-Duplex Dialogue Agents
- Discussion

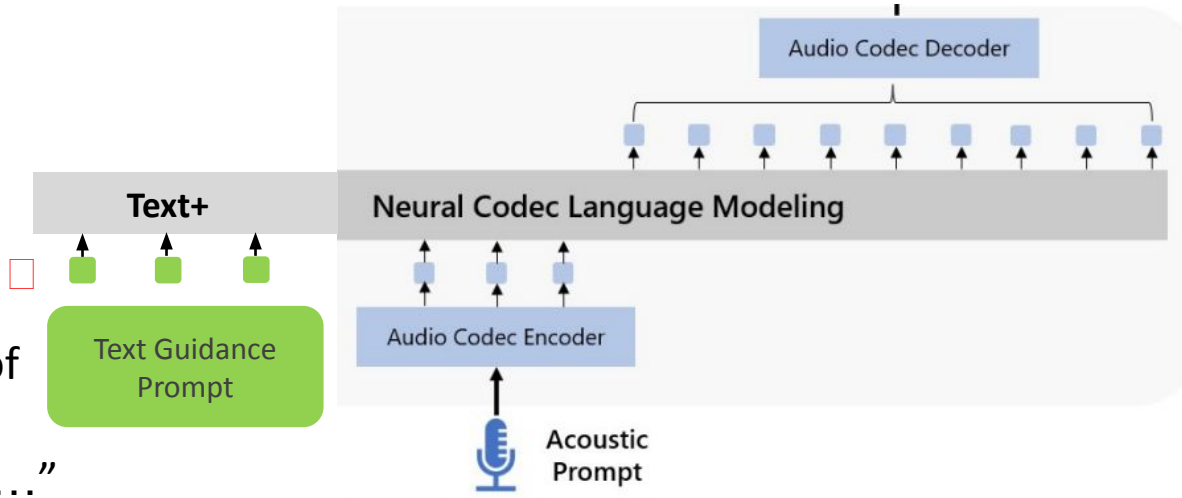
Application: Sound & Music Generation

Text-Guided Audio Generation

Unlike TTS & S2S, the text prompt does not serve as a translation target but as loose guidance. This is akin to text-guided image generation with Stable Diffusion, Midjourney, etc.

Text guidance in…

“Relaxing sound of waves crashing against the shore…”



(Optional) Acoustic style prompt

Audio codec tokens out!

- Music
- Sound effects

Application: Sound & Music Generation

AudioGen: Textually Guided Audio Generation

(Kreuk et al., 2023)

Meta AI, Hebrew University of Jerusalem

Details:

- **Architecture:** Transformer decoder with cross-attention to T5 text encodings
- **Codec:** Custom RVQ-based
- **RVQ Handling:** Parallel pattern (4 codebooks)
- **Demos:** https://felixkreuk.github.io/audiogen/
- https://audiocraft.metademolab.com/audiogen.html
- **Meta Resources:** https://audiocraft.metademolab.com/

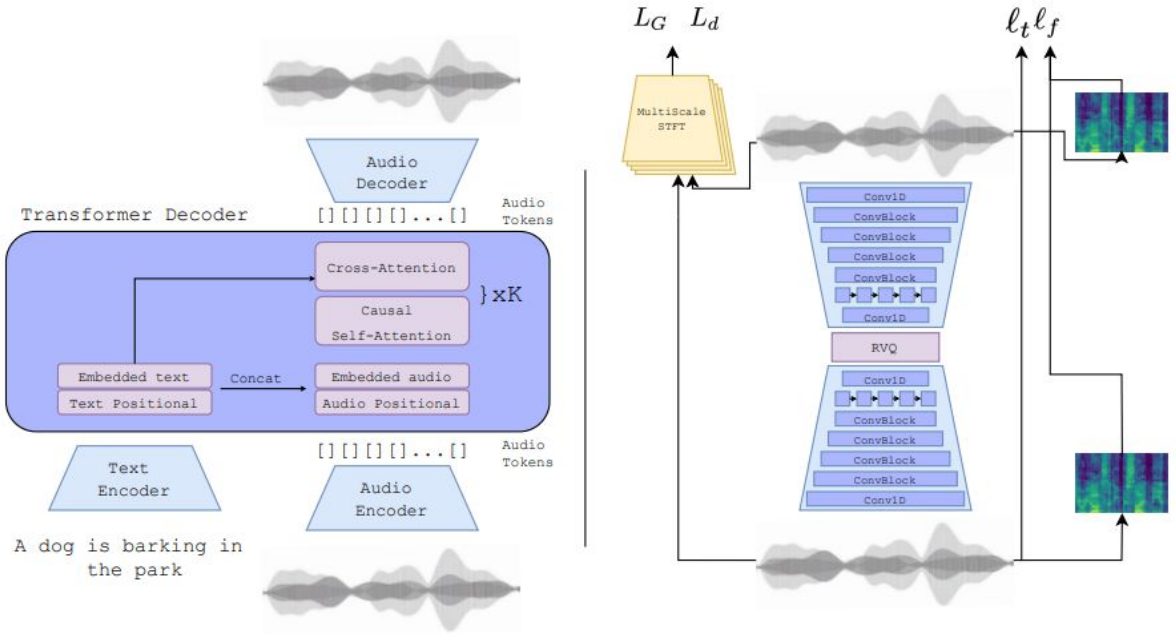


Figure 1: A general overview of the AUDIOGENsystem. Left: the audio representation model. Right: the audio language model. Both text and audio embeddings are concatenated over the time dimension and fed in K causal self-attention and cross-attention blocks with the embedded text.

Application: Sound & Music Generation

Simple and Controllable Music Generation

(Copet et al., 2023)

Meta AI

Details:

- **Architecture:** Transformer decoder
- **Codec:** EnCodec
- **RVQ Handling:** Delay pattern (4 codebooks)
- **Demos:** https://ai.honu.io/papers/musicgen/
- https://audiocraft.metademolab.com/musicgen.html
- **Meta Resources:** https://audiocraft.metademolab.com/

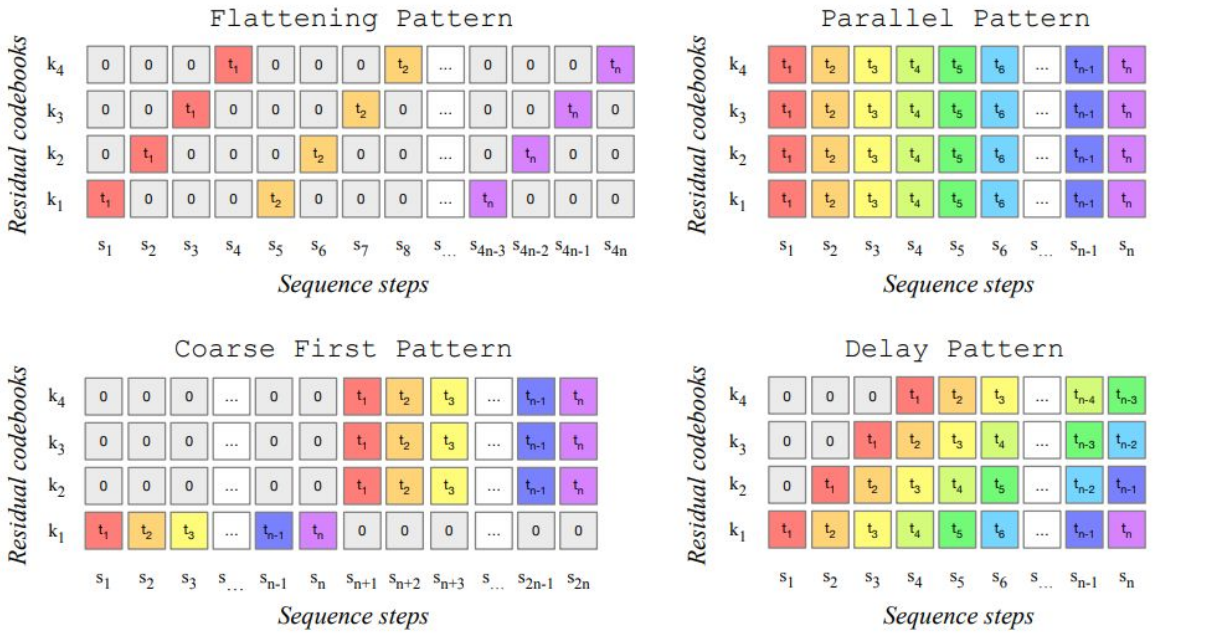


Figure 1: Codebook interleaving patterns presented in Section 2.2. Each time step $t_1, t_2, \ldots, t_n$ is composed of 4 quantized values (corresponding to $k_1, \ldots, k_4$). When doing autoregressive modelling, we can flatten or interleave them in various ways, resulting in a new sequence with 4 parallel streams and steps $s_1, s_2, \ldots, s_m$. The total number of sequence steps S depends on the pattern and original number of steps T. 0 is a special token indicating empty positions in the pattern.

Application: Sound & Music Generation

Additional Resources:

Suno AI's "Chirp" product provides an end-to-end music generation capability including:

- Lyric generation
- Music generation
- Vocal generation adhering to the lyrics (a type of styled TTS)
- Putting it all together

No paper is released, but if they use a similar architecture to Suno Bark, it might be an EnCodec-based Audio LM.

Agenda

- Introduction
- Background: Language Modeling
- Audio Tokenization
- Audio Language Modeling
- Application: ASR, TTS, S2S Tasks
- Application: Sound & Music Generation
- **Application: Full-Duplex Dialogue Agents**
- Discussion

Application: Full-Duplex Dialogue Agents

LLM chatbots are usually text-based. However, speech communication is richer than text:

- Vocal inflections
- Non-linguistic acts (e.g., laughter, backchannels - “mhmm”, fillers – “umm”)
- Non-verbal cues (e.g., hand gestures, head nods, eye contact)
- Timing (e.g., pauses, gaps, overlaps)
- Prosody (e.g., pitch, rate, intensity)

&=laughs it was so good, you know? ↗

Yeah! (0.2) Mhm

It had (0.4) uhh (1.1) it had strawberry icing…

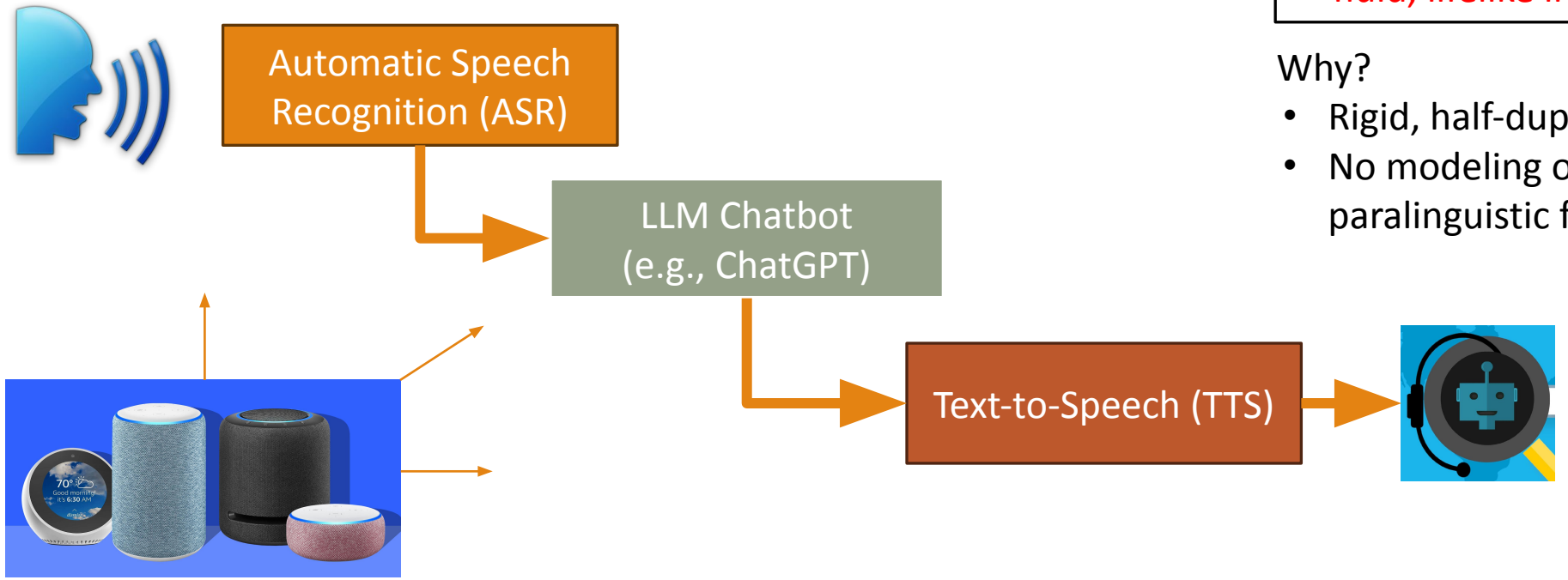
- CHAT-CA annotations (MacWhinney, 2019)

Application: Full-Duplex Dialogue Agents

- You can stick a text LLM in one of these things.

Application: Full-Duplex Dialogue Agents

This would be an ASR->LLM->TTS system, a.k.a a ***cascaded*** model:



These models fail to produce fluid, lifelike interaction!

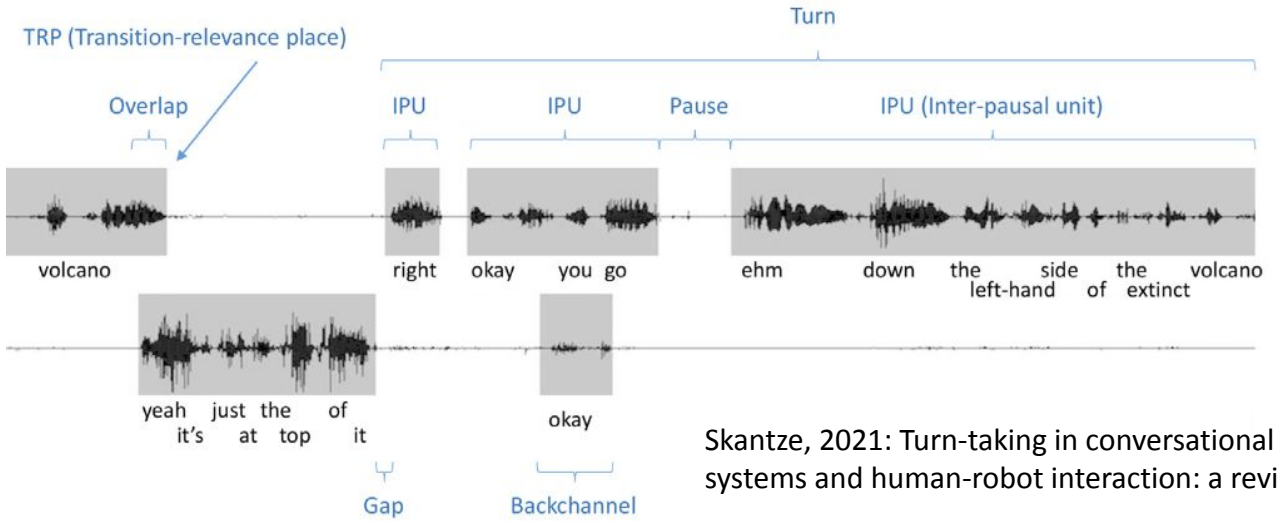
Why?

- Rigid, half-duplex turn-taking
- No modeling of prosodic & paralinguistic features

Application: Full-Duplex Dialogue Agents

Natural human conversations contain:

- Positive & Negative floor transfer offsets (gaps + overlaps)
- Filled Pauses (umm, you know…)
- Backchannels (mhm, right, yeah)
- Prosodic clues toward intention & emotion
  - Intensity, speed, word emphasis, tone, drawn out syllables, etc.
- Paralinguistic speech
  - Laughter, sighing, grunting, squealing, humming, singing, etc.
- Choral speech (speaking in chorus)
- Turn-taking & turn-yielding cues
  - Rising / falling pitch
  - Pause length
  - Semantic cues



Skantze, 2021: Turn-taking in conversational systems and human-robot interaction: a review.

Cascaded SDS do none of these well (if at all!)

Application: Full-Duplex Dialogue Agents

1 S1: &=laughs ▶
2 S2: he was supposed to press one righ- did he say to press one? ▶
3 S1: no ▶
4 S2: oh (.) okay I guess we're okay ▶
5 S2: [Uhm] ▶
6 S1: [mm hmm hmm] ▶
7 S2: alright so um okay it'll complain later when we t hang up ▶
8 S1: ah okay ▶
9 S2: [okay, so what's up] ? ▶
10 S1: [hhh hhh] ▶
11 S1: not much, um I, I'm tell you about the (.) about keep on changing m ▶
12 S2: that's okay ▶
13 S1: uhh (.) I have I dunno, I'm kind of out there right now and I'm won bothering me so I [wanted to] discuss it now [um] ▶
14 S2: [Okay] [let's talk] ▶
15 S1: I don't know you uh you wanted to talk as as you you asked to talk
16 S1: [so I did-] ▶
17 S2: [yeah] I uh (.) haven't been in the mood to write ▶
18 S1: [•huh•]? ▶
19 S2: [&=clears_throat] ▶
20 S2: I've uh I've been really busy and haven't written and then [I uh] ▶
21 S1: [Oh that's okay] ▶
22 S2: [I didn't didn't haven't really] ▶
23 S1: [How well how is everything okay] [or] ▶
24 S2: [haven't really] been in the mood to write ▶
25 S1: oh okay ▶

Cascaded, half-duplex SDS do none of these well (if at all!)

Imagine speaking to Amazon Alexa or Google Assistant.

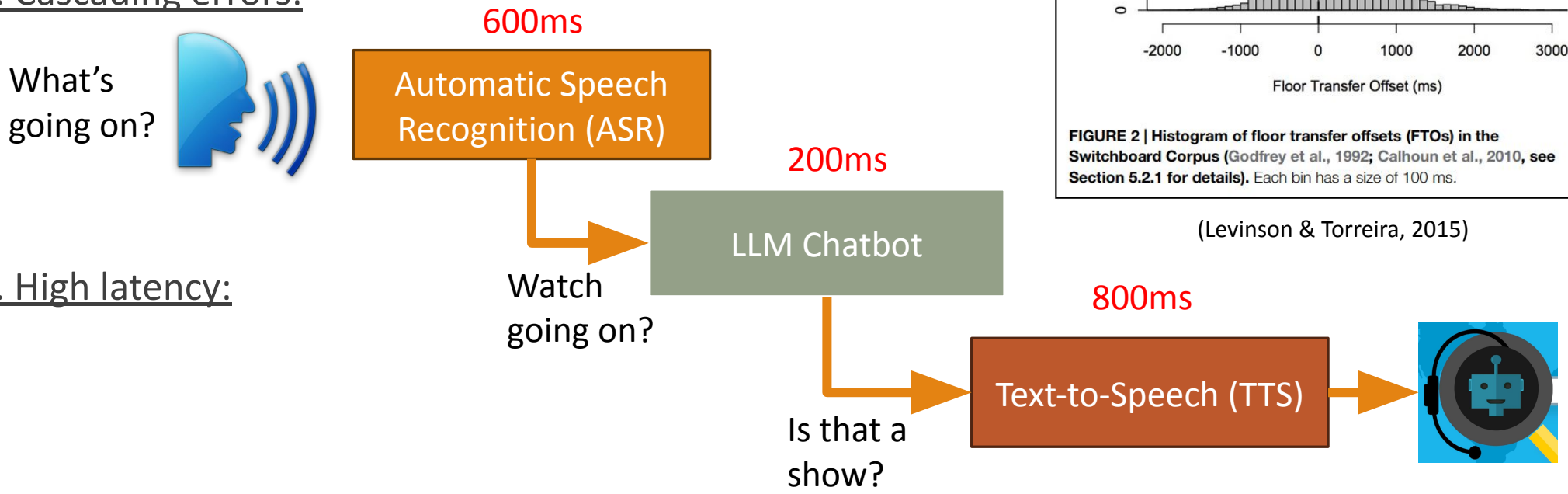
To illustrate, let's compare ChatGPT's voice chat to real human phone calls:
https://www.youtube.com/watch?v=RcgV2u9Kxh0

https://sla.talkbank.org/TBB/ca/CallFriend/eng-n/6062.cha
https://sla.talkbank.org/TBB/ca/CallFriend/eng-n/4708.cha

Application: Full-Duplex Dia Agents

Even if cascaded models could model all of those phenomena, they still suffer from two inescapable issues:

1. Cascading errors:



2. High latency:

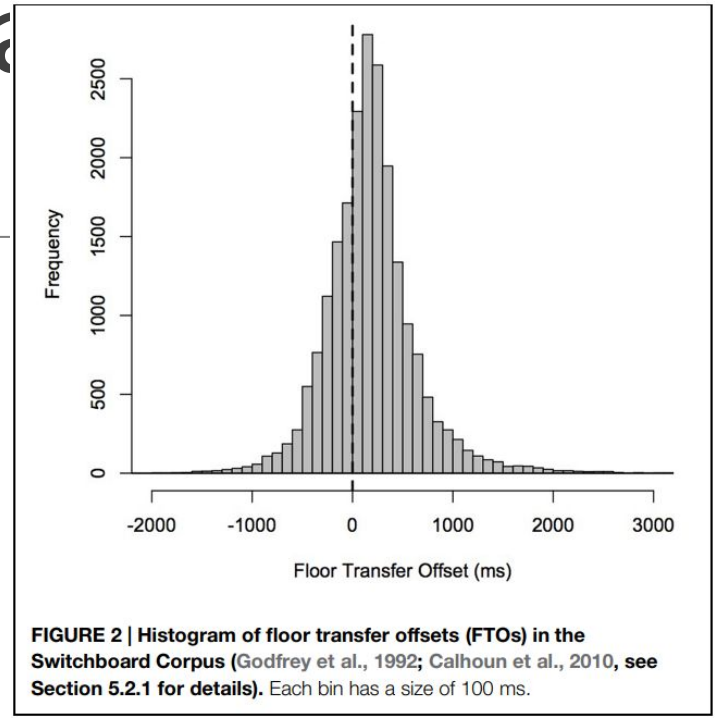
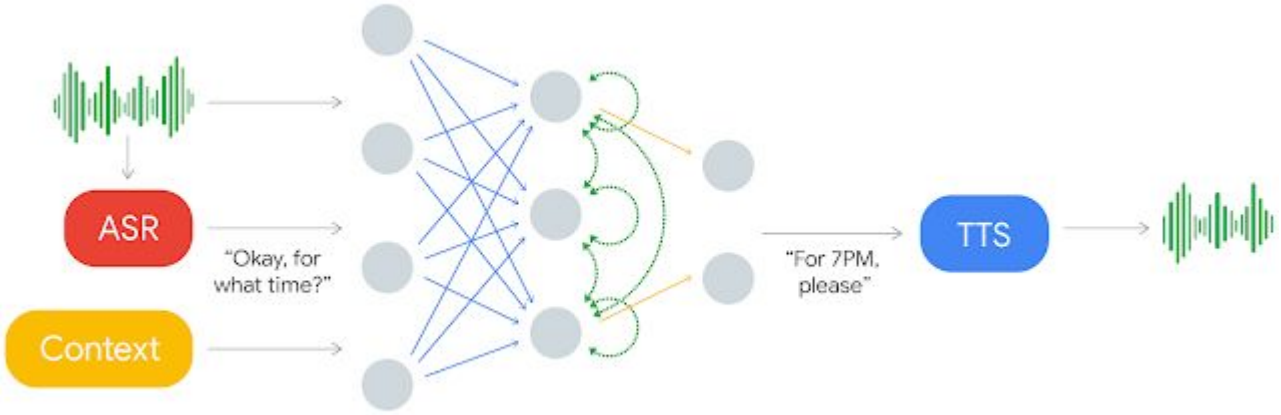


FIGURE 2 | Histogram of floor transfer offsets (FTOs) in the Switchboard Corpus (Godfrey et al., 1992; Calhoun et al., 2010, see Section 5.2.1 for details). Each bin has a size of 100 ms.

(Levinson & Torreira, 2015)

Application: Full-Duplex Dialogue Agents

One early attempt to create a full-duplex dialogue system with continuous turn-taking and pause prediction: Google Duplex (Leviathan & Matias, 2018)



Source: https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html

Demo (@ google I/O 2018):

https://www.youtube.com/watch?v=D5VN56jQMWM&t=40s

Application: Full-Duplex Dialogue Agents

Google Duplex was impressive, but no paper was released. As a 2018 work, it was likely very specialized and domain-restricted (i.e. could not handle *general* conversation)

So, why not use a S2S audio language model to achieve a *generalized* full-duplex experience?

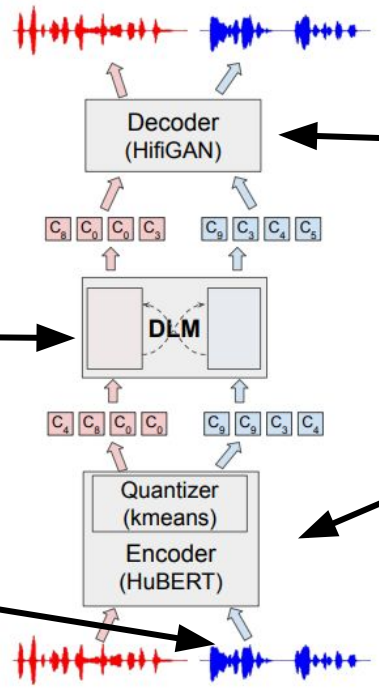
- **This is what dGSLM does!**
  Let's listen:
  https://speechbot.github.io/dgslm/

Unlike the other S2S approaches we looked at, dGSLM does not use intermediate text or semantic tokens and cannot reduce the audio language modeling problem to modal translation!

The lack of semantic conditioning makes this a much harder problem.

Application: Full-Duplex Dialogue Agents

dGSLM (or just "DLM") is GSLM for Dialogue:



HifiGAN instead of Tacotron2+WaveGlow

Similar to AudioGen!

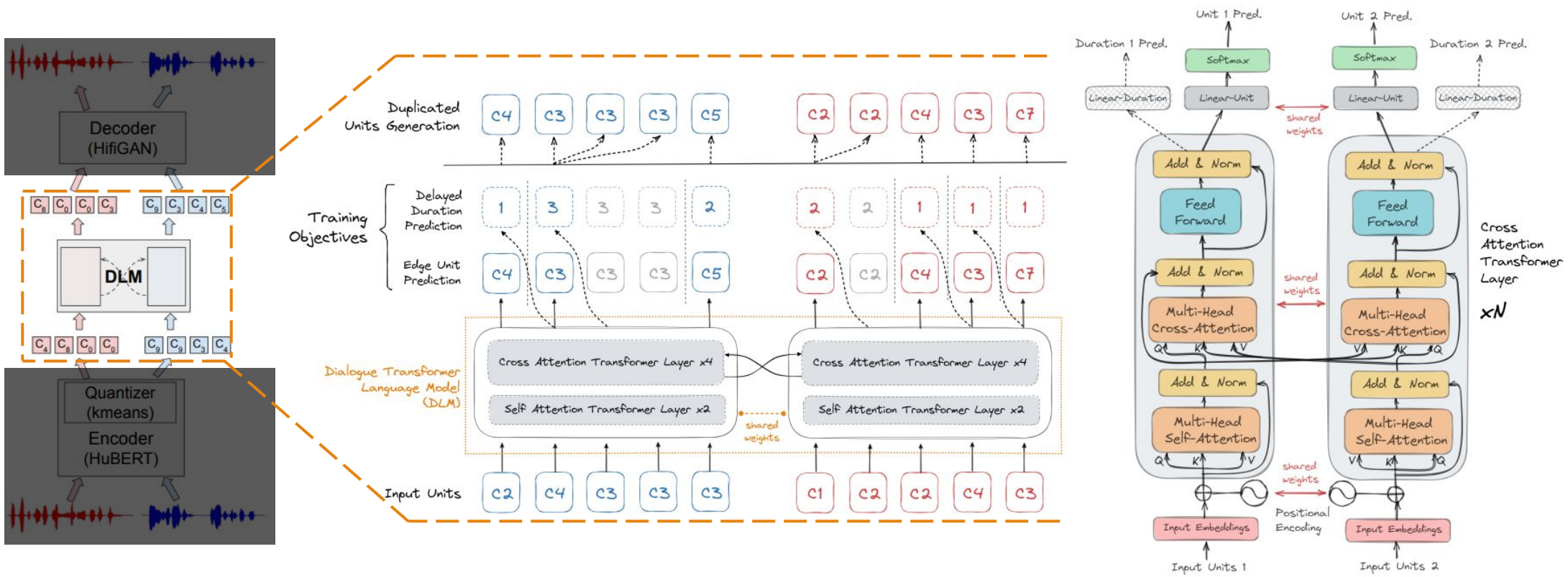
Dual-tower LLM

HuBERT encoder
K-means quantizer
(codewords)

Two concurrent channels
(one per speaker)

Figure 1: General Schema for dGSLM: A discrete encoder (HuBERT+kmeans) turns each channel of a dialogue into a string of discrete units $(c_1, ..c_N)$. A Dialogue Language Model (DLM) is trained to autoregressively produce units that are turned into waveforms using a decoder (HifiGAN).

Application: Full-Duplex Dialogue Agents



Application: Full-Duplex Dialogue Agents

- What is next? Can we solve dGSLM's coherence issue and get something up and running for the open-source AI community?

- Perhaps just scale it up and train on more data?
- Perhaps align its embedding space with text LLMs that are already very coherent?
- Perhaps use intermediate text / semantic tokens?
- This is the topic of my PhD thesis!

Agenda

- Introduction
- Background: Language Modeling
- Audio Tokenization
- Audio Language Modeling
- Application: ASR, TTS, S2S Tasks
- Application: Sound & Music Generation
- Application: Full-Duplex Dialogue Agents
- Discussion

Discussion

Want to get your hands dirty?

Many of the paper, model, and demo links throughout this presentation have open-source implementations on GitHub.

Many are also on HuggingFace, including:

https://huggingface.co/openai/whisper-large-v3 (ASR)

https://huggingface.co/suno/bark (TTS)

https://huggingface.co/parler-tts/parler_tts_mini_v0.1 (TTS)

https://huggingface.co/facebook/audiogen-medium (Sound effect generation)

https://huggingface.co/facebook/musicgen-large (Music generation)

And many others too!

Thank You!