Interspeech 2026
27 September - 1 October
ICC Sydney, Australia
Speaking Together

Speech Large Language Models for Under-Resourced Languages

Zhe LI

Mieradilijiang Maimaiti

Zilong Huang

Hao Huang

Nurmemet Yuluwas

Ricky Chan

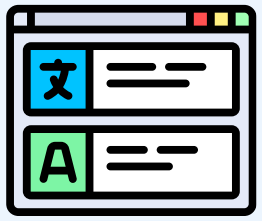
Contents

- Part I: From LLMs to Speech LLMs
- Part II: Components in Speech LLMs
- Part III: Training, Efficient Adaptation and Alignment Recipes
- Part IV: Real-Time Interaction
- Part V: Under-Resourced Speech LLMs

Part I: From LLMs to Speech Language Models

From Text-based LLMs to Speech Interaction

1. Text-based LLMs



- Strong text generation
- Powerful language understanding

2.Text-only Interaction

- Effective, but limited to typed input/output

3. Limitation

- Less natural human-computer communication

4. Speech Interaction

- More intuitive
- More intuitive
- Real-time interaction

5.Speech Language Models

- Unify language understanding with spoken interaction

Why do we need Speech Language Models?

1. Natural interaction

Humans communicate by speaking.

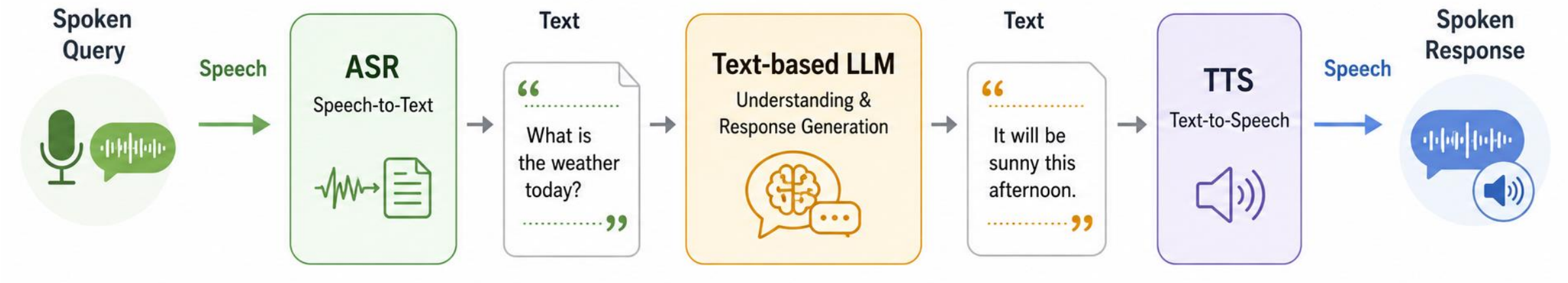
2. Richer information

Speech includes linguistic content plus prosody, speaker traits, and emotion.

3. Real-world usability

Enables hands-free, intuitive, and real-timeAI interaction.

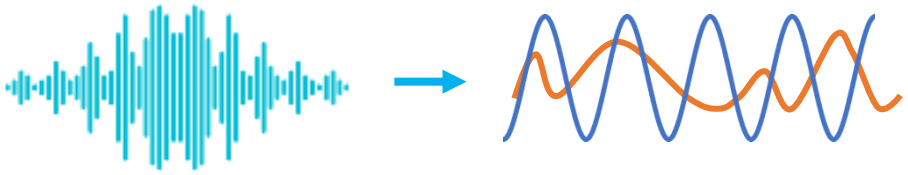
Why Do We Need Speech Language Models



Naive speech interaction pipeline:

Speech → ASR → Text → LLM → Text → TTS → Speech

Hypothesis

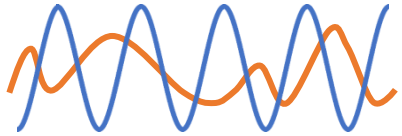


Speech

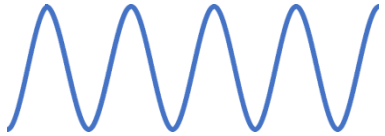
Speech is a combination of **Static** and **Dynamic** components.

Hypothesis

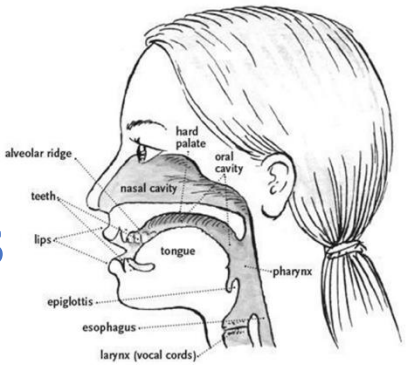
Speech



Dominated by **speaker characteristics**



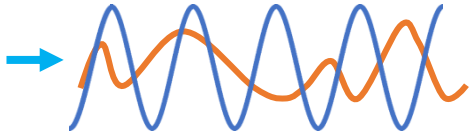
Static



vocal organs

Tone, Pitch, Prosody, Gender, et al.

Hypothesis



Speech

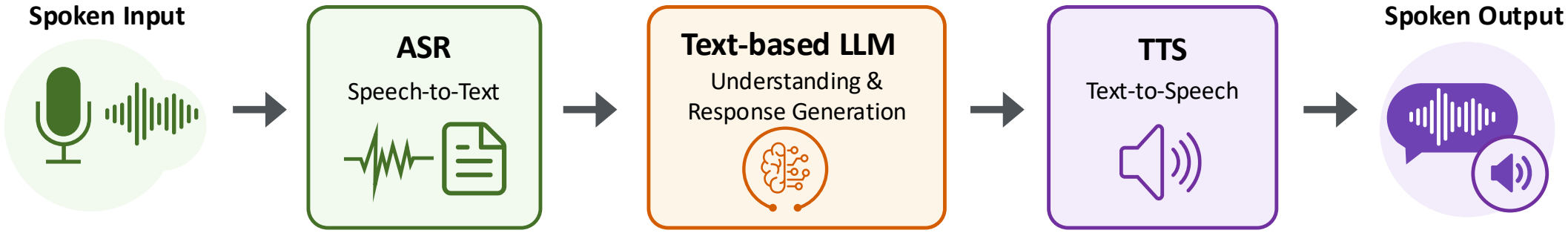
Static

Dominated by **speaker characteristics**

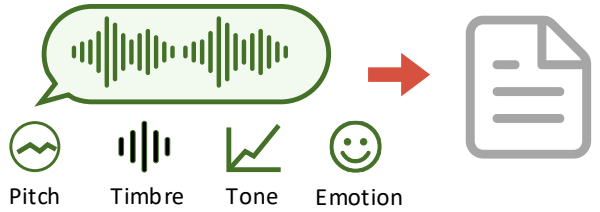
Dynamic

Mainly consist of **speech content**

Limitations of Cascaded Speech Interaction

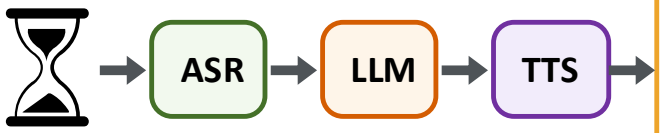


1. Information Loss



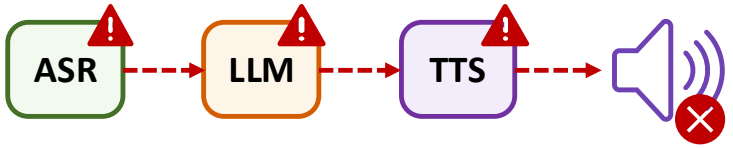
Speech is reduced to text, losing paralinguistic cues such as pitch, timbre, tone, and emotion.

2. Significant Latency



ASR, LLM, and TTS operate sequentially, leading to high end-to-end response time.

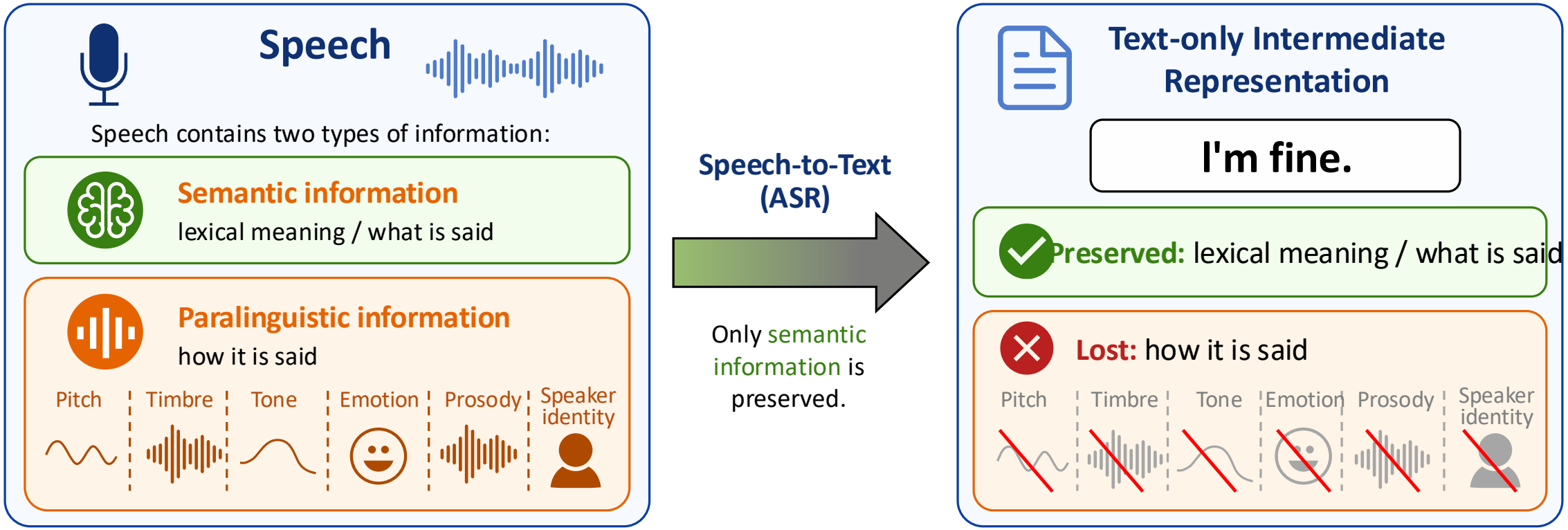
3. Cumulative Error



Errors from one stage can propagate to later stages and degrade the final response.

These limitations motivate the development of Speech Language Models (Speech LLMs)

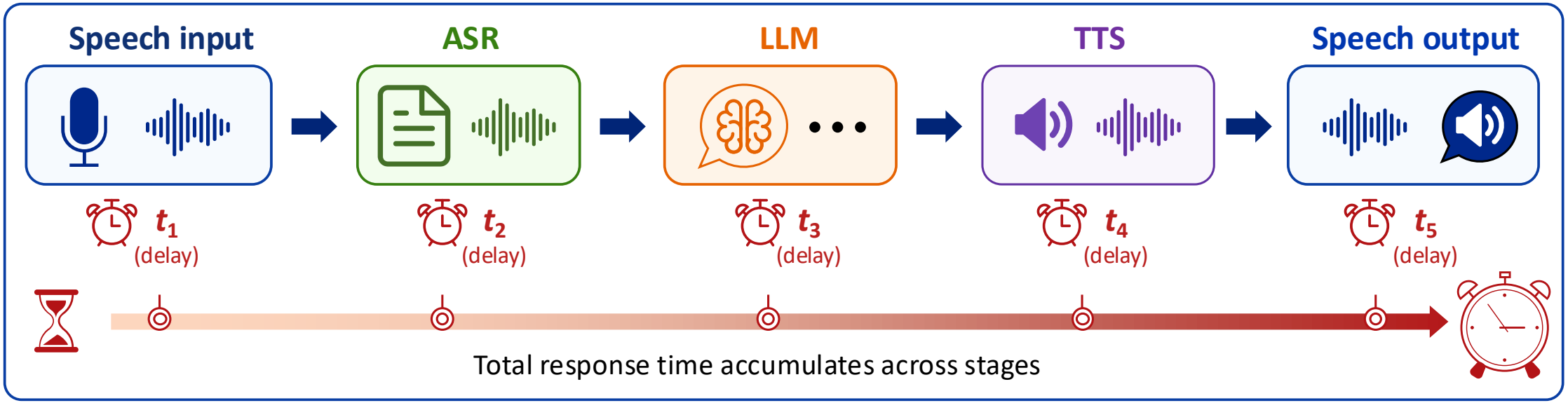
Limitation 1: Information Loss



Text preserves what is said, but not how it is said.

Limitation 2: Significant Latency

A modular pipeline is easy to build, but hard to make real-time.



ASR cost
Speech recognition and decoding

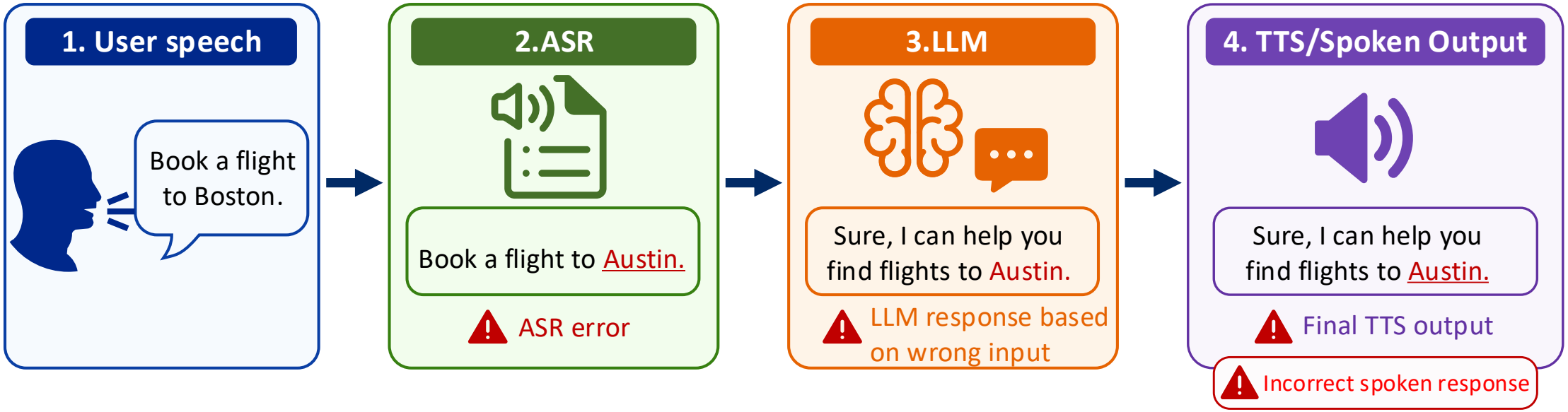
LLM cost
Reasoning and text generation

TTS cost
Text processing and waveform synthesis

High end-to-end latency limits smooth and natural real-time voice interaction

Limitation 3: Cumulative Error

In a cascaded system, one mistake can affect every downstream stage.



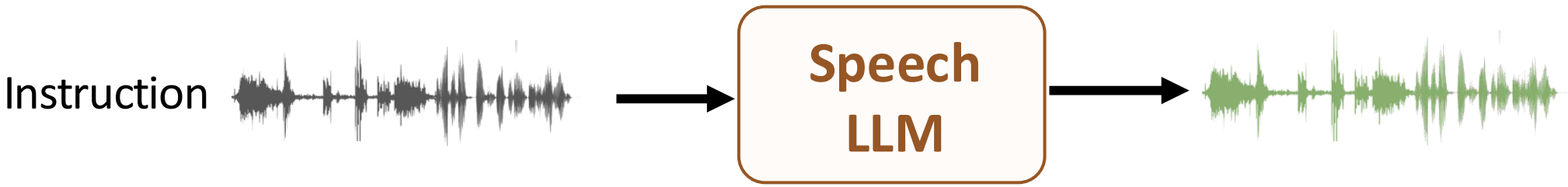
The output of each stage becomes the input to the next stage.

Cascaded architectures are vulnerable to error propagation and reduced robustness.

What is Speech LLM

Different sources of literature provide various definitions for spoken LMs. The definition mentioned above applies specifically to the discussion here.

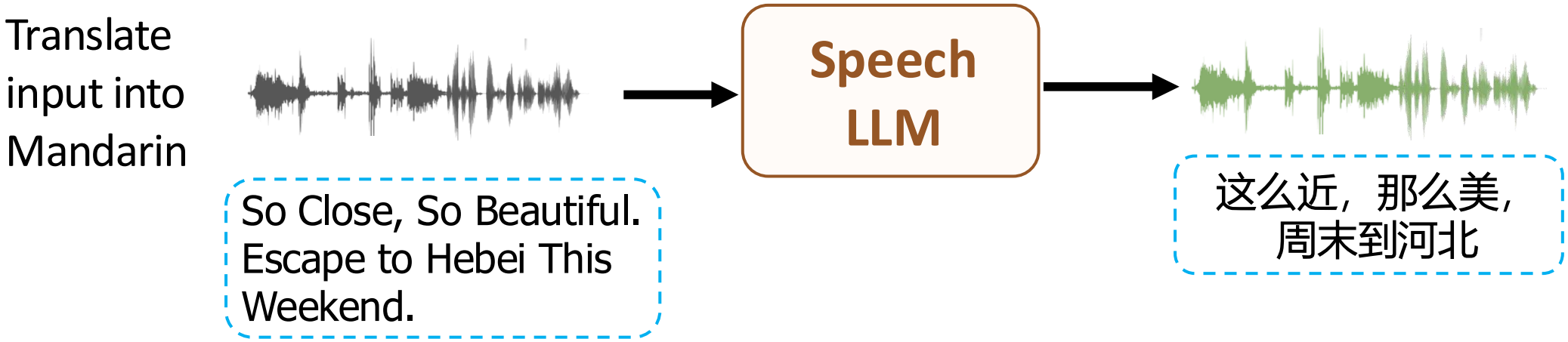
- At least one of the input or output must be speech.
- It can perform various tasks, without the need for fine-tuning.
- It follows natural language instructions.



Different Types of Spoken LMs

- Type 1: Command Mode

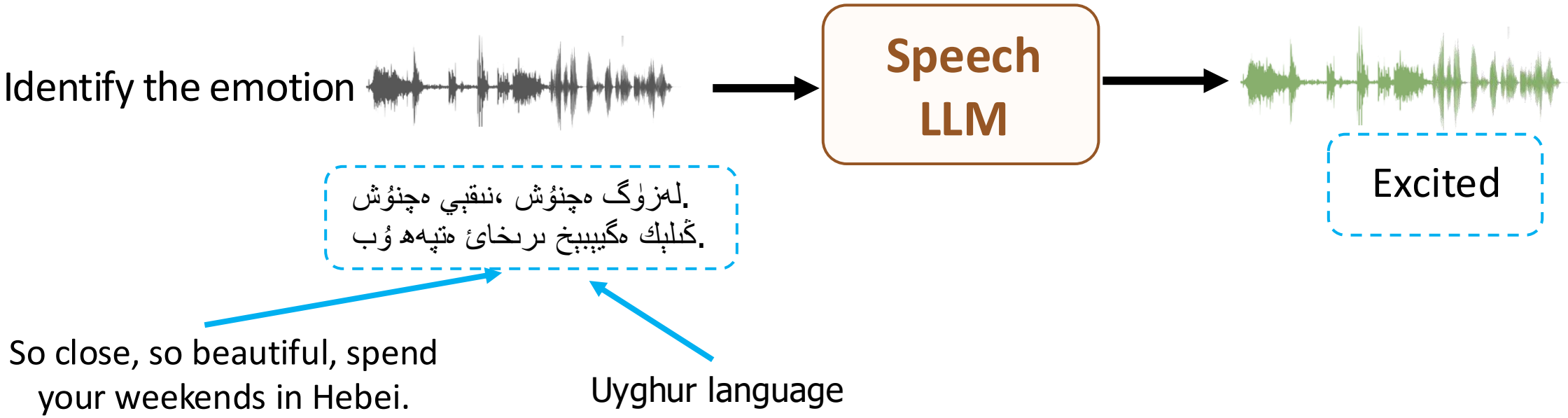
  The spoken LM processes the input speech based on instruction.
  Instruction and speech input are separate.



Different Types of Speech LLM

- Type 1: Command Mode

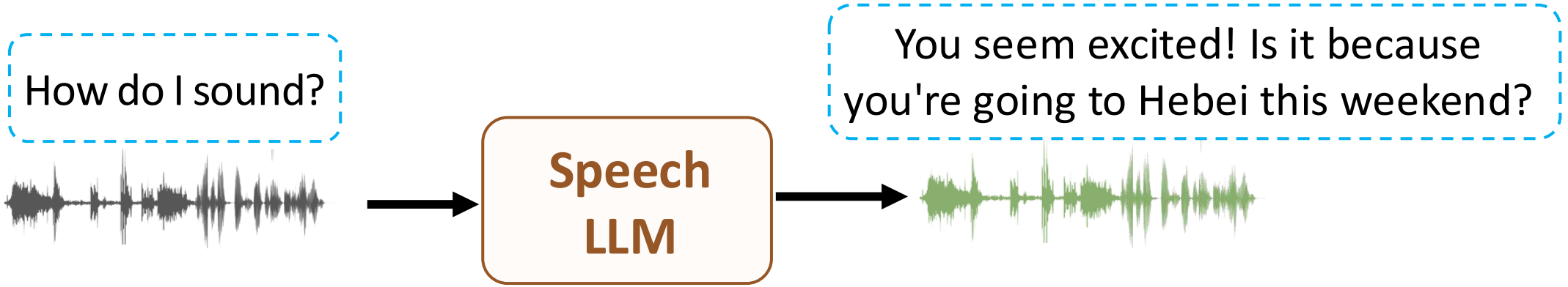
  The spoken LM processes the input speech based on instruction.
  Instruction and speech input are separate.



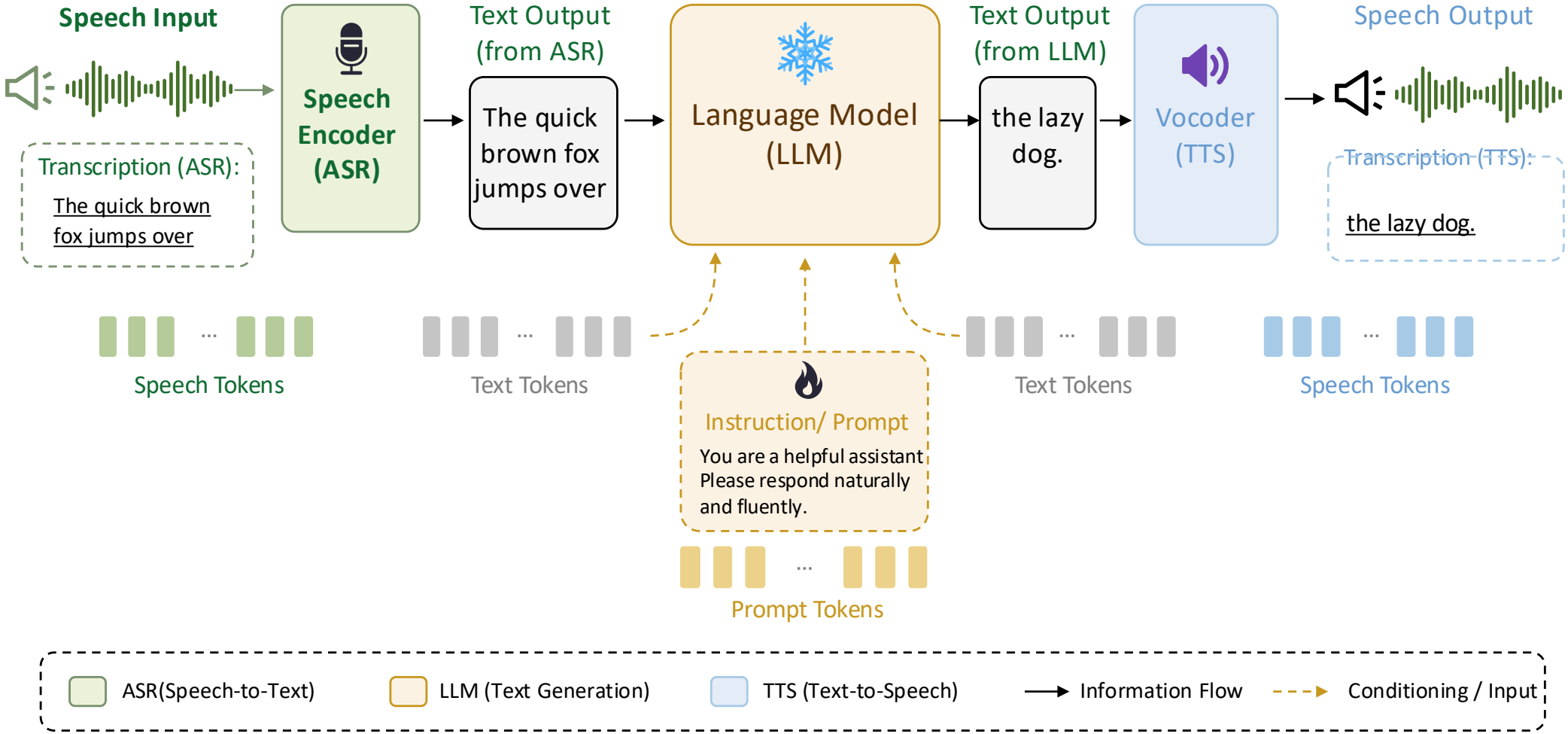
Different Types of Speech LLM

- Type 2: Dialogue Mode

  The speech LLM responds to input speech. The instruction is in the input speech.



What is Speech LLM

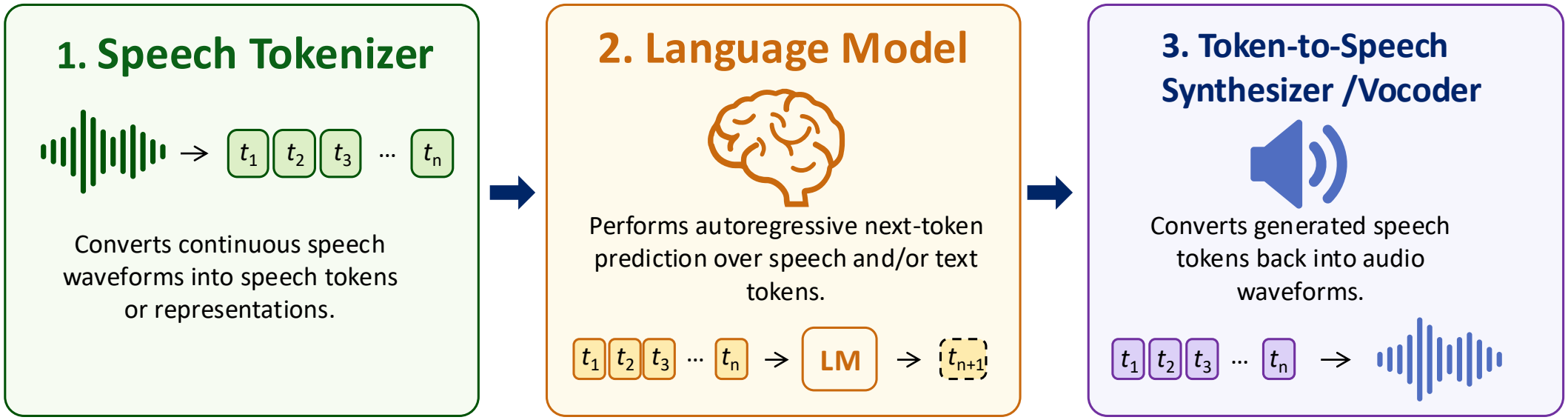


A **Speech Language Model (SpeechLM)** is an **autoregressive foundation model** that can process and generate speech **end-to-end**. SpeechLM provides a unified framework for modeling speech, text, and speech-text interleaved interactions.

Components in Speech LLM

- Speech Tokenizer
- Language Model
- Vocoder

What is Speech LLM



Speech LLMs adapt language modeling architectures to speech by adding modules that handle speech input and speech output.

Components in Speech LLM

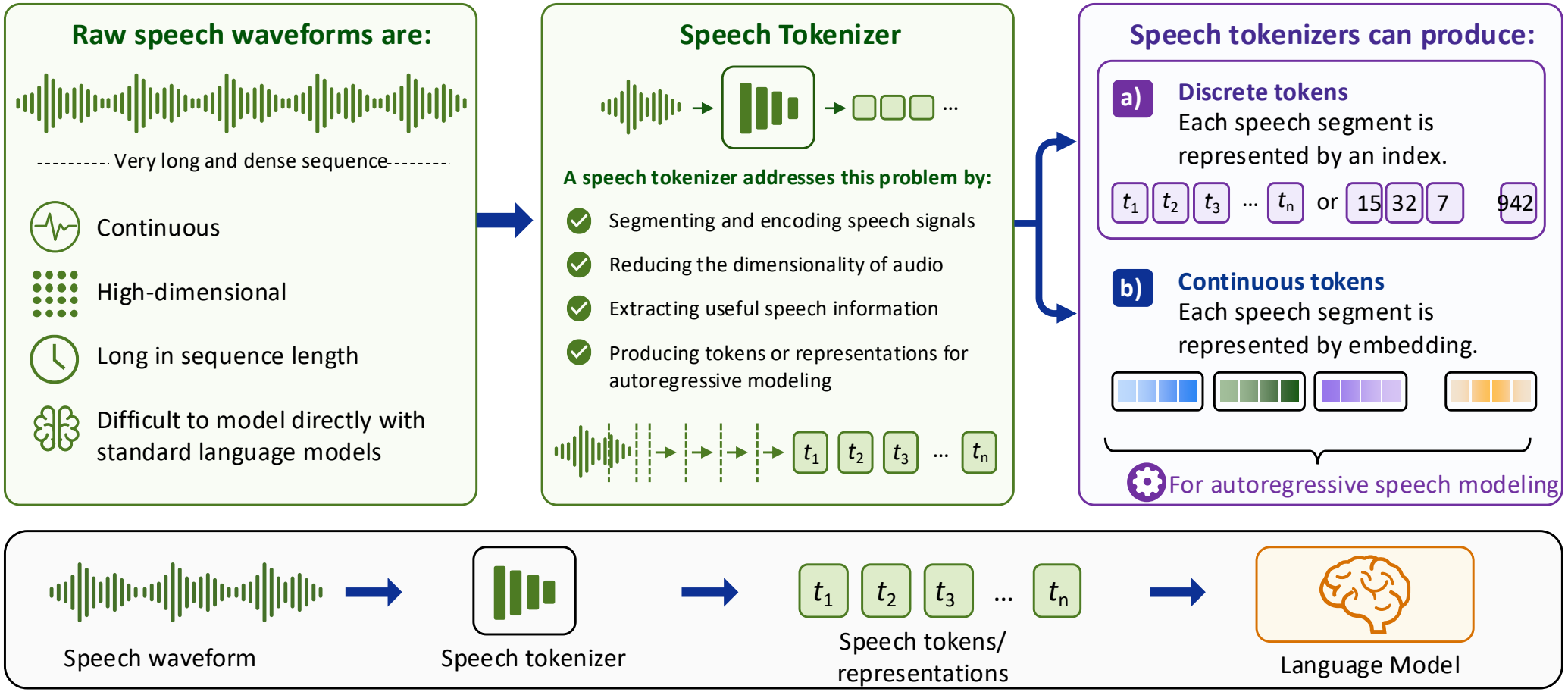
This table summarizes the architectural choices of mainstream SpeechLMs for speech tokenizers, language models, and vocoders.

Approach	Speech Tokenizer	Language Model	Vocoder
Kimi-Audio [78]	Whisper Encoder [12] + Linear Projector	Qwen2.5 [79]	BigVGAN [80]
Qwen2.5-Omni [81]	Whisper	Qwen2.5	Talker + Codec Decoder [81]
Minmo [82]	SenseVoice [83]	Qwen2.5	CosyVoice 2 [84]
Lyra [85]	Whisper [12]	Qwen2-VL [86]	HuBERT + HiFi-GAN
Flow-Omni [87]	Whisper Encoder + Linear Projector	Qwen2 [41]	Flow Matching (Transformer + MLP) + HiFi-GAN
SLAM-Omni [54]	Whisper Encoder + Linear Projector	Qwen2 [41]	-
OmniFlatten [53]	CosyVoice Encoder [88]	Qwen2	CosyVoice Decoder [88]
SyncLLM [89]	HuBERT [33]	LLaMA-3 [2]	HiFi-GAN [47], [48]
EMOVA [90]	SPIRAL [91]	LLaMA-3	VITS [92]
Freeze-Omni [67]	Transformer [38]	Qwen2	TiCodec [93]
IntrinsicVoice [94]	HuBERT	Qwen2	HiFi-GAN
Mini-Omni2 [66]	Whisper	Qwen2	Mini-Omni [10]
SALMONN-omni [71]	Mamba Streaming Encoder [95]	-	VoiceCraft [96] + Codec Decoder
Zeng et al. [97]	Whisper + VQ	GLM [42]	CosyVoice
NTPP [69]	VQ-VAE	LLaMA-3, Mistral, Gemma 2	HiFi-GAN
GPST [98]	EnCodec [49]	Transformer	Codec Decoder
GLM-4-Voice [55]	Whisper + VQ [9]	GLM-4-9B-Base [42]	CosyVoice
Moshi [9]	Mimi [9]	Transformer*	Mimi
VITA [70]	CNN + Transformer + MLP [70]	Mixtral [43]	Text-to-Speech Toolkit [70]
LSLM [65]	vq-wav2vec [31]	Decoder-Only Transformer	UniVATS [99]
SPIRIT-LM [5]	HuBERT, VQ-VAE [77], speechprop	LLaMA-2 [40]	HiFi-GAN
TWIST [51]	HuBERT	OPT [3], LLaMA [39]	HiFi-GAN
PSLM [100]	HuBERT	NekoMata [101]	HiFi-GAN
VOXTLM [102]	HuBERT	OPT [3]	HiFi-GAN
Voicebox [103]	EnCodec	Transformer* [38]	HiFi-GAN
Park et al. [104]	AV-HuBERT [105]	OPT	HiFi-GAN
USDM [106]	XLS-R [107]	Mistral	Voicebox [108]
VioLA [57]	EnCodec	Transformer*	Codec Decoder [49]
FunAudioLLM [83]	SAN-M [109]	Transformer*	HiFTNet [110]
SpeechGPT-Gen [60]	SpeechTokenizer [34]	LLaMA-2	SpeechTokenizer decoder [34]
ICoT [?]	SpeechTokenizer	LLaMA-2	SoundStorm
AnyGPT [111]	SpeechTokenizer	LLaMA-2	SoundStorm
LauraGPT [63]	Conformer*	Qwen [112]	Transformer + Codec Decoder
Spectron [61]	Conformer*	PaLM 2* [113]	WaveFit [114]
AudioLM [115]	w2v-BERT [32]	Decoder-Only Transformer*	SoundStream* [35]
UniAudio [116]	EnCodec, Hifi-codec [117], Improved RVQGAN [118]	Transformer*	Codec Decoder
Llama-Omni [11]	Whisper	LLaMA-3.1	HiFi-GAN
Mini-Omni [10]	Whisper + ASR Adapter [10]	Qwen2	TTS Adapter [10]
tGSLM [62]	Segmentation + SSE [119] + Lexical embedder	Transformer*	Tacotron-2 + Waveglow [45], [46]
SpeechGPT [8]	HuBERT	LLaMA	HiFi-GAN
dGSLM [4]	HuBERT	Dialogue Transformer [4]	HiFi-GAN
SUTLM [64]	HuBERT	Transformer*	-
pGSLM [56]	HuBERT	MS-TLM [56]	HiFi-GAN
GSLM [50]	HuBERT, CPC [29], Wav2vec 2.0 [30]	Transformer*	Tacotron-2 + Waveglow

Cui, W., Yu, D., Jiao, X., Meng, Z., Zhang, G., Wang, Q., Guo, S. Y., and King, I., "Recent Advances in Speech Language Models: A Survey," in Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vienna, Austria, 2025, pp. 13943–13970

Why Do We Need a Speech Tokenizer?

Speech must be converted into a form that language models can process

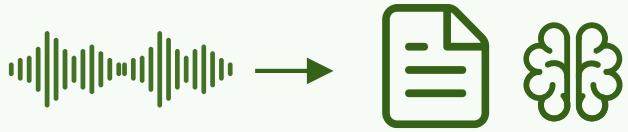


The speech tokenizer is the bridge between raw audio signals and language-model-style sequence modeling.

Three Types of Speech Tokenizers

Different tokenizers preserve different types of speech information

1. Semantic Tokenizer



- Focuses on speech content and meaning
- Designed to support semantic understanding tasks
- Useful for tasks such as ASR

Examples：

HuBERT | wav2vec 2.0 | WavLM

Content-oriented tokenization

Content / semantic tokens

2. Acoustic Tokenizer

- Focuses on acoustic details for nigh-quality speech reconstruction
- Preserves information needed for speech synthesis
- Often based on neural audio codecs

Examples:

SoundStream | EnCodec

Acoustically rich tokenization and reconstruction

Acoustic tokens

3. Mixed Tokenizer

- Balances semantic understanding and acoustic generation
- Aims to preserve both content and acoustic details

Examples:

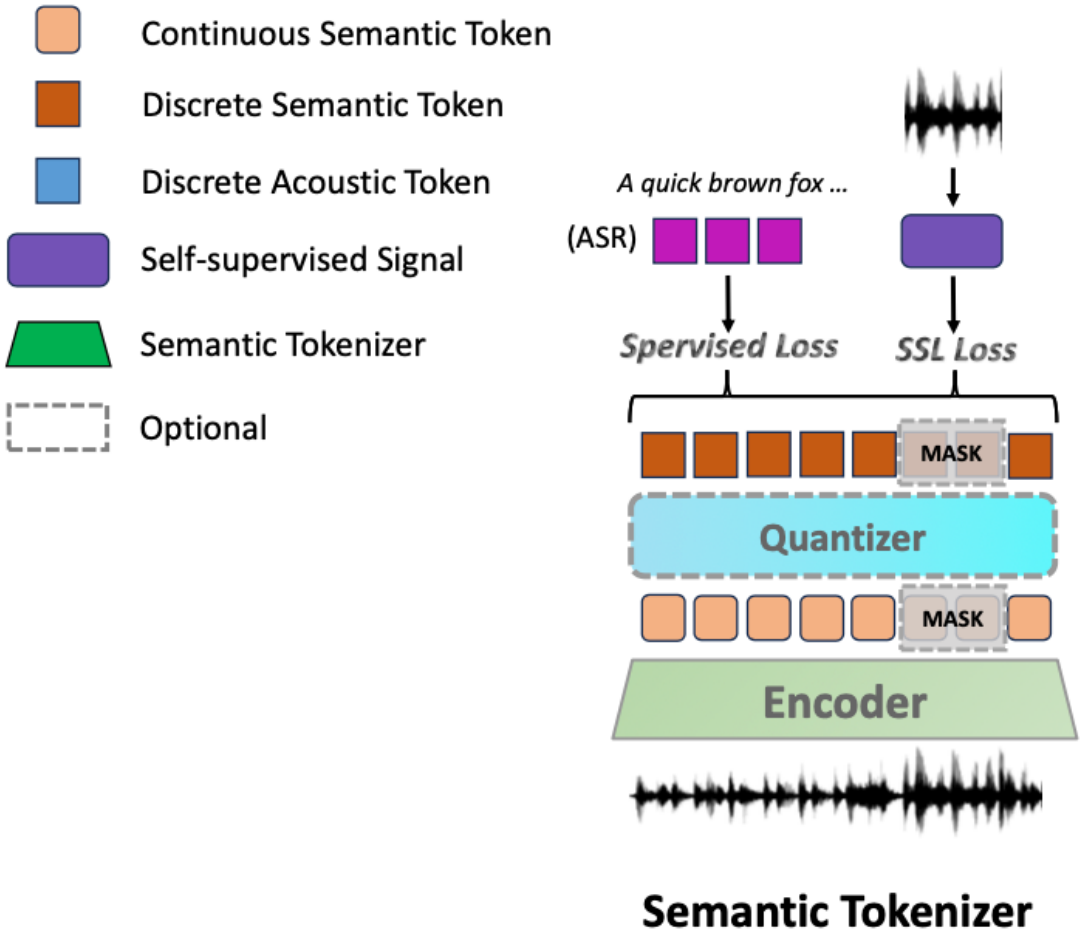
SpeechTokenizer | Mimi

Combined semantic + acoustic tokenization

Mixed tokens

Tokenizer design determines whether the model focuses more on meaning, acoustic quality, or both.

Semantic Tokenizer



Semantic Tokenizer-Wav2Vec

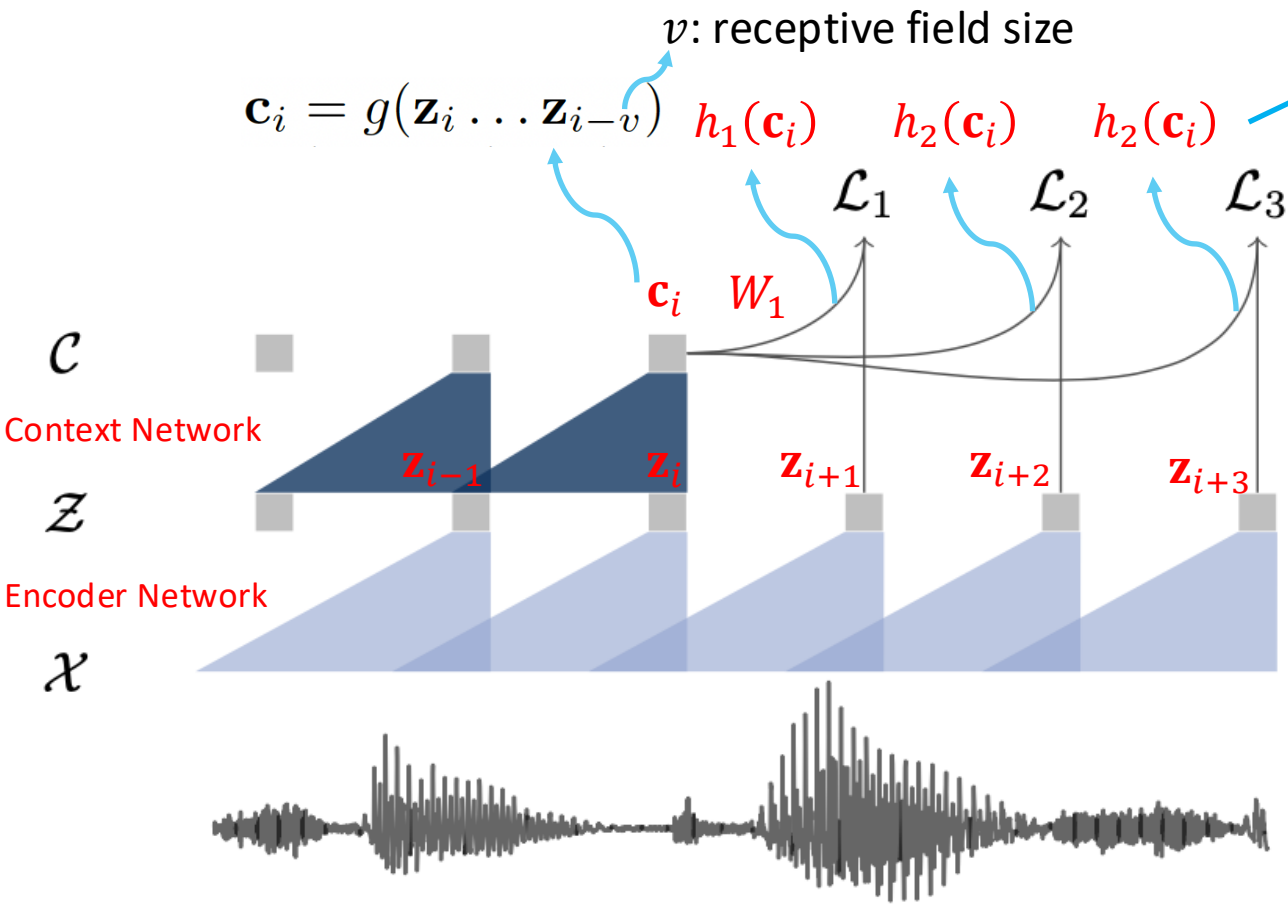


Fig . Illustration of pre-training from audio data X, which is encoded with two convolutional neural networks that are stacked on top of each other. The model is optimized to solve a next-time step prediction task.

$$h_k(\mathbf{c}_i) = W_k \mathbf{c}_i + \mathbf{b}_k$$

sigmoid $\sigma(x) = 1/(1+\exp(-x))$

Contrastive Loss:

$$\mathcal{L}_k = -\sum_{i=1}^{T-k} \left(\log \sigma(\mathbf{z}_{i+k}^\top h_k(\mathbf{c}_i)) + \lambda \mathop{\mathbb{E}}_{\tilde{\mathbf{z}} \sim p_n} [\log \sigma(-\tilde{\mathbf{z}}^\top h_k(\mathbf{c}_i))] \right)$$

Distractor sample

$P_n(\tilde{z}) = \frac{1}{T}$:Distribation of distractor, T: Squence length

Optimize the loss: $\mathcal{L} = \sum_{k=1}^{K} \mathcal{L}_k$

Schneider, S., et al. wav2vec: Unsupervised Pre-Training for Speech Recognition. *Proc. InterSpeech 2019*, 3465-3469.

Semantic Tokenizer-Wav2Vec 2.0

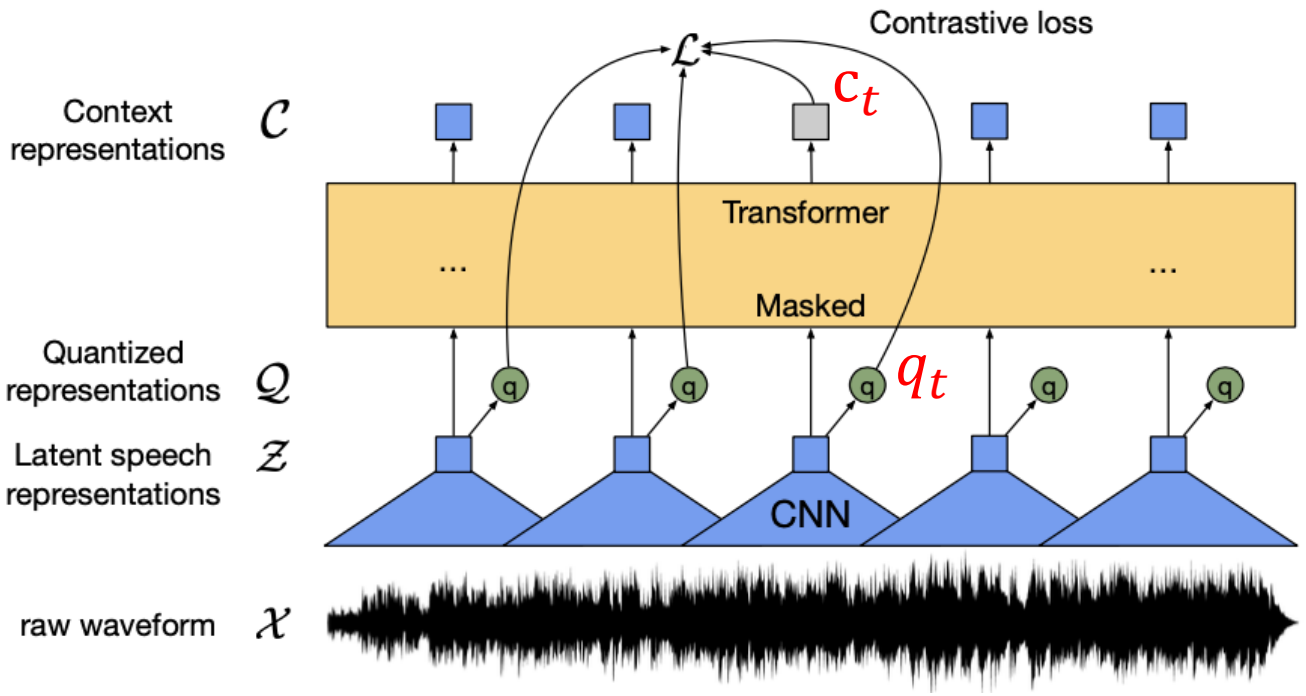


Fig . Illustration of Wav2Vec 2.0.

Objective Function : $\mathcal{L} = \mathcal{L}_m + \alpha \mathcal{L}_d$

Mask-based Contrastive Loss:

Context representation

Discrete latent speech representation

$$\mathcal{L}_m = -\log \frac{\exp(sim(\mathbf{c}_t, \mathbf{q}_t)/\kappa)}{\sum_{\tilde{\mathbf{q}} \sim \mathbf{Q}_t} \exp(sim(\mathbf{c}_t, \tilde{\mathbf{q}})/\kappa)}$$

Distractors $\tilde{q} \in Q_t$

Diversity Loss:

$$\mathcal{L}_d = \frac{1}{GV} \sum_{g=1}^{G} -H(\bar{p}_g) = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} \bar{p}_{g,v} \log \bar{p}_{g,v}$$

Entropy of $p_{g,v}$

$$p_{g,v} = \frac{\exp(l_{g,v} + n_v)/\tau}{\sum_{k=1}^{V} \exp(l_{g,k} + n_k)/\tau}$$

Baevski, A., et al. wav2vec 2.0: A framework for self-supervised learning of speech representations. Proc. *NeurIPS*

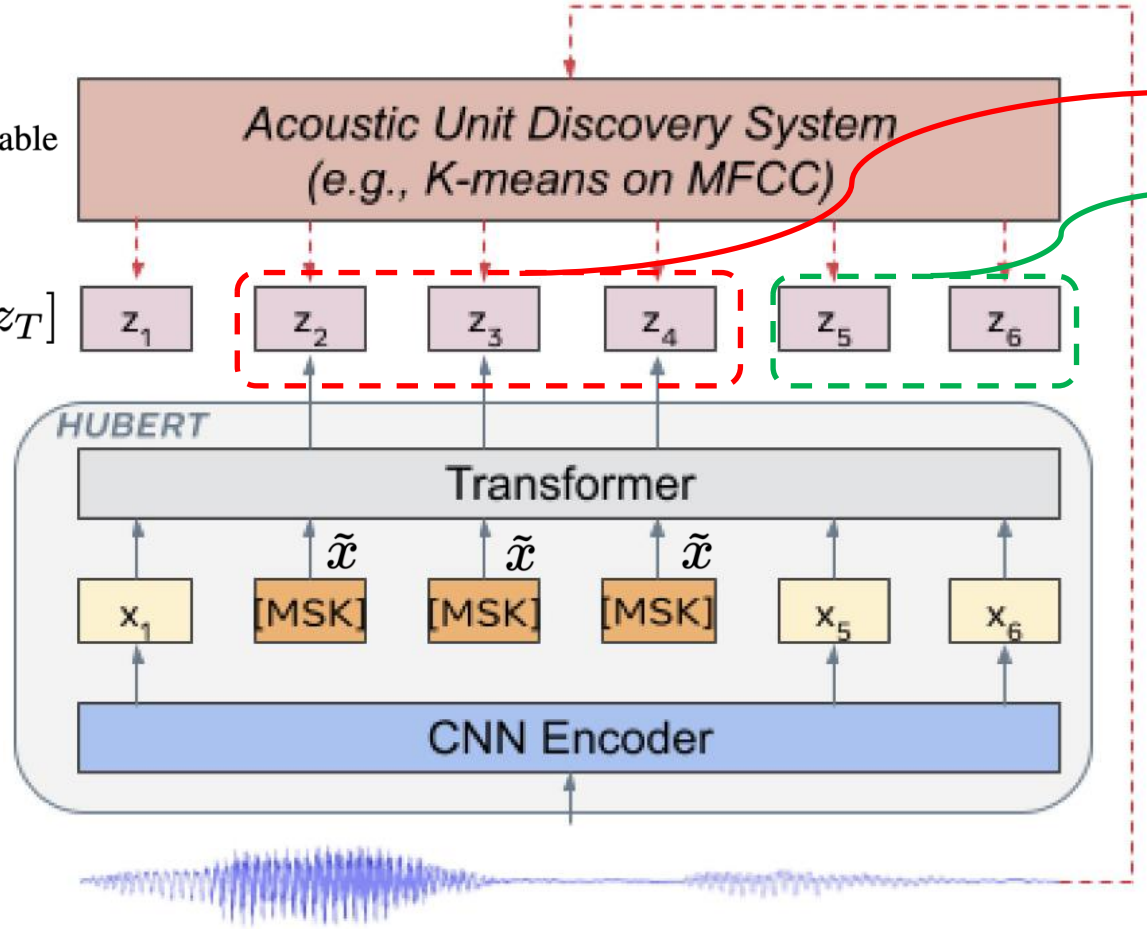
Semantic Tokenizer- HuBERT

$z_t \in [C]$ is a C-class categorical variable

$h(X) = Z = [z_1, \cdots, z_T]$

Clustering model

$X = [x_1, \cdots, x_T]$



$$L_m\left(f; X, \left\{Z^{(k)}\right\}_k, M\right) = \sum_{t \in M} \sum_k \log p_f^{(k)}\left(z_t^{(k)} \mid \tilde{X}, t\right)$$

$$L_u\left(f; X, \left\{Z^{(k)}\right\}_k, M\right) = \sum_{t \notin M} \sum_k \log p_f^{(k)}\left(z_t^{(k)} \mid \tilde{X}, t\right)$$

Codeword

$$p_f^{(k)}(c \mid \tilde{X}, t) = \frac{\exp(\mathrm{sim}(A^{(k)} o_t, e_c)/\tau)}{\sum_{c'=1}^{C} \exp(\mathrm{sim}(A^{(k)} o_t, e_{c'})/\tau)}$$

Projection Matrix

Output feature

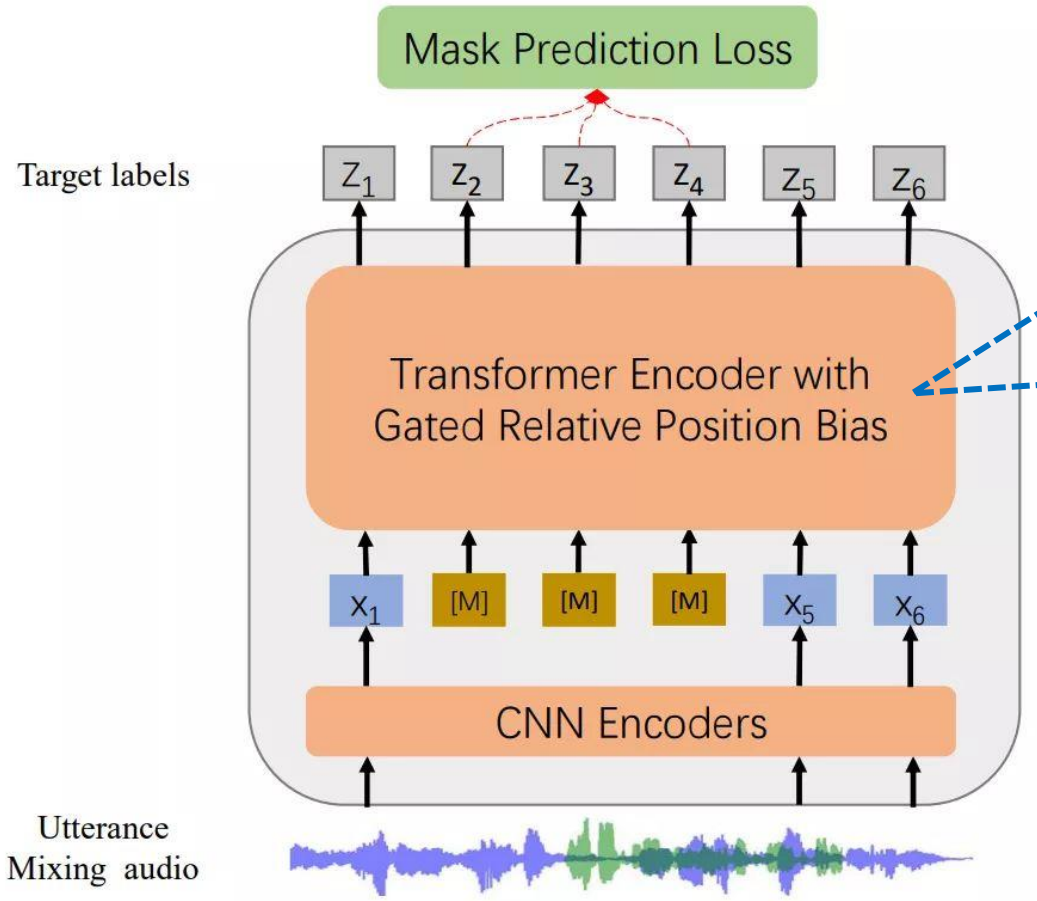
Optimize Loss :

$$L = \alpha L_m + (1 - \alpha) L_u$$

Fig . The HuBERT approach predicts hidden cluster assignments of the masked frames (y_2, y_3, y_4 in the figure) generated by one or more iterations of k-means clustering.

Hsu, Wei-Ning, et al. "HuBERT: Self-supervised speech representation learning by masked prediction of hidden units." *ICASSP 2021*.

Semantic Tokenizer- WavLM



Let $\{\mathbf{h}_i\}_{i=1}^{T}$ denote the input hidden states for the self-attention module. Each $\mathbf{h}_i$ is linearly projected to a triple of query, key, and value $(\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i)$ as:

$$\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i = \mathbf{h}_i \mathbf{W}^Q, \mathbf{h}_i \mathbf{W}^K, \mathbf{h}_i \mathbf{W}^V \quad (1)$$

The self-attention outputs $\left\{\tilde{\mathbf{h}}_i\right\}_{i=1}^{T}$ are computed via:

$$a_{ij} \propto \exp\left\{\frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}} + r_{i-j}\right\}$$

$$\tilde{\mathbf{h}}_i = \sum_{j=1}^{T} a_{ij} \mathbf{v}_j \quad (2)$$

Enhancing robustness in noisy or multi-speaker scenarios by capturing long-range dependencies and local speech patterns.

- The algorithm has been enhanced with the ability to remove noise.
- The pre-training dataset has been expanded from the 60k hours to 94k hours.

Whisper as a Semantic Speech Tokenizer

Large-scale weakly supervised sequence-to-sequence learning for language-aware speech representations

1 Large-scale weak supervision

Multitask training data (680k hours)

English transcription
Audio→English text

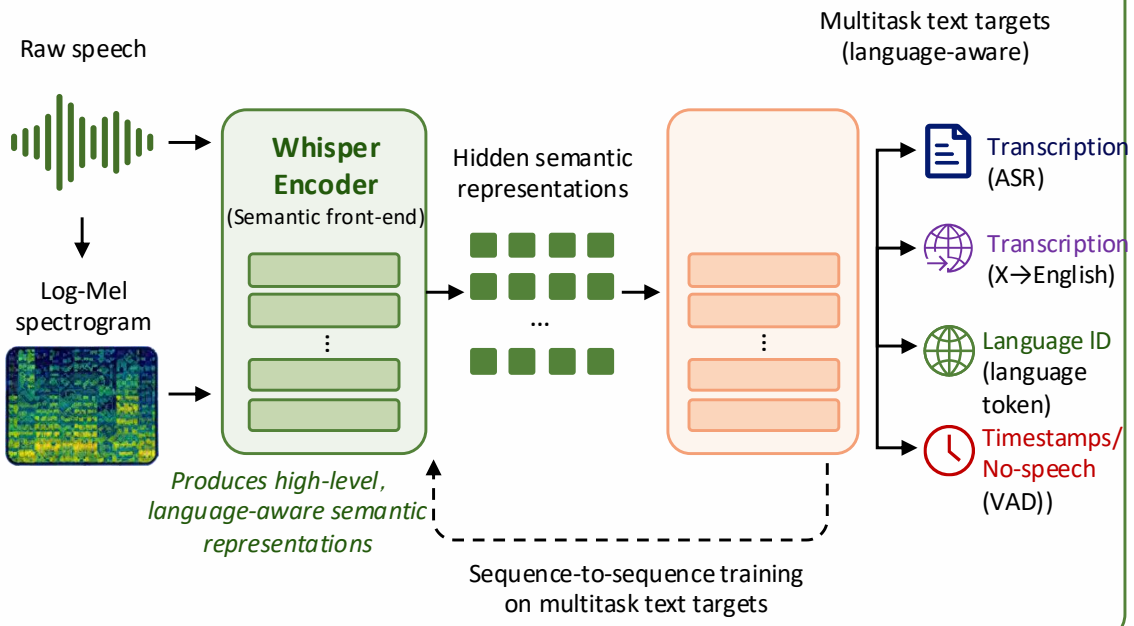
X→English speech translation
Audio (any language) →English text

Multilingual transcription
96 non-English languages
117k hours

No-speech/VAD
Voice activity detection

125k hours of X→English translation data

2 Whisper sequence-to-sequence model



3 Why this makes Whisper semantic

- Predicts transcripts / task tokens rather than reconstructing waveforms
- Learns language-aware, content-centric representations
- Joint multilingual + multitask supervision improves abstraction and transfer
- Strong zero-shot generalization without task-specific fine-tuning

Whisper maps raw speech to high-level semantic representations that are closer to linguistic content than to raw acoustics, making it a strong semantic speech tokenizer / encoder.

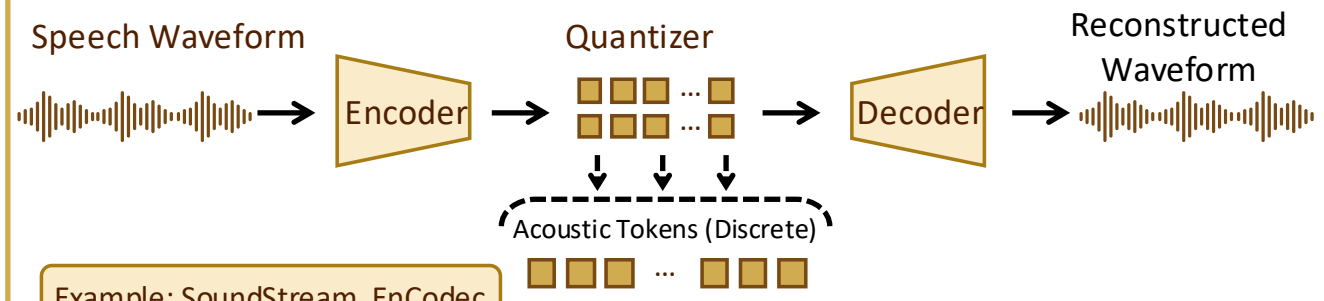
Supports transcription, translation, understanding, and other language tasks

Acoustic Tokenizers

2 EnCodec:
Acoustic Objective

Focus:Acoustic Quality

- Uses an encoder-quantizer decoder architecture
- Compresses speech into discrete acoustic tokens
- Preserves high-fidelity audio information for reconstruction



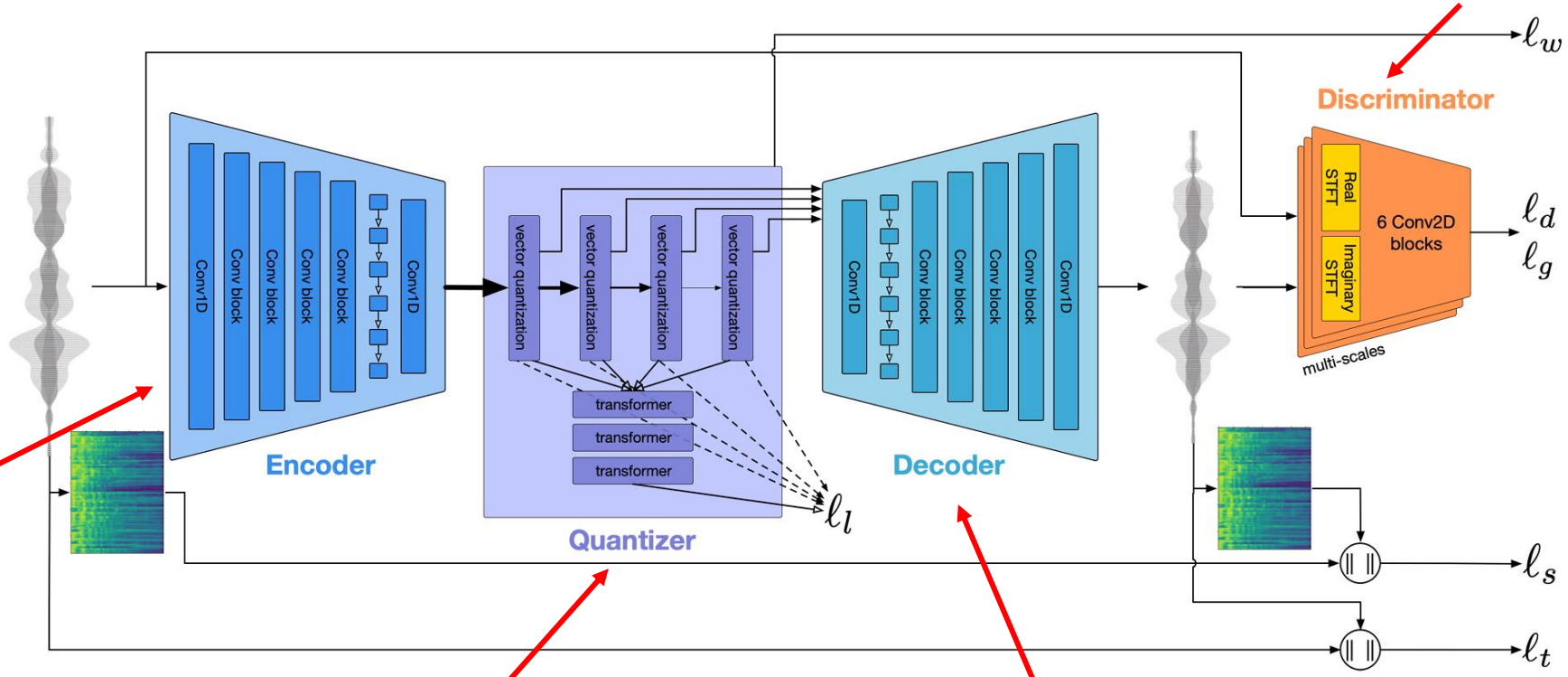
Example: SoundStream, EnCodec

Acoustic Tokenizers-Encodec

Why it is acoustic

EnCodec converts waveforms into discrete acoustic tokens and reconstructs high-quality audio, preserving how speech sounds rather than only what it means.

Discriminator : improve perceptual quality and reduce reconstruction artifacts.



Encoder: maps waveform into latent acoustic representations.

Quantizer: uses residual vector quantization to produce discrete acoustic codes.

Decoder: reconstructs the waveform from quantized codes.

A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High Fidelity Neural Audio Compression," *Transactions on Machine Learning Research*, vol. 2023.

Acoustic Tokenizers

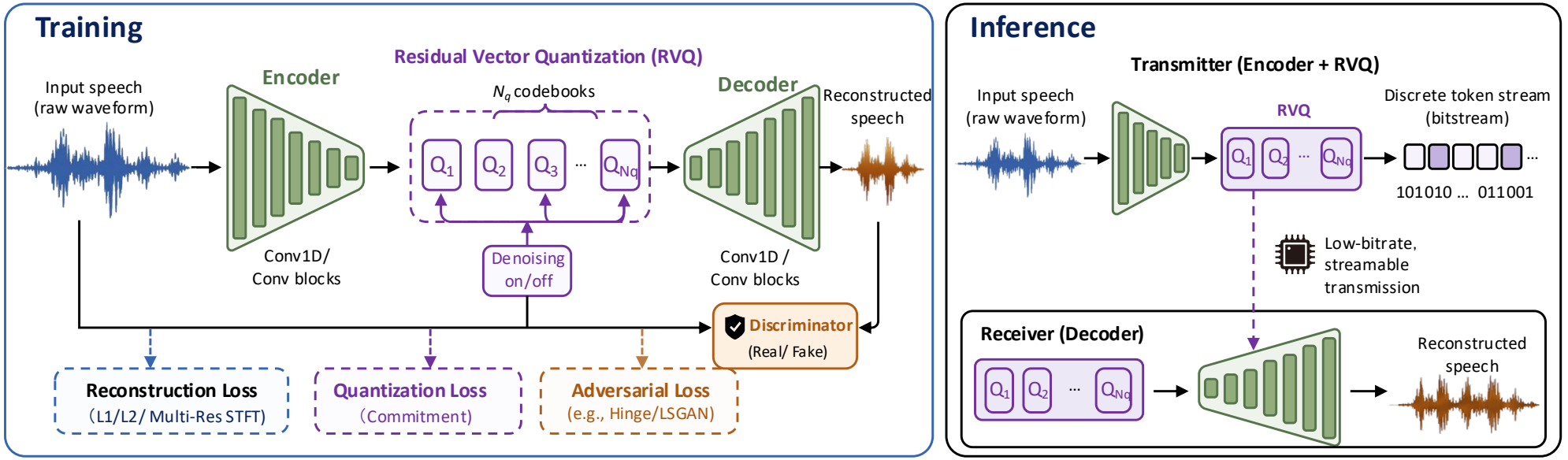
SoundStream

SoundStream as an Acoustic Tokenizer

An end-to-end neural audio codec for low-bitrate, high-quality speech representation

3-18 kbps variable bitrate

Real-time/ Streamable



What SoundStream does

- Encodes waveform into compact latent acoustic representations.
- Uses residual vector quantization to produce discrete acoustic tokens.
- Reconstructs high-quality speech/audio from compressed tokens.
- Supports joint compression and enhancement.

Why it matters for SpeechLMs

- A representative acoustic tokenizer /neural audio codec.
- Preserves timbre, prosody, and fine acoustic detail.
- Better for reconstruction and speech generation than semantic modeling.
- Provides discrete acoustic units for downstream audio language modeling.

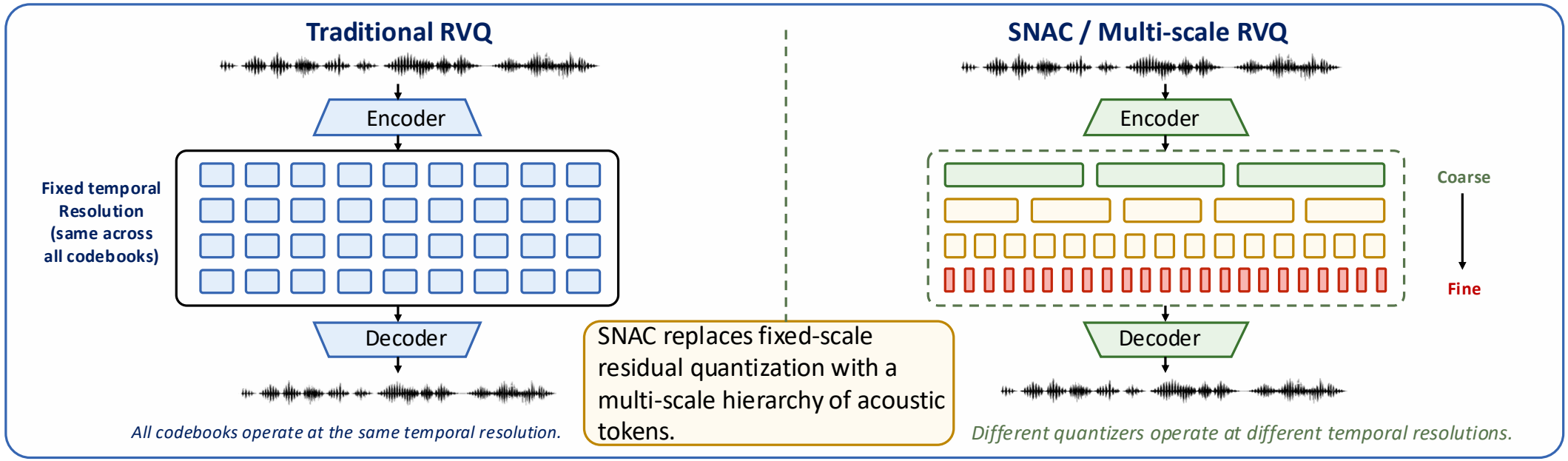
SoundStream maps raw speech into discrete acoustic codes optimized for efficient compression and high-fidelity reconstruction, making it a foundational acoustic tokenizer.

N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "SoundStream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2022.

Acoustic Tokenizers

SNAC: Multi-scale Neural Audio Codec

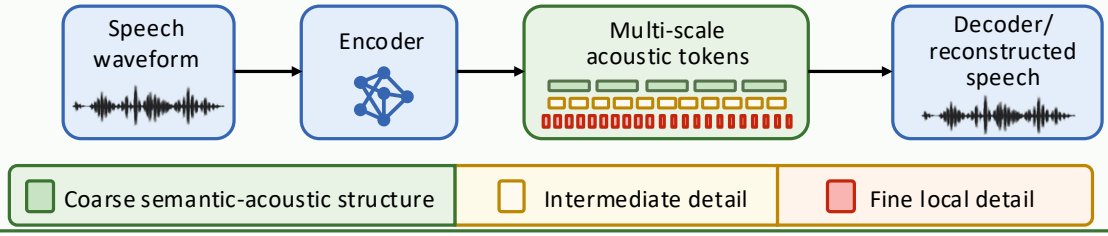
Hierarchical acoustic tokenization from coarse-to-fine temporal resolutions



Why SNAC matters

- Captures both global structure and fine acoustic detail
- Uses token hierarchy more efficiently than fixed-scale RVQ
- Produces acoustic tokens for high-quality reconstruction
- Better aligned with multi-scale temporal patterns in speech/audio

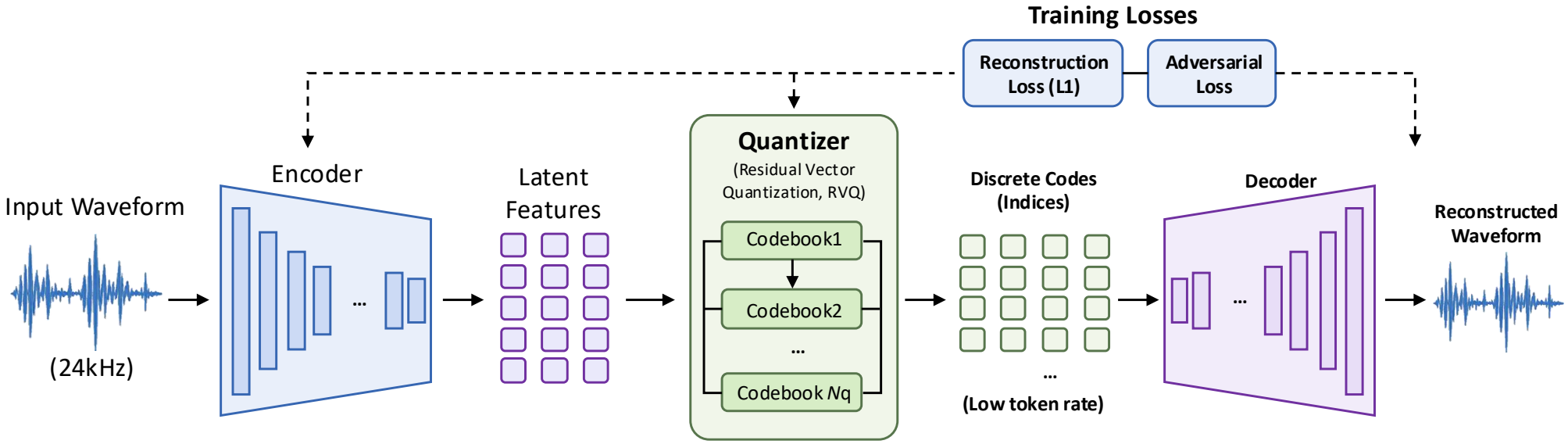
How to understand SNAC



SNAC organizes acoustic tokens across multiple time scales, enabling more efficient and expressive speech/audio reconstruction than traditional RVQ.

H. Siuzdak, F. Grötschla, and L. A. Lanzendörfer, "SNAC: Multi-scale neural audio codec," in *Audio Imagination: NeurIPS Workshop AI-Driven Speech, Music, and Sound Generation*, 2024.

Acoustic Tokenizers- WavTokenizer



1. **Encoder:** maps the raw waveform into compact latent acoustic features.
2. **RVQ Quantizer:** converts latent features into discrete acoustic codes using multiple codebooks.
3. **Low token rate:** produces compact acoustic tokens, making modeling more efficient for Audio LMs.
4. **Decoder + losses:** reconstructs, high-fidelity waveform, optimized with reconstruction and adversarial objectives.

✓ **Extremely low token rate** | ✓ **High-fidelity reconstruction** | ✓ **Efficient for Audio LMs**

WavTokenizer provides compact discrete acoustic tokens that preserve reconstruction quality while improving eficiency for audio language modeling

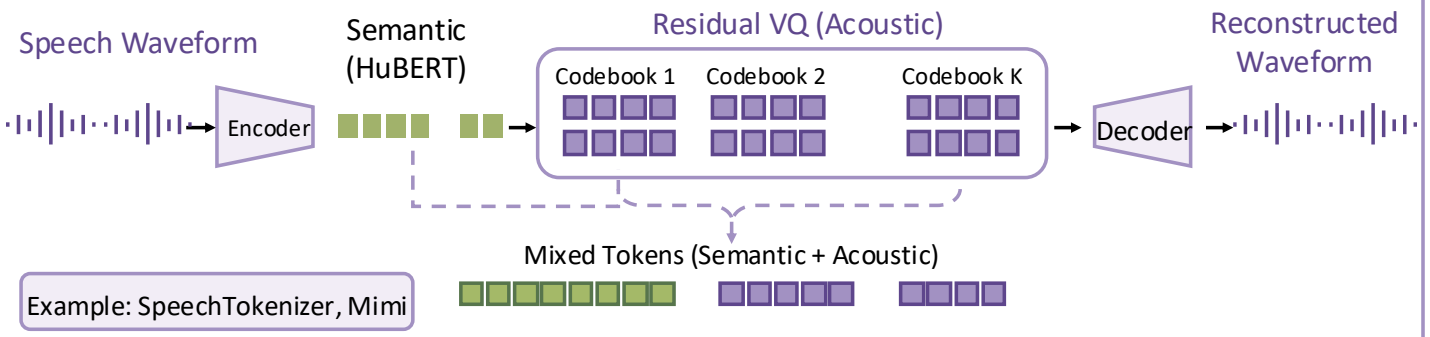
Shengpeng Ji, Ziyue Jiang, Wen Wang, et al., "WavTokenizer: An Efficient Acoustic Discrete Codec Tokenizer for Audio Language Modeling," Proc. ICLR, 2025.

Mixed Tokenizers

SpeechTokenizer: Mixed Objective

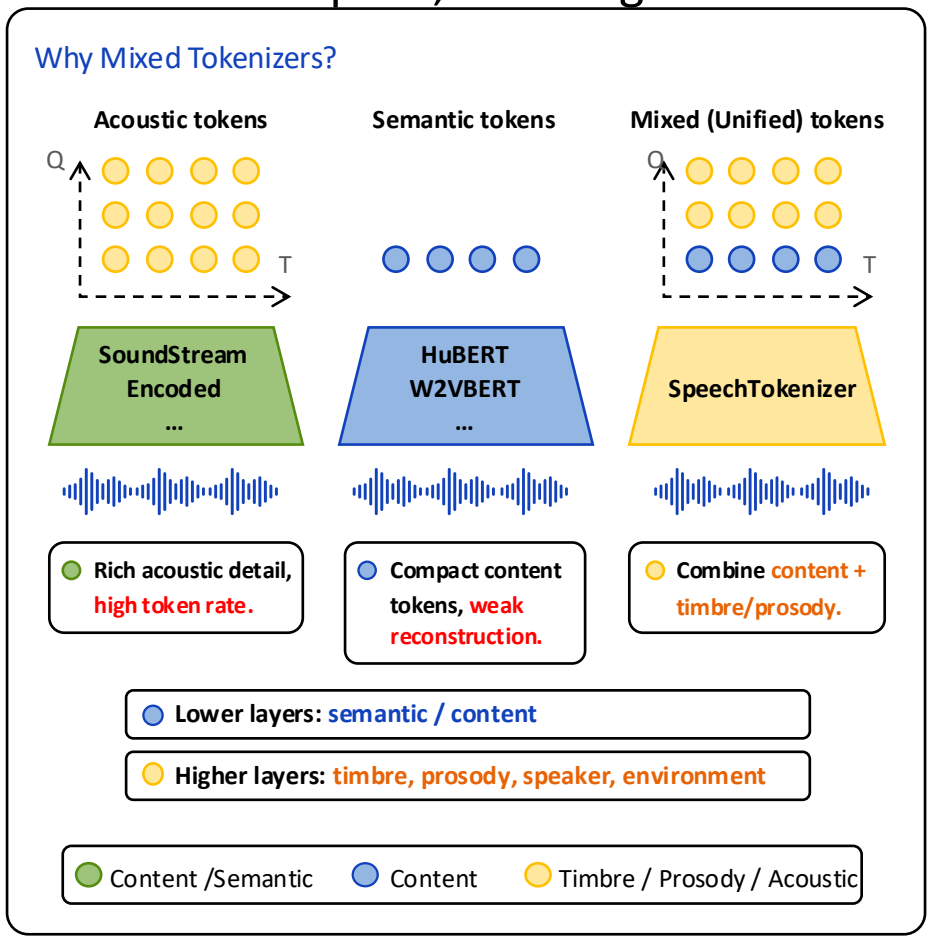
Focus: Both

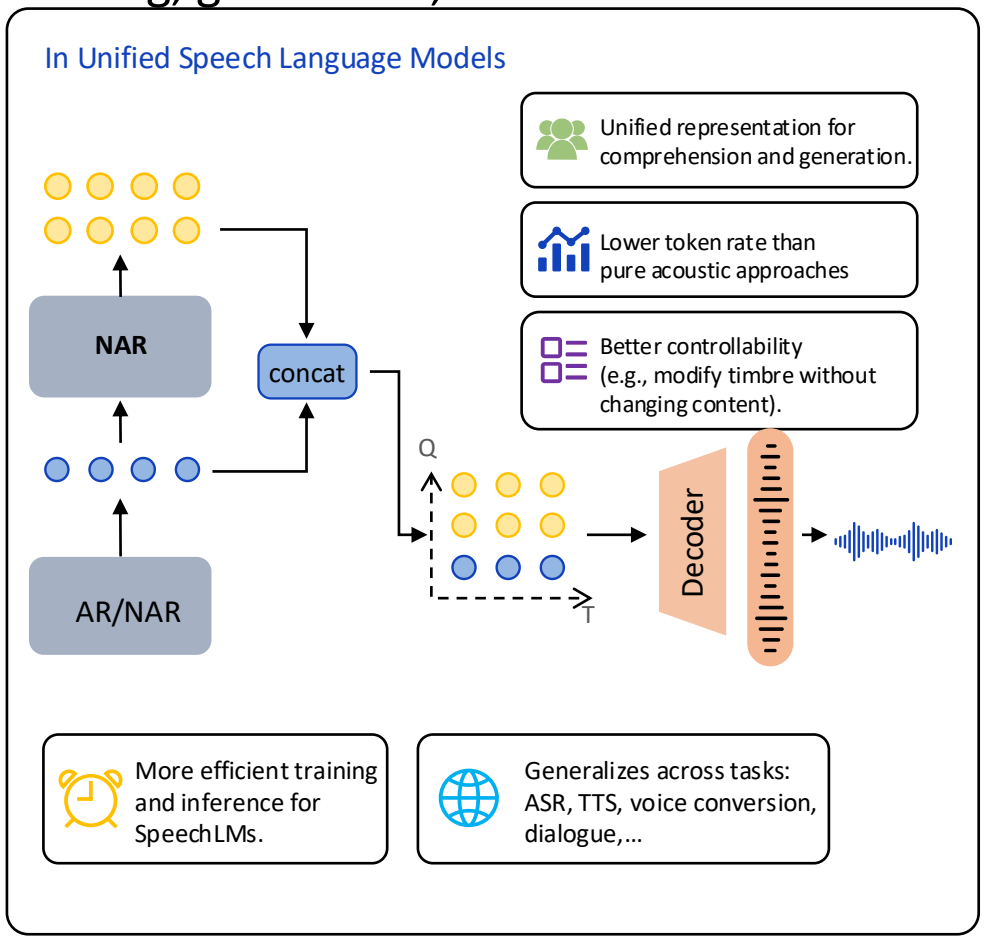
- Combines semantic and acoustic objectives
- Distills semantic information from HuBERT
- Uses residual vector quantization to capture acoustic details



Mixed Tokenizers-SpeechTokenizer

Mixed tokenizers aim to represent both the **meaning (semantic/content)** and the **delivery (acoustic/paralinguistic)** of speech in a unified discrete space, enabling better understanding, generation, and interaction.





Mixed tokenizers bridge 'what is said ' and 'how it is said', enabling more natural, efficient, and controllable Speech LLMs

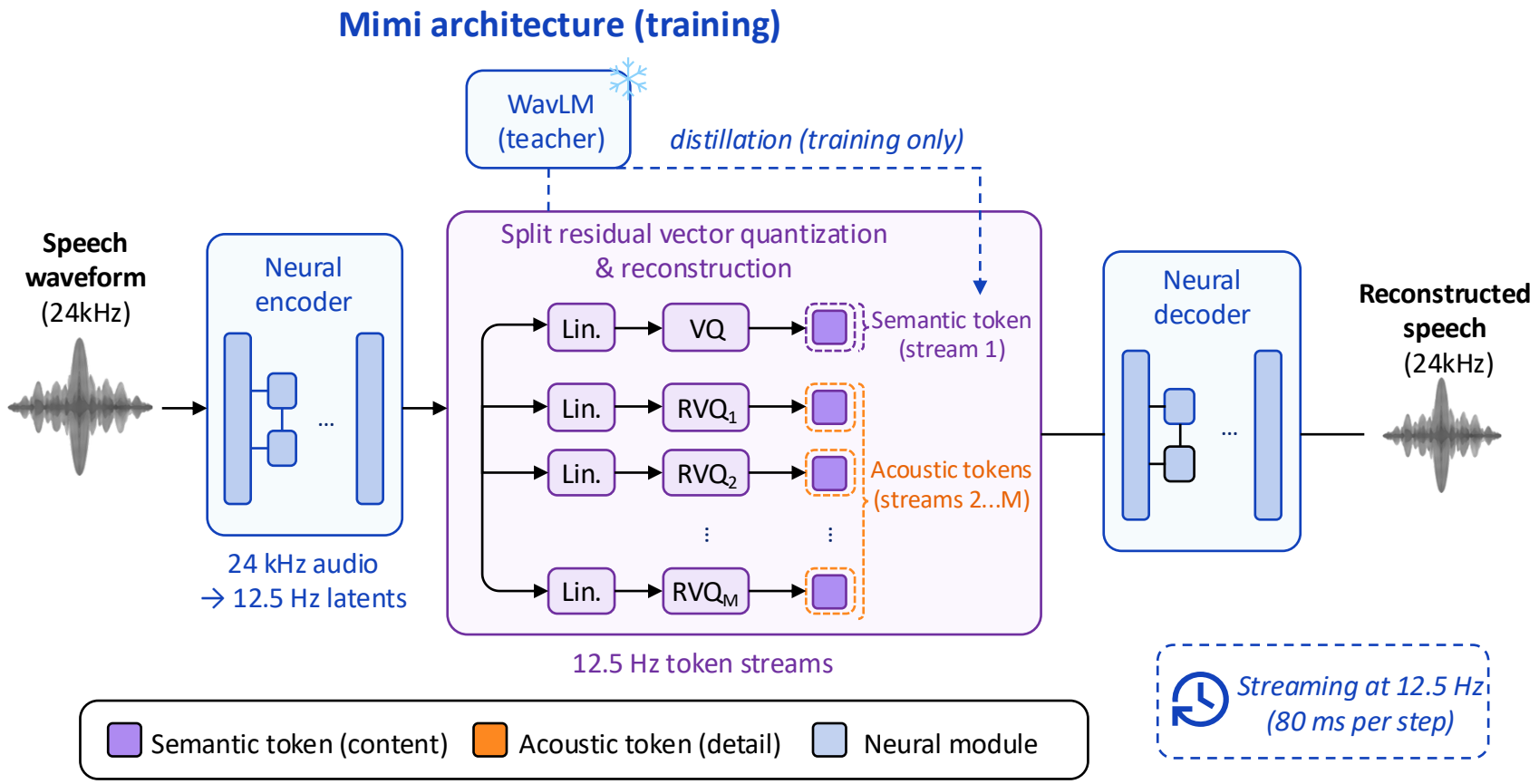
X. Zhang, D. Zhang, S. Li, et al., "SpeechTokenizer: Unified speech tokenizer for speech language models," in Proc. ICLR, 2024.

Mixed Tokenizers-Mimi

Why mixed tokenization?

- Pure semantic tokens capture content but miss fine acoustic detail.
- Pure acoustic tokens reconstruct well but are inefficient for language modeling.
- Mimi combines both in one tokenizer for speech understanding and generation.

- One semantic code stream models linguistic content
- Additional acoustic streams preserve timbre, prosody, and reconstruction detail.
- The mixed representation is low-rate and streaming-friendly



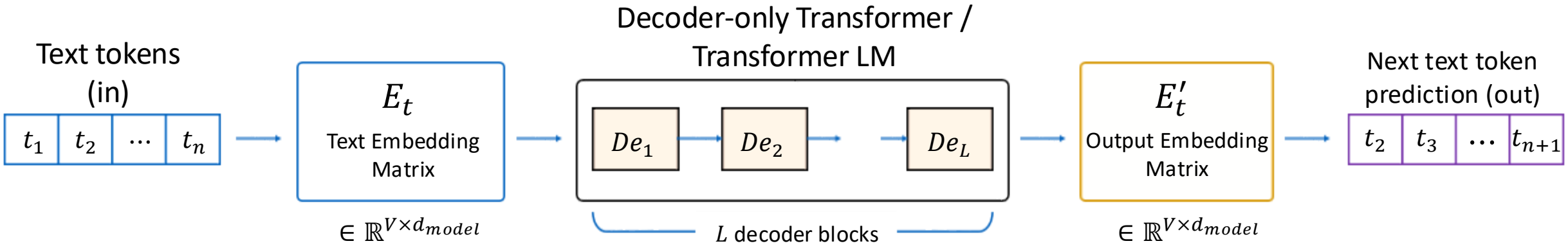
Mimi acts as a mixed tokenizer: it bridges *what is said* and *how it is said*, making speech tokens more useful for real-time Speech LLMs.

A. Défossez, L. Mazaré, M. Orsini, A. Royer, P. Pérez, H. Jégou, E. Grave, and N. Zeghidour, "Moshi: A speech-text foundation model for real-time dialogue," arXiv preprint arXiv:2410.00037, 2024

LM in Speech LLMs: Text LLM Backbone

Speech LLMs usually inherit the architecture of text-based language models

Transformer | Decoder-only | OPT | LLaMA



$$t^{out} \approx LM\left(t^{in}, (E_t, De, E'_t)\right)$$

Backbone
Most SpeechLMs adopt architectures from successful TextLMs.

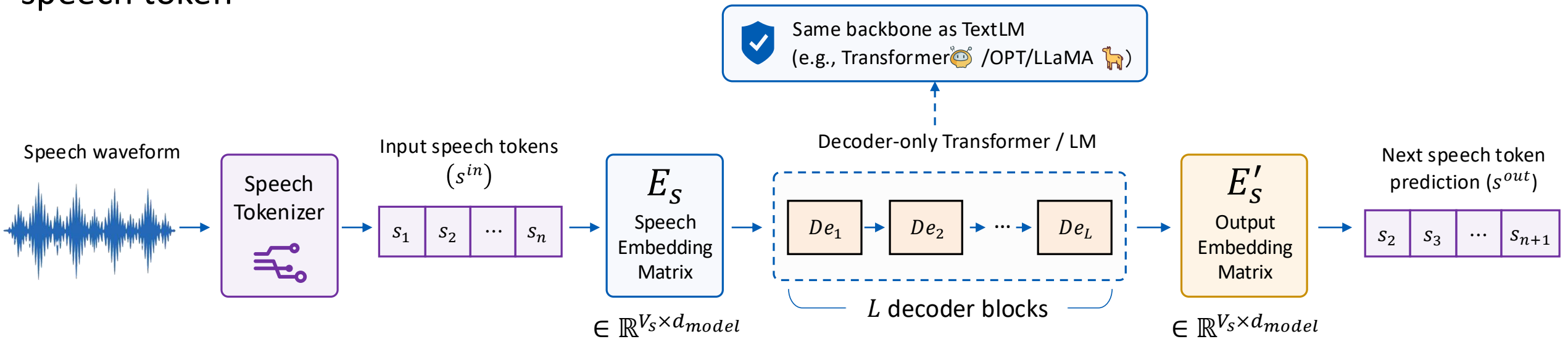
Core components
Embedding matrix, decoder blocks, output embedding.

Autoregressive prediction
The model predicts the next token step by step.

Before modeling speech, Speech LLMs first inherit the language-model backbone from Text LLMs.

LM in Speech LLMs: From Text Tokens to Speech Tokens

Speech LLMs keep the language-model backbone but replace text token embeddings with speech token



$$s^{out} \approx LM\left(s^{in}, (E_s, De, E'_s)\right)$$

New input/output space
Text tokens are replaced by speech tokens.

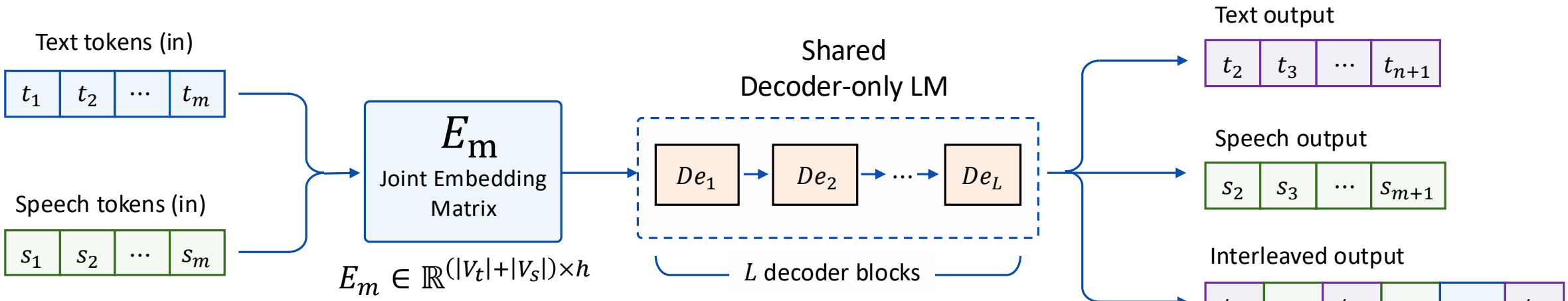
Same decoder blocks
The Transformer backbone remains unchanged.

Autoregressive speech modeling
The model predicts the next speech token step by step.

Adapting a Text LLM to a Speech LLM mainly means changing the token interface from text embeddings to speech embeddings.

LM in Speech LLMs: Unified Speech-Text Modeling

By expanding the vocabulary, one LM can model text tokens and speech tokens in a single sequence



$$m^{out} \approx LM\left(m^{in}, (E_m, De, E'_m)\right)$$

Vocabulary expansion
Append speech tokens to the original text vocabulary.

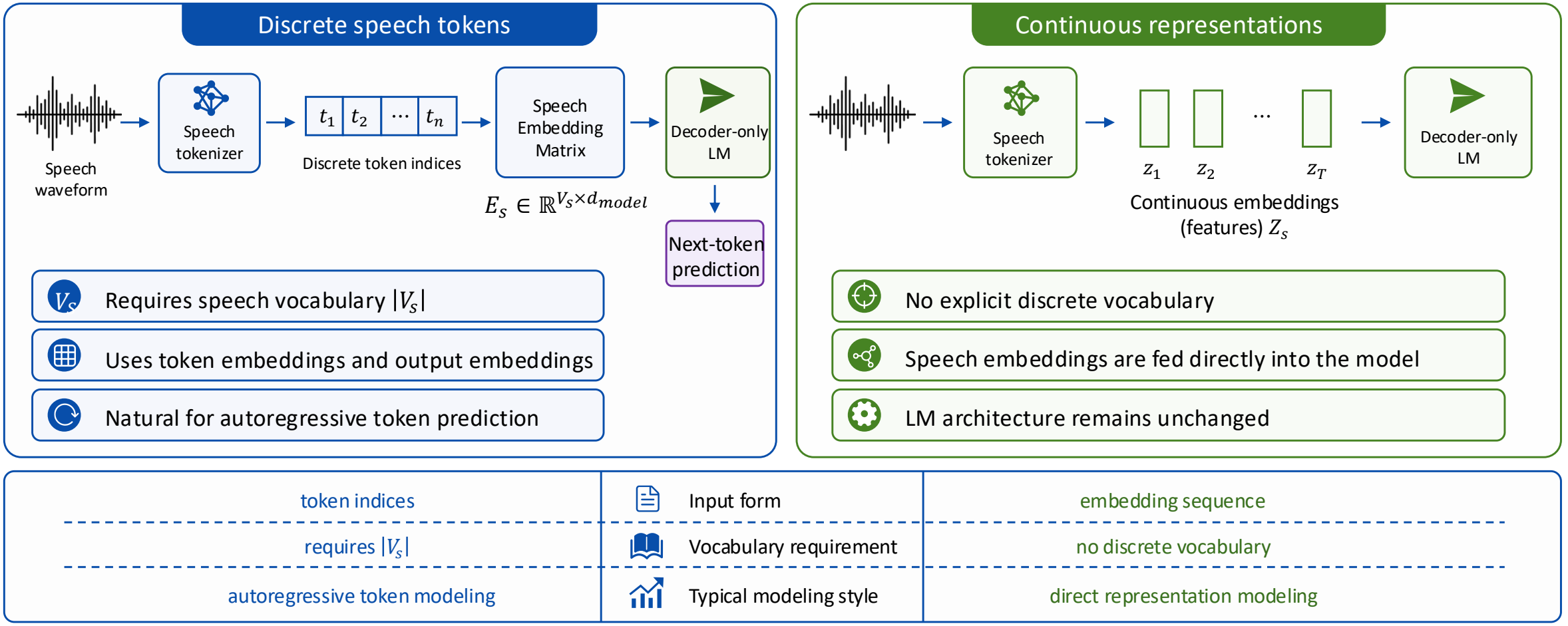
Mixed sequence modeling
The input sequence may contain both speech and text tokens.

Diverse applications
Supports ASR, TTS, speech translation, and interleaved interaction.

A unified token space allows one language model to jointly generate and understand both speech and text

LM in Speech LMs: Discrete vs Continuous Interfaces

Speech tokenizers can connect to the language model through discrete tokens or continuous representations

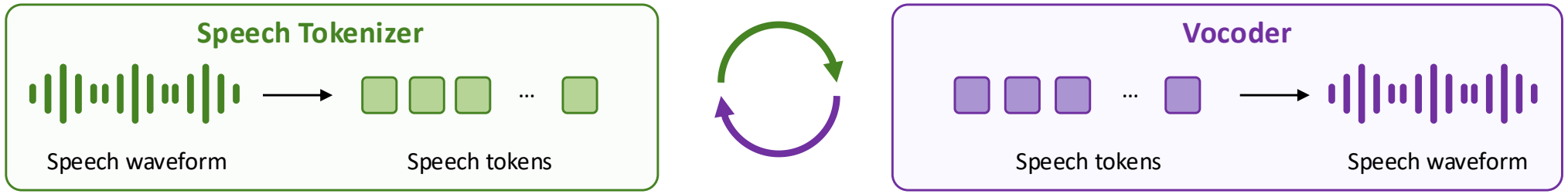


token indices	Input form	embedding sequence
requires $\|V_s\|$	Vocabulary requirement	no discrete vocabulary
autoregressive token modeling	Typical modeling style	direct representation modeling

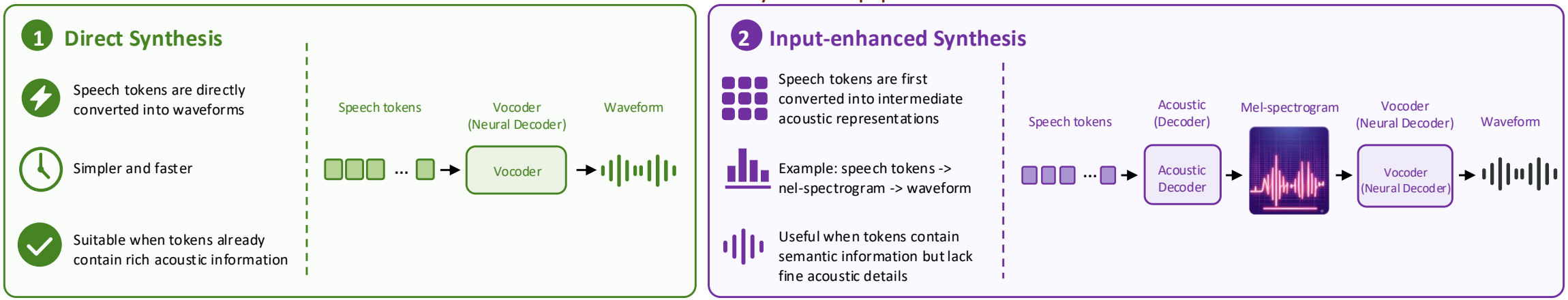
The main difference is the interface to the LM-discrete modeling uses a speech vocabulary, while continuous modeling feeds speech embeddings directly.

Token-to-Speech Synthesizer (Vocoder)

Converting generated speech tokens back into audible speech waveforms.



Two common synthesis pipelines



The vocoder determines how natural, clear, and efficient the final speech output will be.

Vocoder in Speech LLM : From Tokens Back to Speech

The vocoder converts generated speech tokens back into audible speech waveforms

Reverse process of speech tokenization

Generated speech tokens s

Semantic information

Acoustic information

Vocoder/
Token-to-Speech Synthesizer

Audio waveform a

$$a = V_O(s; \theta_{V_O})$$

Reverse mapping

Speech tokens are mapped back to waveform space

Information recovery

Linguistic and paralinguistic cues are rendered into audible speech.

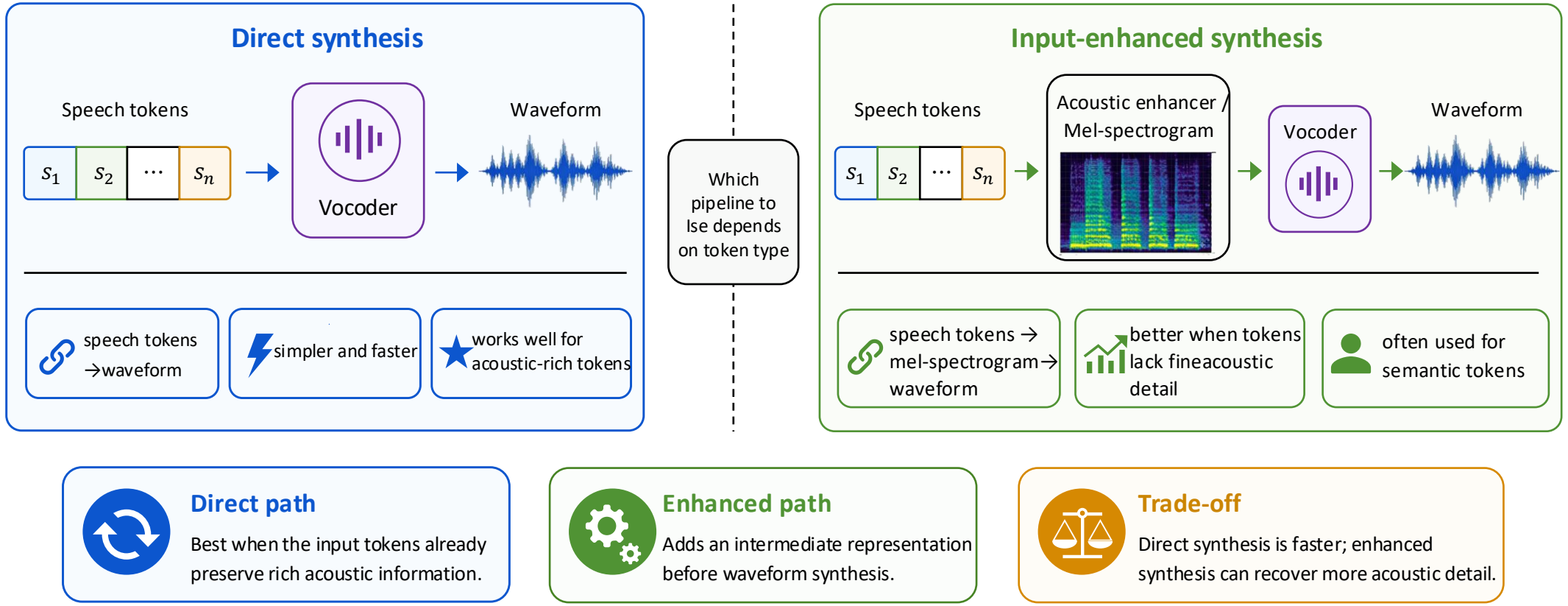
Final output

The vocoder determines naturalness and clarity.

After the language model generates speech tokens, the vocoder reconstructs the final speech signal.

Vocoder in Speech LLM: Two Synthesis Pipelines

Speech tokens can be converted to waveforms either directly or through an intermediate acoustic representation



Direct path
Best when the input tokens already preserve rich acoustic information.

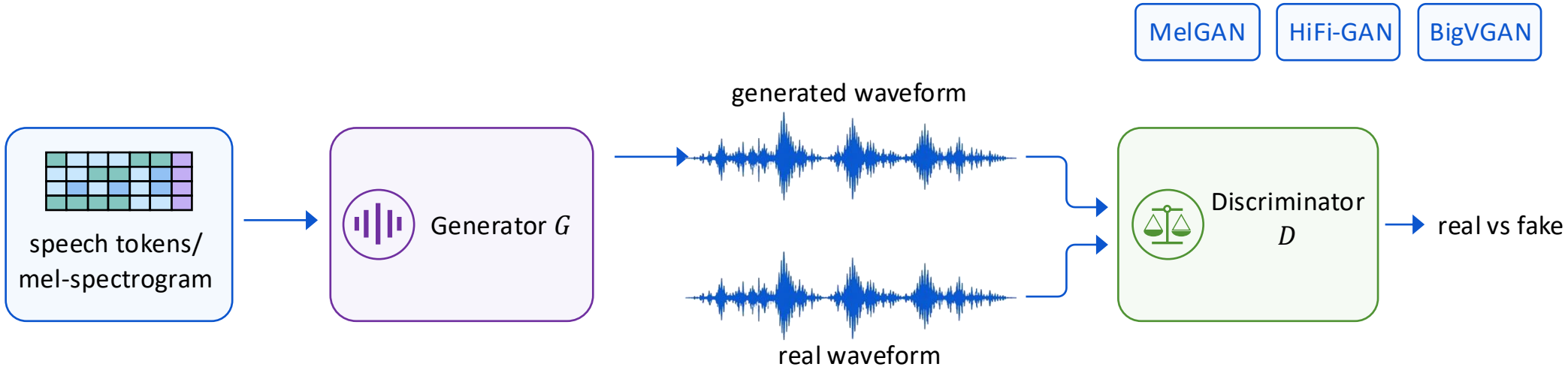
Enhanced path
Adds an intermediate representation before waveform synthesis.

Trade-off
Direct synthesis is faster; enhanced synthesis can recover more acoustic detail.

Aoustic tokens often support direct synthesis, while semantic tokens benefit from an input-enhanced pipeline

Vocoder in Speech LLM: GAN-based Vocoder

GAN vocoders synthesize high-fidelity speech using a generator-discriminator framework



$$a = G(s; \theta_G)$$

Generator
Maps input speech features to an output waveform.

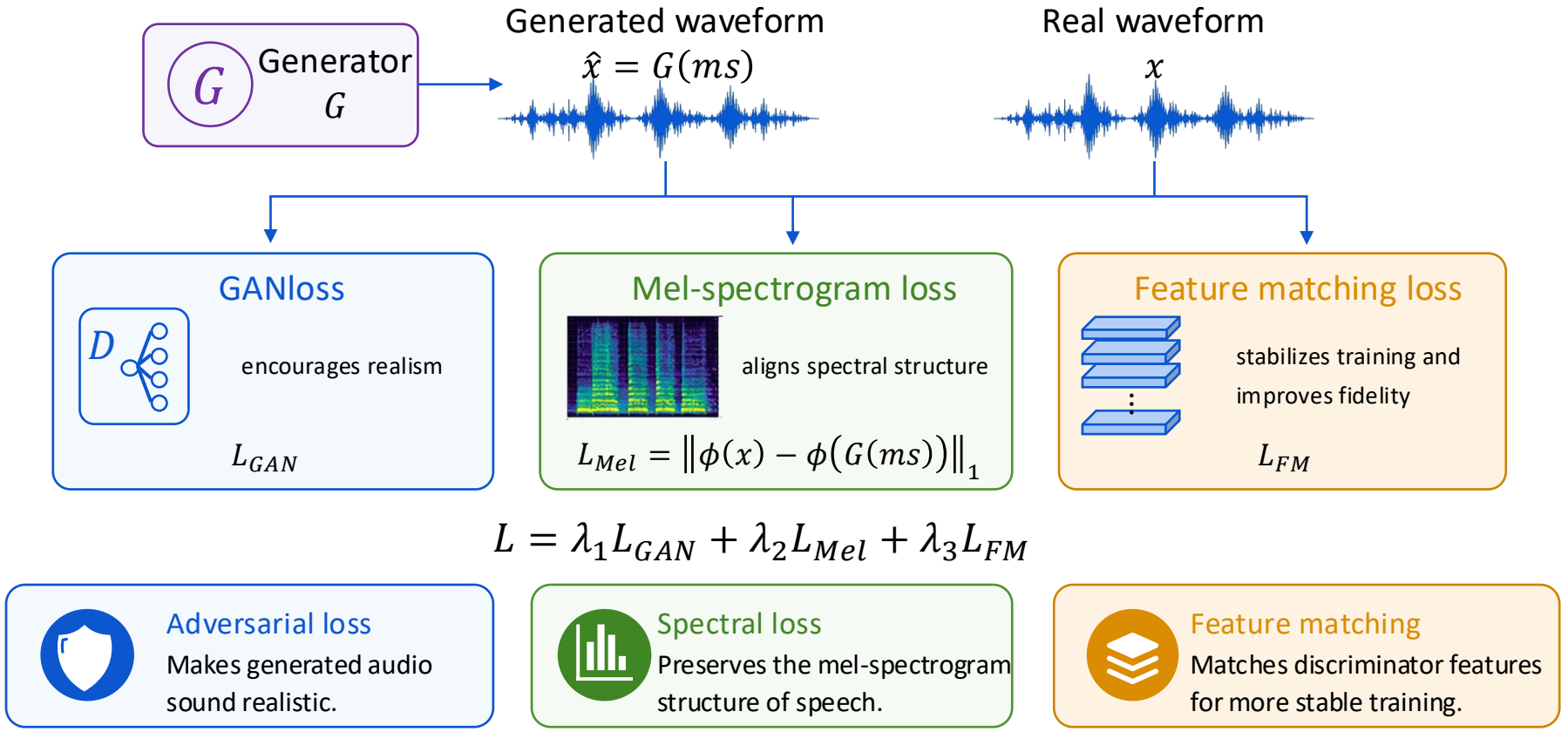
Discriminator
Encourages realism by distinguishing real and generated audio.

Why GAN?
Fast parallel synthesis with strong perceptual quality.

GAN-based vocoders are widely adopted in Speech LLMs because they generate high-fidelity speech efficiently.

Vocoder in Speech LLM: Training Objectives

GAN-based vocoders are usually optimized by combining adversarial, spectral, and feature-level losses

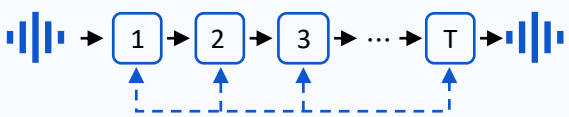


GAN loss improves realism, mel loss preserves spectral fidelity, and feature matching stabilizes training.

Vocoder in Speech LLM: Other Vocoder Families

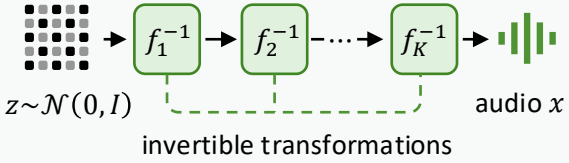
Beyond GAN vocoders, other generative families include autoregressive, flow-based, and VAE-based models

Autoregressive vocoder



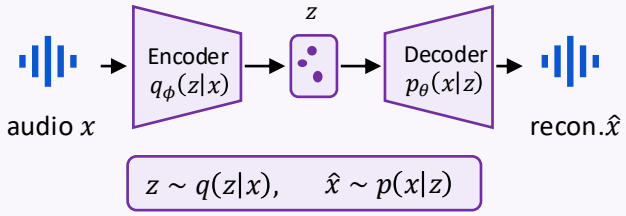
- generates one sample at a time
- captures temporal dependencies
- high quality but slow

Flow-based vocoder



- maps simple noise to audio
- parallel generation
- more memory and parameters

VAE-based vocoder



- learns a compact latent space
- supports generation and reconstruction
- less common in SpeechLM pipelines

Quality: autoregressive ≥ flow-based ≥ VAE (often, but task-dependent)

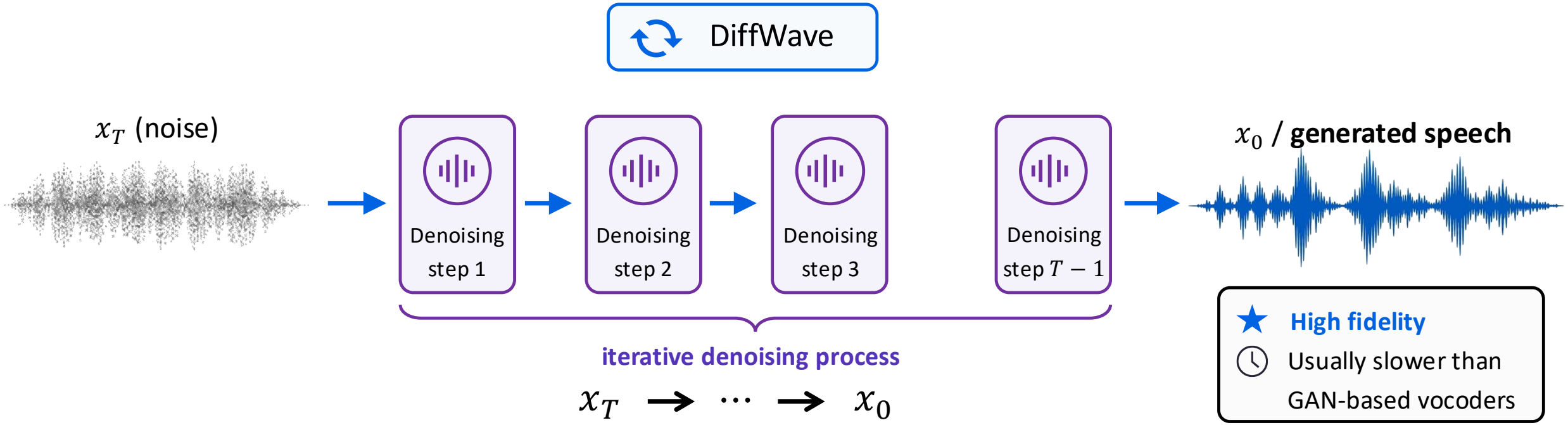
Speed: flow-based > VAE > autoregressive (parallelism vs. sequential generation)

Modeling style: **sample-level (AR/Flow)** Vs. latent-level (VAE)

Different vocoder families trade off quality, speed, and modeling assumptions VAE-based vocoders model speech through a latent space.

Vocoder in Speech LLM: Diffusion-based Vocoder

Diffusion vocoders synthesize speech by iterative denoising and can achieve very high fidelity



Generation process

Gradually removes noise to recover a speech waveform.

Strength

Produces very natural and high-quality audio.

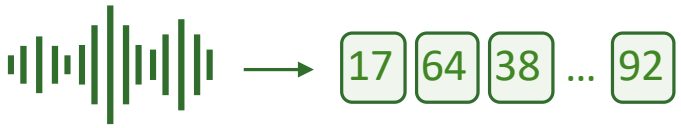
Limitation

Inference is typically slower because multiple denoising steps are required.

Diffusion vocoders offer excellent fidelity through iterative denoising, but often trade speed for quality.

Components in SpeechLM: Summary

1. Speech Tokenizer



- Converts raw speech into model-friendly representations
- Controls the balance between semantic meaning and acoustic detail

Semantics | Acoustics

2. Language Model

- Performs autoregressive token prediction
- Enables reasoning, generation, and speech-text interaction

Reasoning | Generation | Speech-text

3. Vocoder

- Converts generated speech tokens back into waveforms
- Determines speech quality, naturalness, and synthesis speed

Quality | Naturalness | Speed

Key system-level trade-offs

Semantic understanding	Acoustic fidelity
Discrete tokens	Continuous representations
Direct synthesis	Enhanced synthesis
Modular design	End-to-end optimization

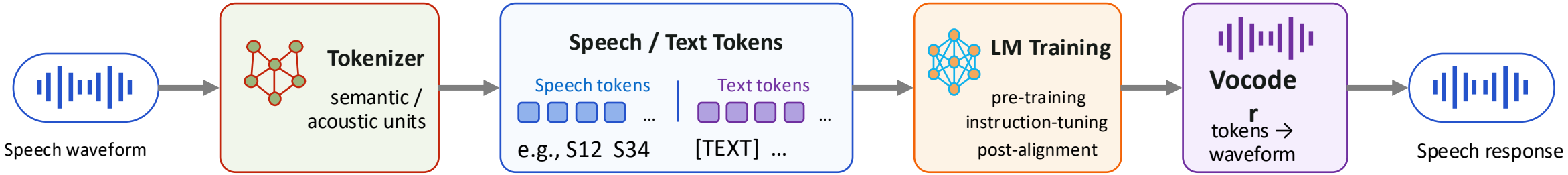
A strong SpeechLM is not defined by any single module, but by how wellthe tokenizer, language model, and vocoder are jointly designed and optimized.

Part III: Training Recipes

Training Recipes for Speech Language Models

Speech LLM performance is determined not only by architecture, but also by what features are modeled, how the language model is trained, and how speech generation is aligned

Section opening schematic



SpeechLM training involves three levels

1) Feature level

What speech representation is modeled

- semantic /acoustic / mixed tokens
- continuous speech representations

2) Stage level

How the language model is trained

- pre-training
- instruction-tuning
- post-alignment

3) Interaction level

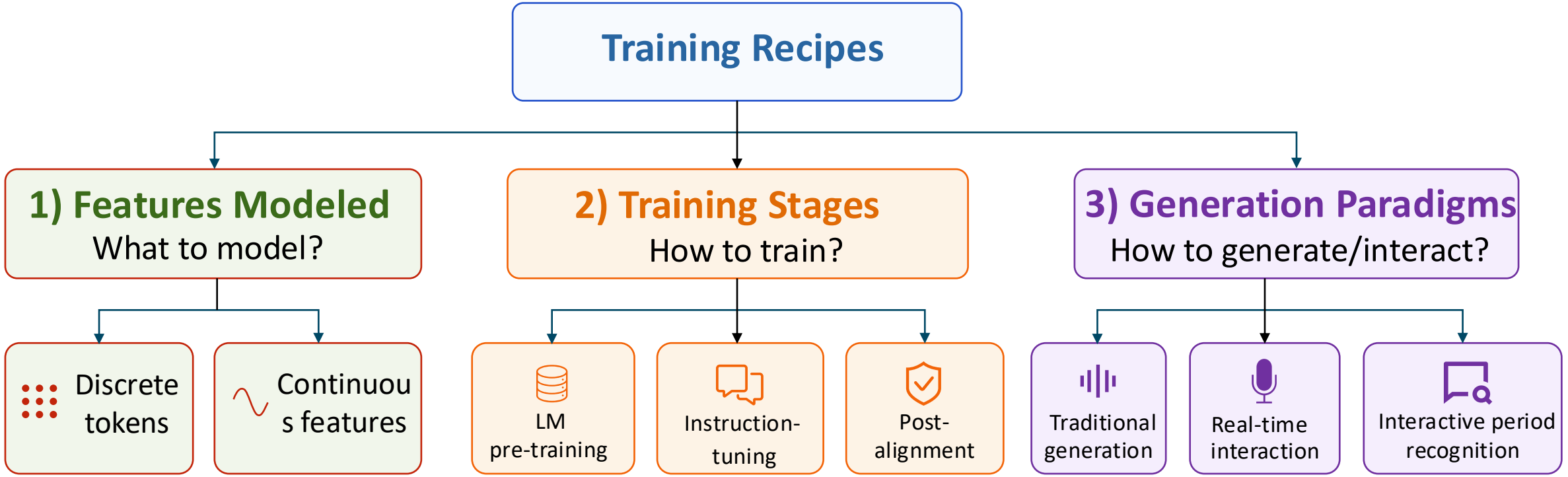
How speech generation is aligned

- offline generation
- streaming
- full-duplex interaction

Understand how Speech LLMs become instruction-following speech agents.

Taxonomy of Training Recipes

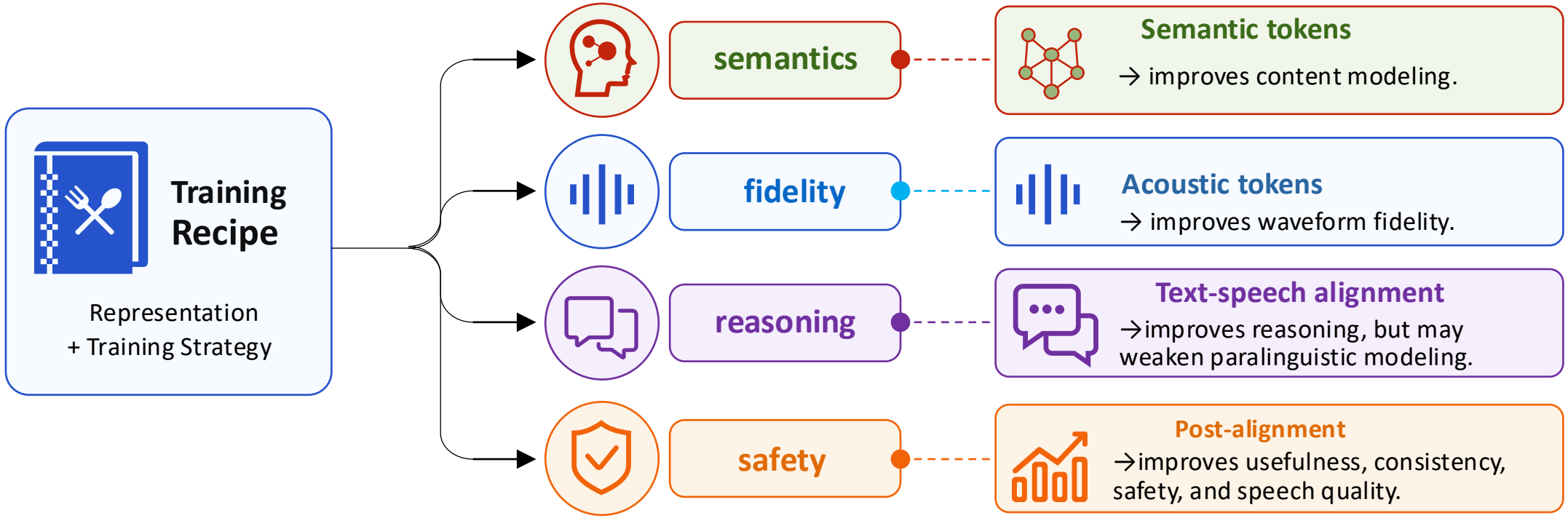
Training recipes can be organized into three questions: what to model, how to train, and how to generate/interact.



Structure the design space of training recipes across modeling choices, training steps, and generation paradigms.

Why Training Recipes Matter

The same Speech LLM backbone can behave differently depending on the representation and training strategy.

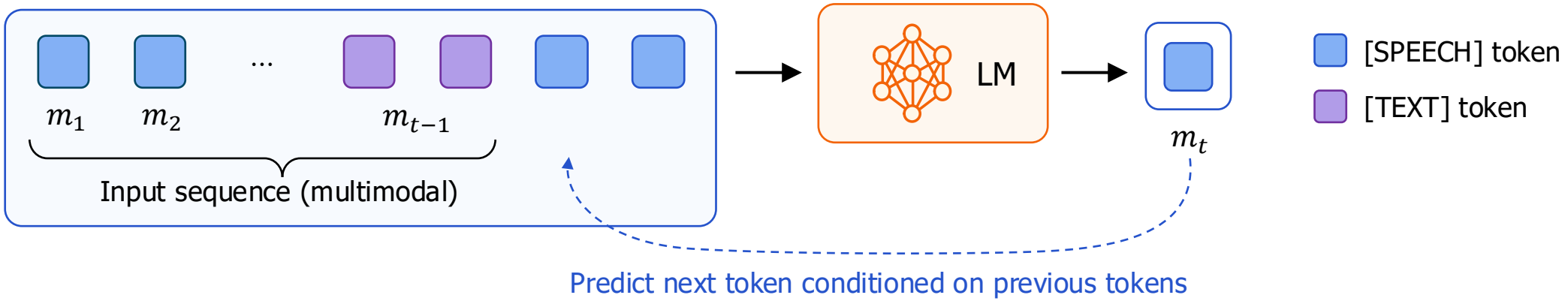


Choosing the right recipe helps SpeechLMs excel on the capabilities that matter most.

Autoregressive Training Objective

Speech LLMs train the language model to predict the next speech, text, or multimodal token.

Autoregressive next-token prediction



Training objective (autoregressive)

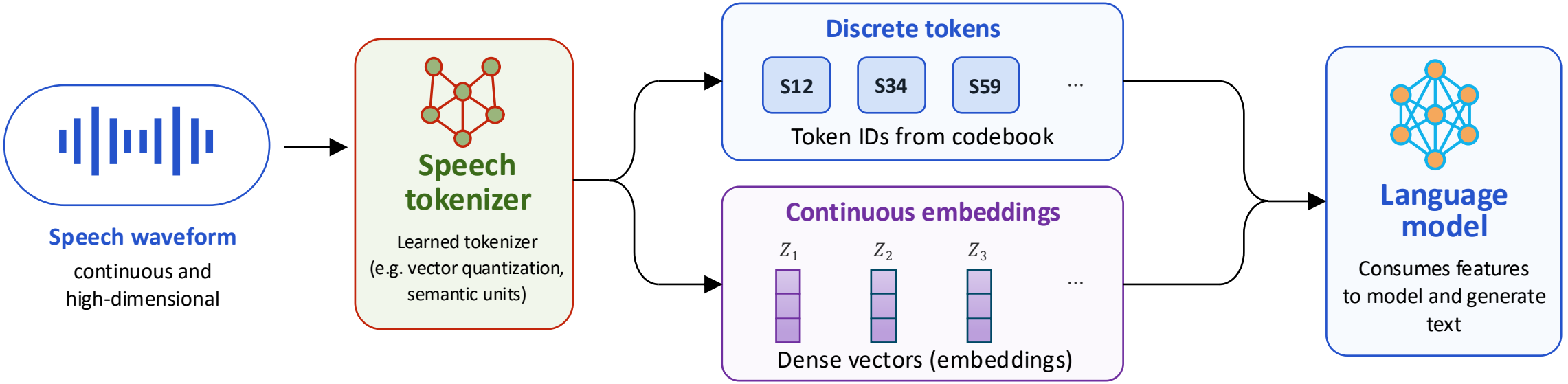
$$\mathcal{L}_{AR}(\theta) = -\sum_{t=1}^{T} \log p_\theta(m_t \mid m_{<t})$$

- Input sequence can contain speech tokens, text tokens, or both.
- The LM learns next-token prediction over a multimodal sequence.
- This objective supports speech continuation, spoken dialogue, ASR, TTS, and speech-to-speech response.

Applications | ASR | TTS | Spoken dialogue | speech-to-speecc

Features Modeled in Speech LLMs

The modeled feature is the representation produced by the speech tokenizer and consumed by the language model.



- Speech waveform is continuous and high-dimensional.
- Speech tokenizer converts waveform into LM-compatible representations.
- Speech LLMs mainly model discrete features and continuous features.

Feature modeling formula

$$s = d(f_E(a))$$

a:speech waveform (continuous)

f_E:speech tokenizer /encoder

d: discretization or embedding mapping

s:features consumed by the LM

Discrete Features vs. Continuous Features

Discrete features are easier to model with LLMs, while continuous features preserve richer speech details.

Discrete features

Codebook indices / token IDs

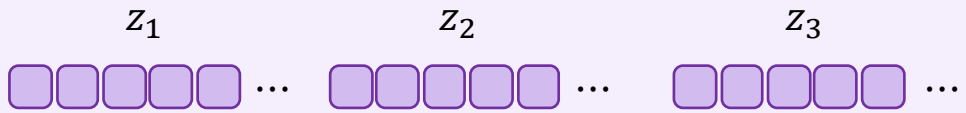
S12 S34 S59

- quantized speech units
- similar to text tokens
- easy for decoder-only LMs

VS.

Continuous features

Real-valued embeddings



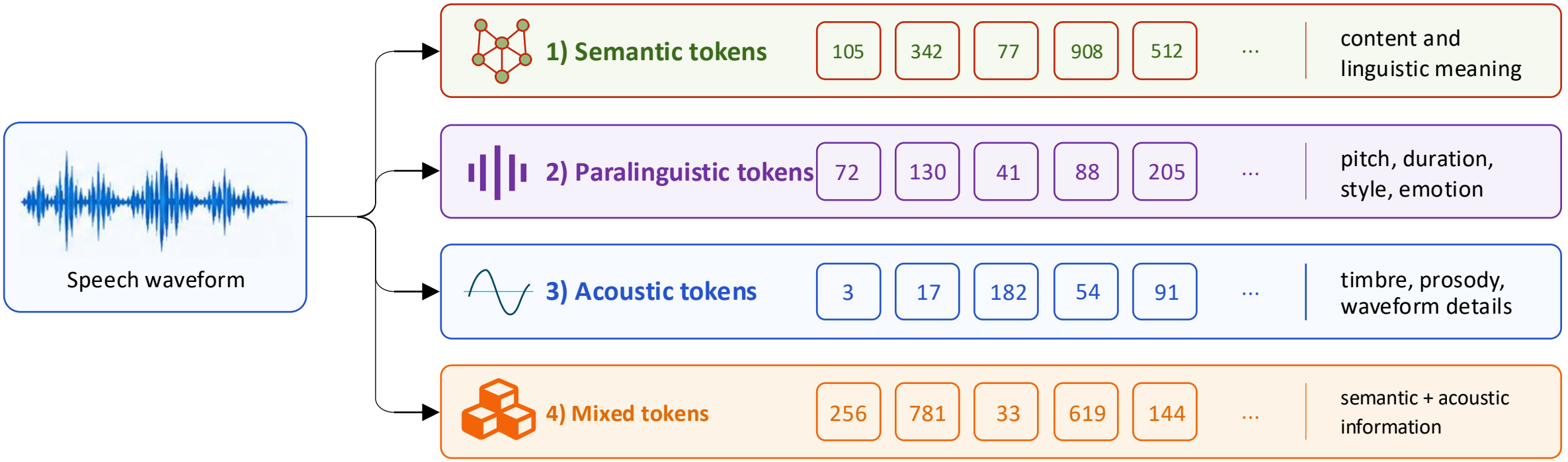
- real-valued embeddings
- preserve fine-grained speech information
- require architecture or training adaptation

Token compatibility vs. information richness

$$s_i \in \{1, .., .. K\}, \quad z_i \in \mathbb{R}^d$$

Types of Discrete Speech Tokens

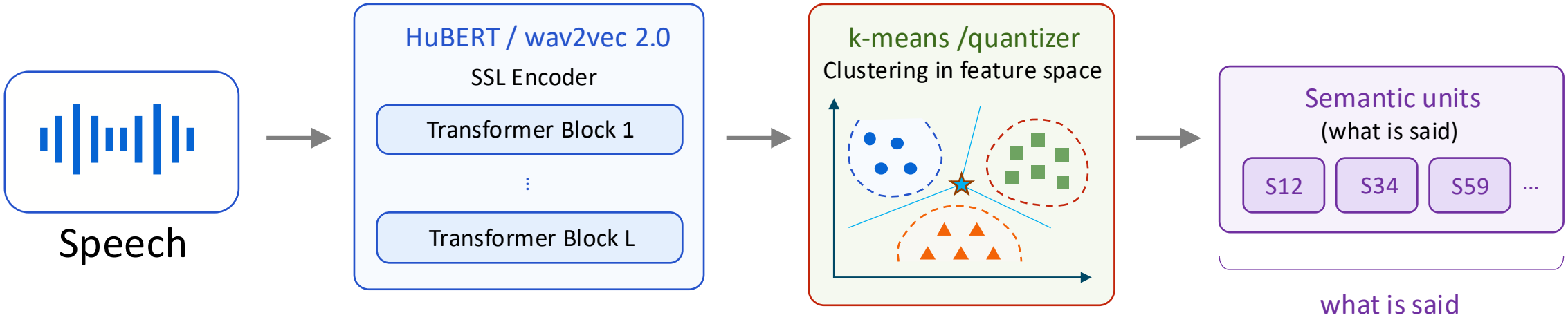
Discrete features can be divided into semantic, paralinguistic, acoustic, and mixed tokens.



Different token types expose different aspects of speech to the LLM

Semantic Tokens

Semantic tokens focus on speech content and support speech understanding and semantically coherent generation.



- Capture speech content
- Extracted from SSL speech models
- Widely used in early Speech LLMs

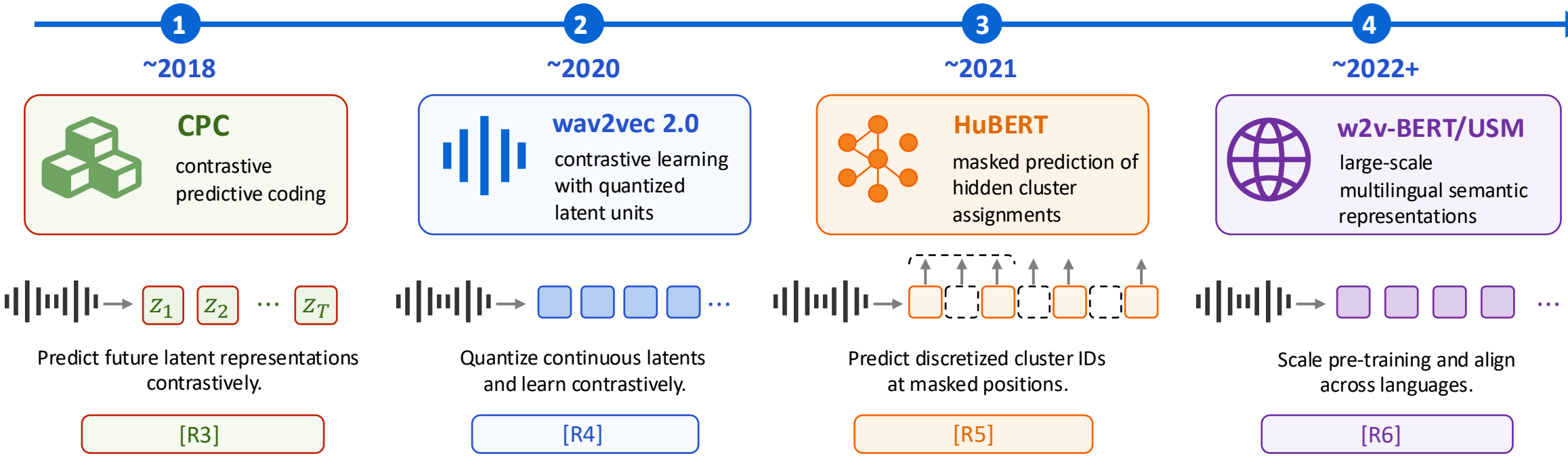
Semantic unit extraction

$$s = d(f_E(a))$$

Representative models: GSLM, TWIST, SpeechGPT, SPIRIT-LM.

Semantic Tokenizer Examples

Different semantic tokenizers differ in how they learn and discretize speech content.



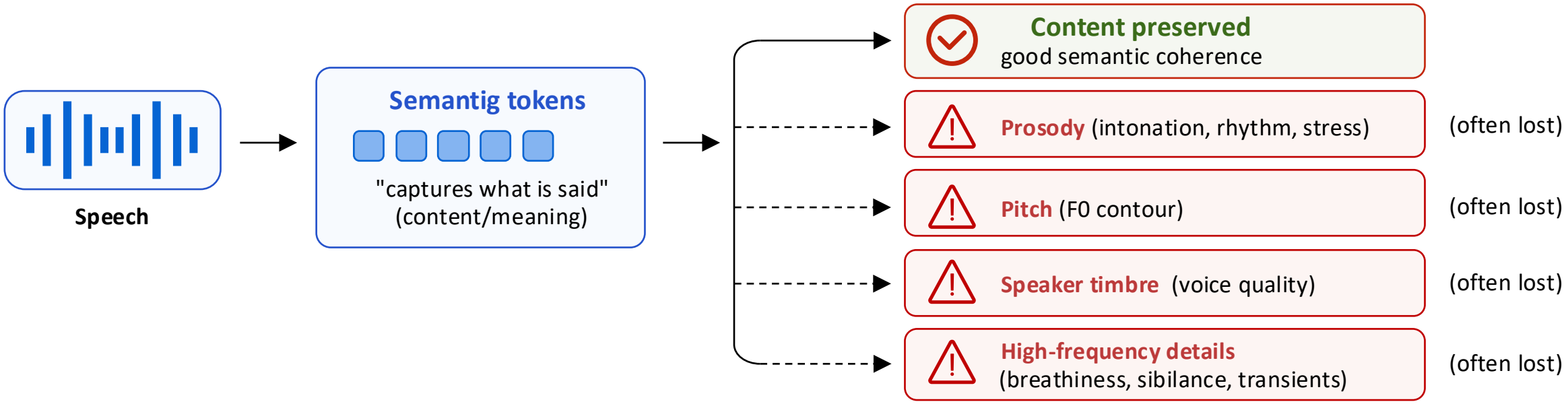
Masked prediction objective (e.g., HuBERT)

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in \mathcal{M}} \log p\left(s_i \mid v_{\setminus \mathcal{M}}\right)$$

S_i:target discrete cluster ID at position i
v:contextual latent representations
M:masked positions

Limitation of Semantic Tokens

Semantic tokens preserve content well but often lose acoustic and expressive information.



Pros:

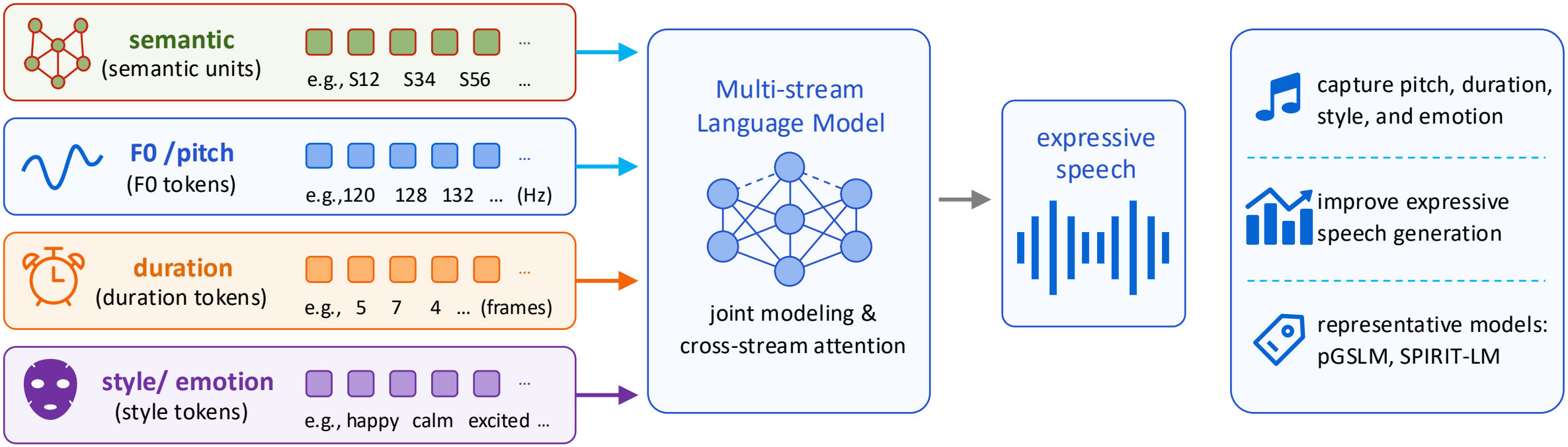
- good semantic coherence
- easy text alignment

Cons:

- weak prosody and speaker cues
- limited acoustic detail
- often needs extra generation modules

Paralinguistic Tokens

Paralinguistic tokens complement semantic tokens by modeling how speech is expressed



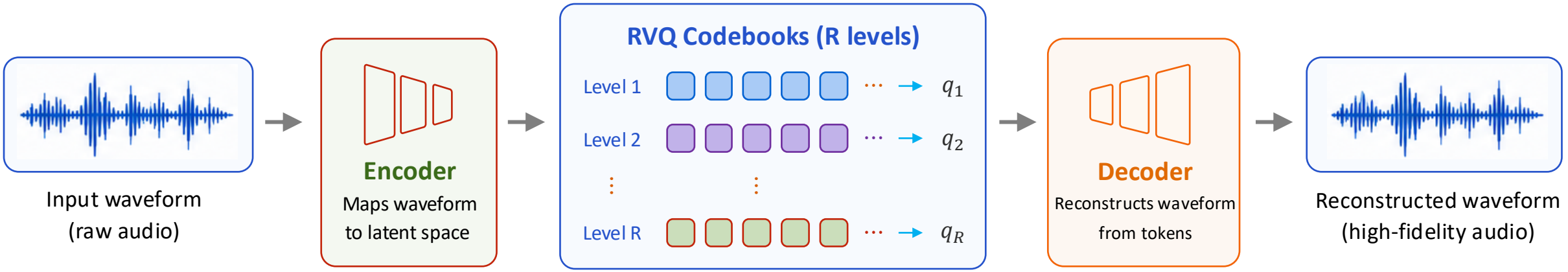
Multi-stream autoregressive factorization

$$p(s, f, d) = \prod_{t} p\left(s_t, f_t, d_t \mid s_{<t}, f_{<t}, d_{<t}\right)$$

s: semantic, f:Fo/pitch, d:duration
t: time step

Acoustic Tokens

Acoustic tokens are codec-based representations optimized for reconstructing high-fidelity speech



Acoustic tokens capture how speech sounds

- codec-based representations
- capture timbre, prosody, waveform detail
- useful for speech synthesis, voice conversion, and codec language modeling
- representative models: EnCodec, SoundStream, VioLA, NTPP

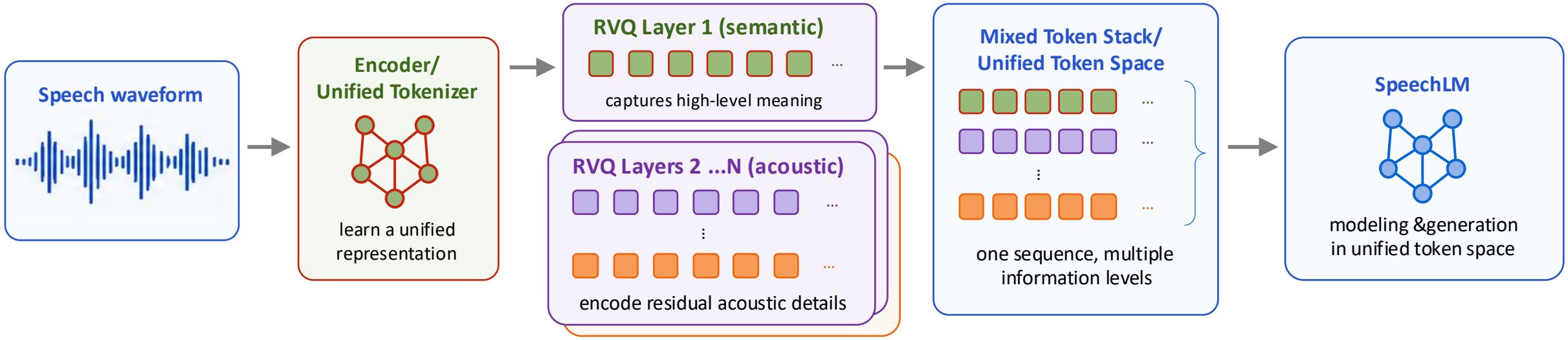
Residual Vector Quantization (RVQ)

$$q_r = Q_r(r_r), \quad \hat{v} = \sum_{r=1}^{R} q_r$$

Q_r(-): codebook (quantizer) at level r
q_r: quantized vector at level r
$\hat{v}$:reconstructed latent vector

Mixed Tokens

Mixed tokens jointly represent semantic content and acoustic detail in a unified token space.



Goal: balance semantic correctness and acoustic fidelity

- semantic distillation into codec layers
- residual quantization for acoustic details
- representative models: SpeechTokenizer, Mimi, SpeechGPT-Gen, Moshi

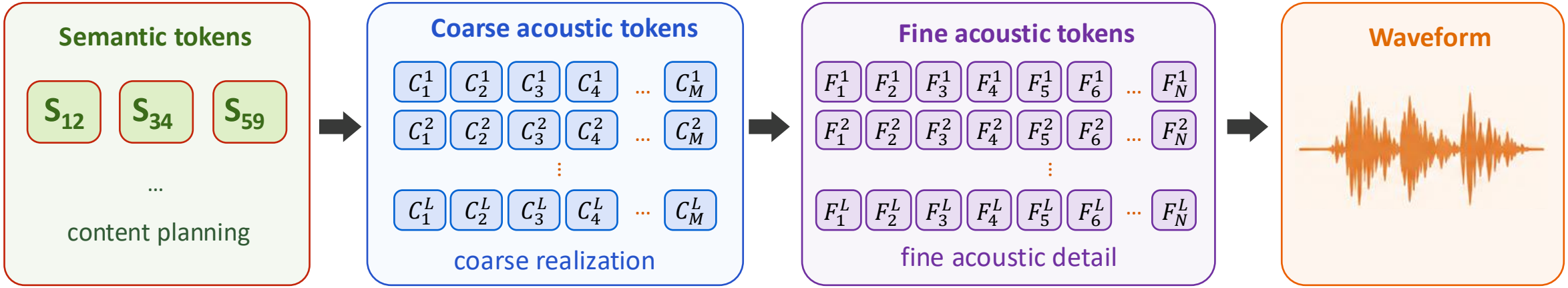
Unified mixed-token sequence

$$s^{\text{mixed}} = [s^{\text{sem}}; s_1^{\text{acoustic}}; \cdots; s_R^{\text{acoustic}}]$$

semantic · acoustic layer 1 · acoustic layer R

Hierarchical Semantic-to-Acoustic Modeling

Some models first predict semantic tokens, then use them to generate acoustic tokens



Stage 1: model semantic structure

Stage 2: predict acoustic realization

Takeaways

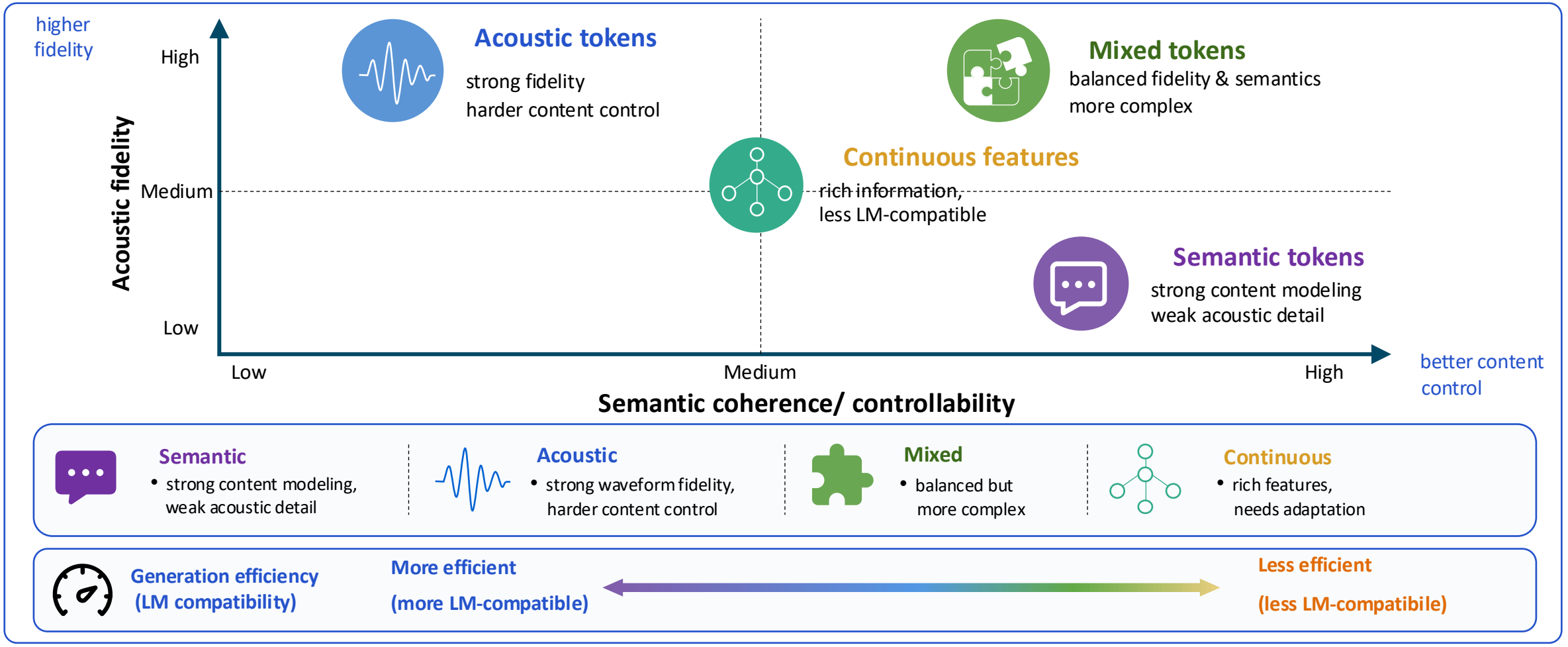
- separates content planning from waveform generation
- improves controllability of synthesis
- longer sequences and higher latency

$$p(a) \approx p(s^{\text{sem}}) \cdot \left(s^{\text{acoustic}} \mid s^{\text{sem}}\right)$$

a= speech waveform S^{sem} = semantic tokens $s^{acoustic}$ = acoustic tokens

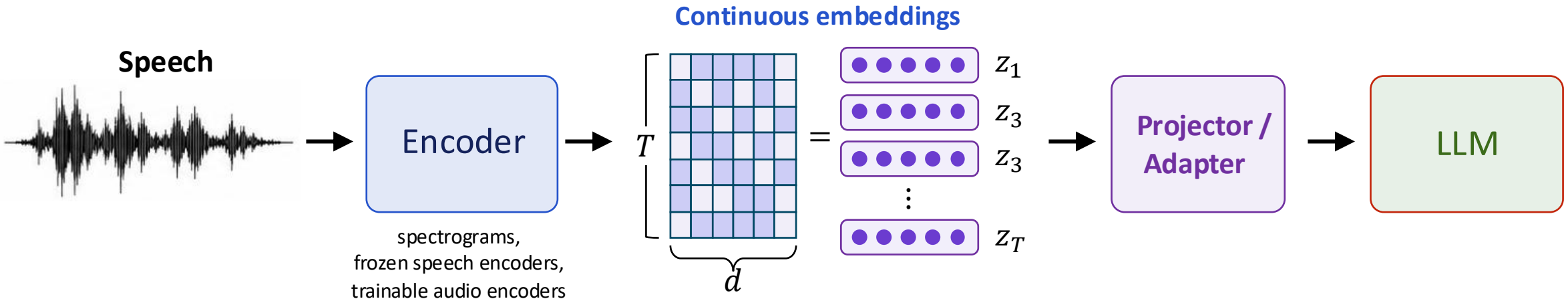
Token Trade-off: Semantics vs. Fidelity

Different token choices reflect a trade-off between semantic controllability, acoustic quality, and generation efficiency.



Continuous Features

Continuous features preserve fine-grained speech information without quantization



- real-valued representations from speech encoders
- reduce discretization loss
- require projectors, adapters, or LM input modification

$$\mathbf{z} = f_E(\mathbf{a}), \quad \mathbf{z} \in \mathbb{R}^{T \times d}$$

a :raw audio or acoustic input

f_E: speech encoder function

T:number of time steps (frames)

d:embedding dimensionality

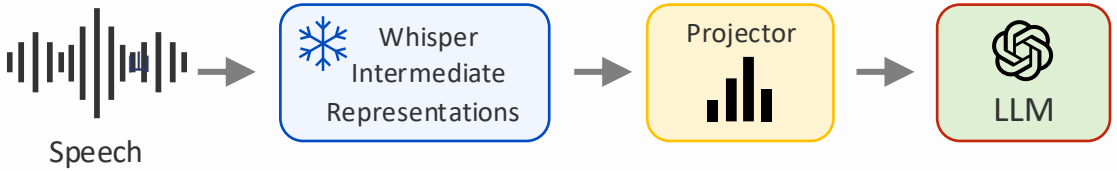
Examples of Continuous Feature Modeling

Continuous-feature SpeechLMs often connect frozen or trainable audio encoders to LLMs

1 Spectron

predicts spectrograms frame by frame

2 Mini-Omni

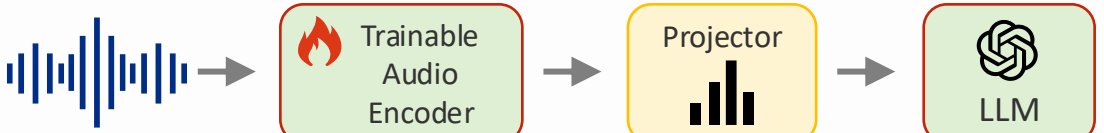


uses Whisper intermediate representations

3 SLAM-Omni

speech encoder + LLM alignment

4 LauraGPT



jointly trains audio encoder and LLM

$$h = \mathrm{Proj}\big(f_E(\mathbf{a})\big), \quad y \sim \mathrm{LM}(\mathbf{h}, x)$$

h: continuous hidden features passed to the LLM
a:input audio (e.g., waveform)
x:text/prompt input to the LLM

Frozen encoder + projector
Leverage strong pre-trained encoders;lightweight adapter.

Encoder-LLM alignment
Align representations for better interaction.

Jointly trained audio front-end
End-to-end training for maximum adaptation.

How to Choose the Feature Type?

Feature choice should match the target capability of the Speech LLM.

Task (target capability)	Semantic tokens	Paralinguistic +semantic	Acoustic/ mixed tokens	Continuous encoder features
ASR / semantic QA	★	○	◯	★
Expressive dialogue	○	★	★	○
High-fidelity TTS	◯	○	★	○
Real-time interaction	○	✓	✓	✓

★ Strongly recommended (best fit) ✓ Recommended (good fit) ○ Moderate (conditional) ◯ Possible (weak fit)

Task requirement
(e.g., accuracy, expressiveness, fidelity, latency)

→

Feature representation
(tokenizer / features) matched to requirement

→

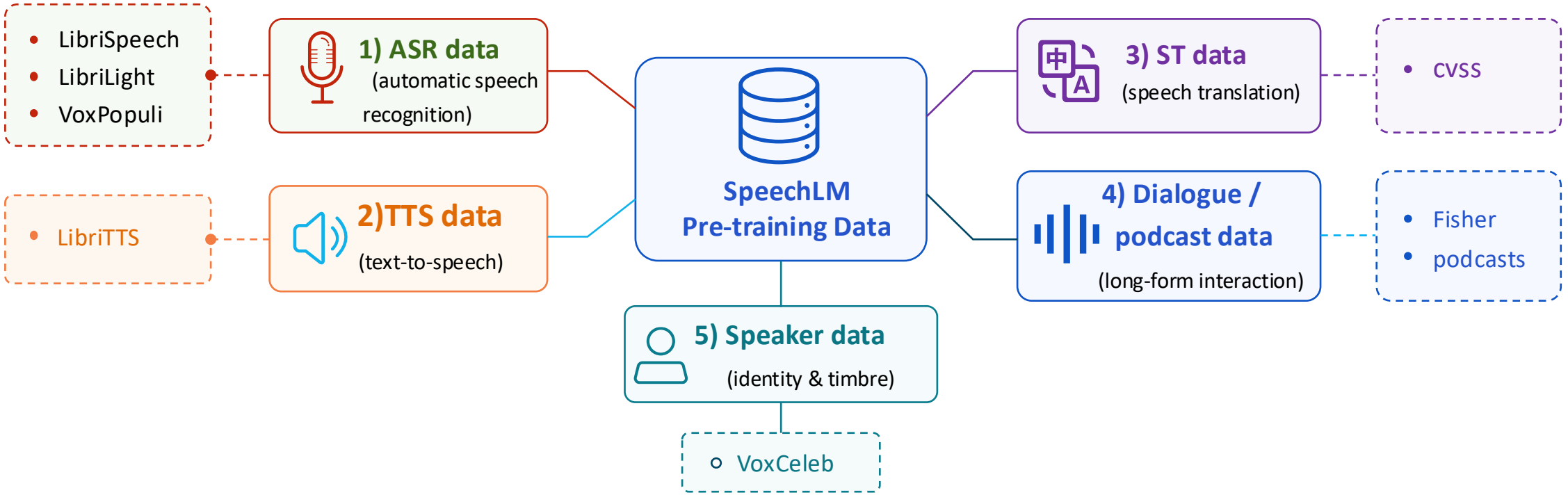
LM capability
(understanding, generation, control, robustness)

Takeaways

- Match token type to target capability.
- No single feature is optimal for all tasks.
- Latency and vocoder design also matter.

Training Data for Speech LLMs

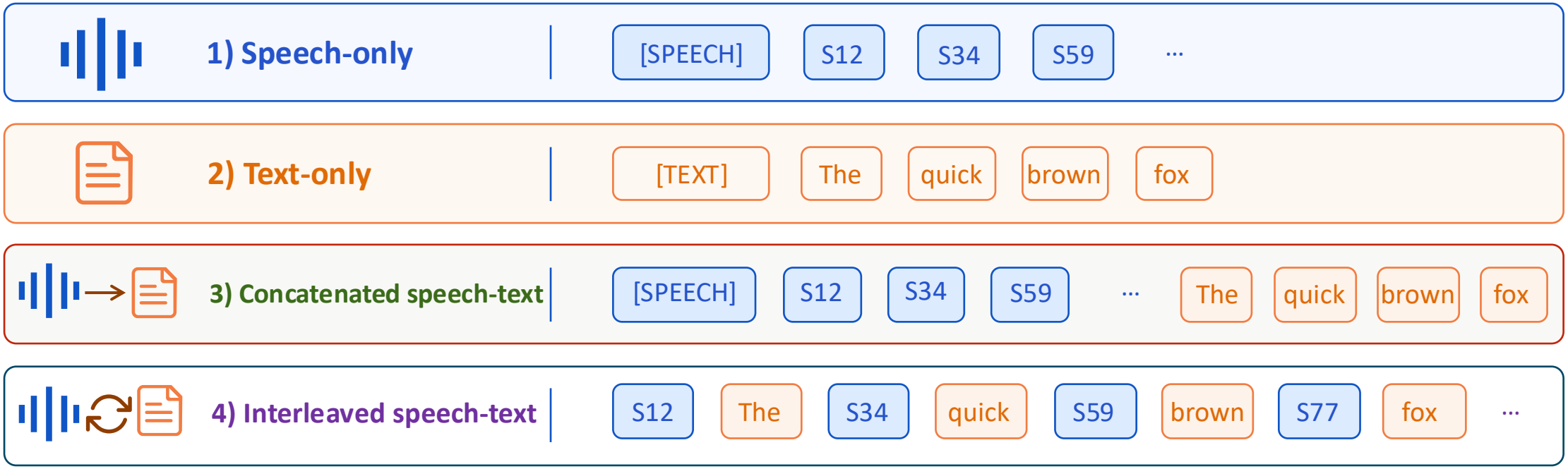
Speech LLM pre-training uses heterogeneous speech data:ASR, TTS, speech translation, podcasts, dialogues, and speaker data.



Data type determines what the model learns: ASR helps content modeling, TTS helps generation, ST enables cross-lingual transfer, dialogue/podcast data supports long-form interaction, and speaker data enables identity and timbre modeling.

Four Ways to Model Speech and Text

Speech LLMs can model pure speech, pure text, concatenated speech-text, or interleaved speech-text sequences.

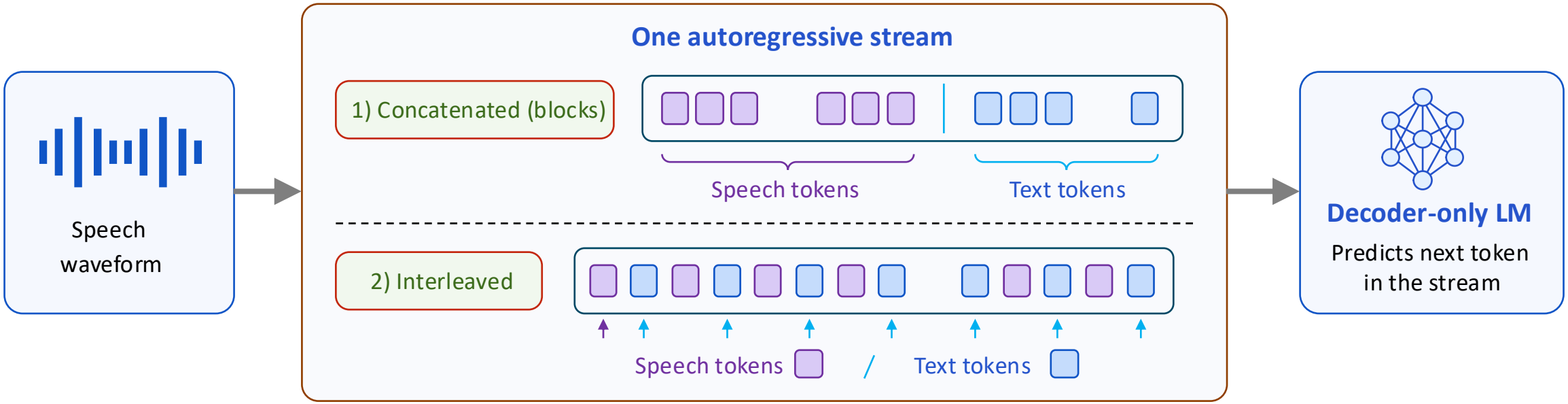


These four forms show how speech and text are organized into one LM input sequence.

$$m = [s_1, \ldots, s_i, t_1, \ldots, t_j, \ldots]$$

Single-Sequence Alignment

Single-sequence alignment inserts speech and text tokens into one autoregressive stream.



Advantages

Simple autoregressive training.

Direct text-speech knowledge transfer.

$$p(m) = \prod_t p\left(m_t \mid m_{<t}\right)$$

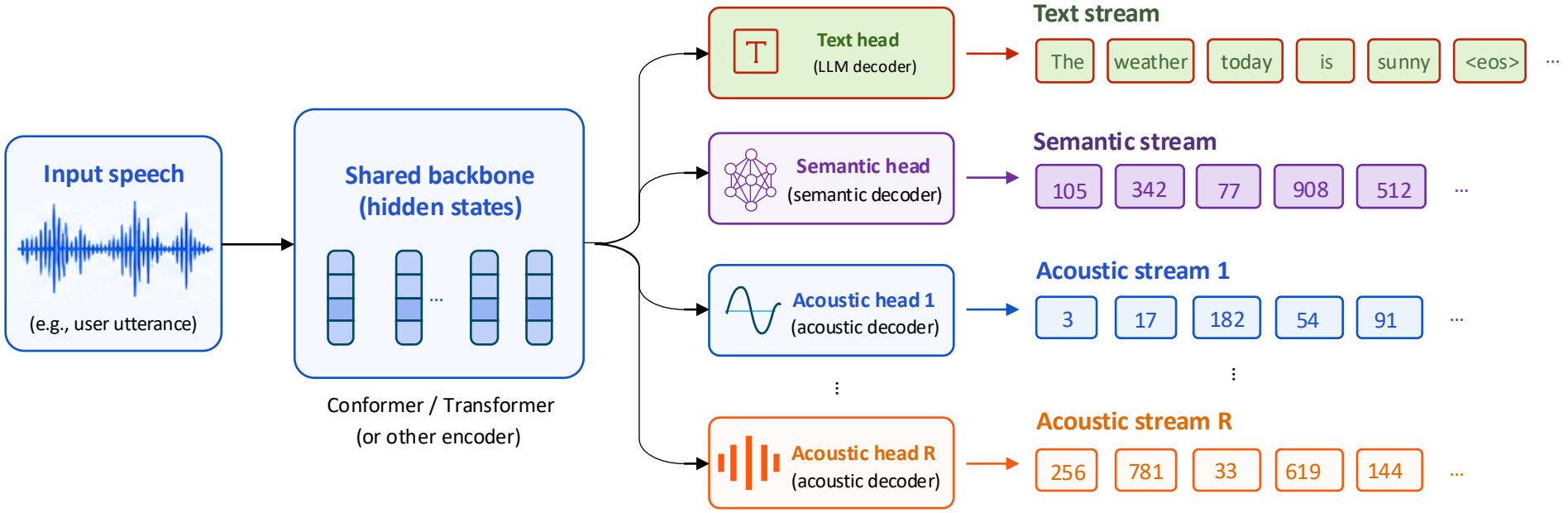
m: token sequence (speech + text)

Limitations

- Longer sequence to model.
- Possible loss of paralinguistic detail.

Multi-Sequence Alignment

Multi-sequence alignment generates text and speech streams in parallel



Parallel decoding
All streams are decoded at each time step.

Multi-stream generation
Text, semantic, and multiple acoustic streams are produced together.

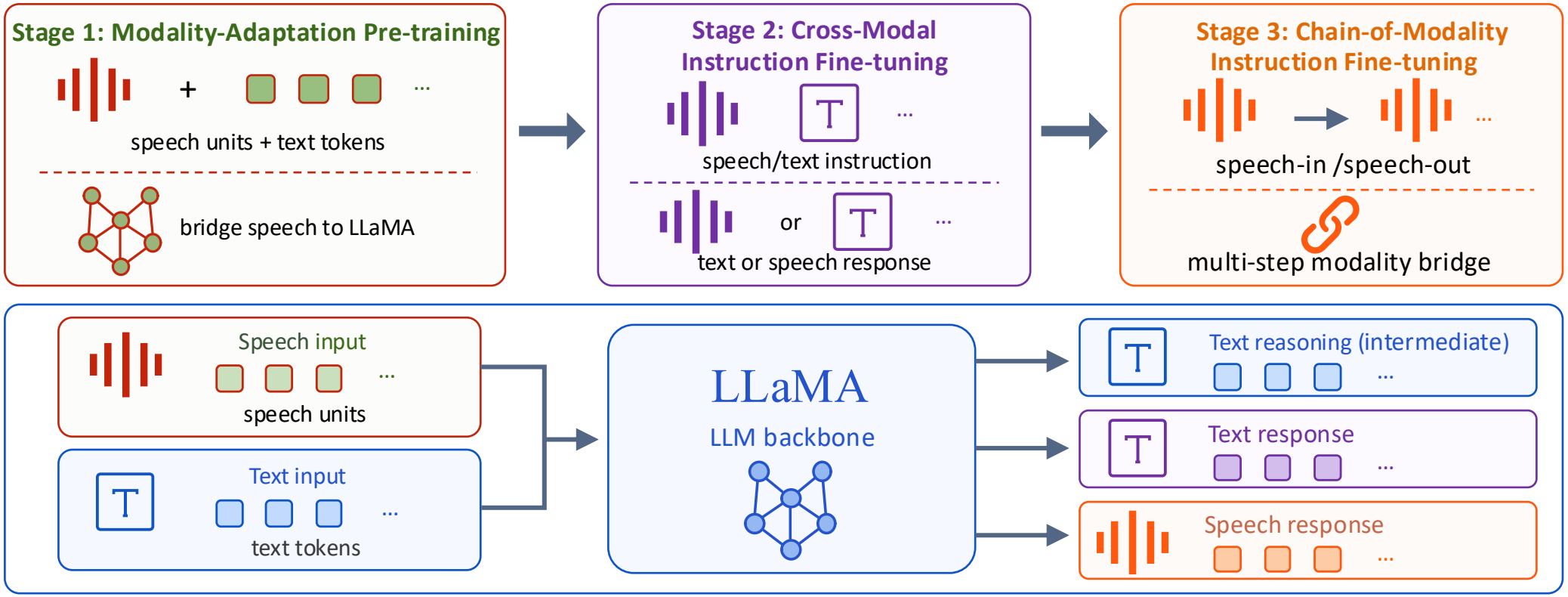
Low-latency interaction
Enables real-time speech understanding and response generation.

Useful for real-time speech interaction

$$p(y^{text}, y^{speech} \mid x) = \prod_t p\left(y_t^{text}, y_t^{speech} \mid y_{<t}, x\right) \quad \Big| \quad y^{speech} = \{y^{sem}, y_1^{acoustic}, \ldots, y_R^{acoustic}\}$$

SpeechGPT: Cross-Modal Instruction Tuning

SpeechGPT uses staged instruction tuning to bridge speech input, text reasoning, and speech output.

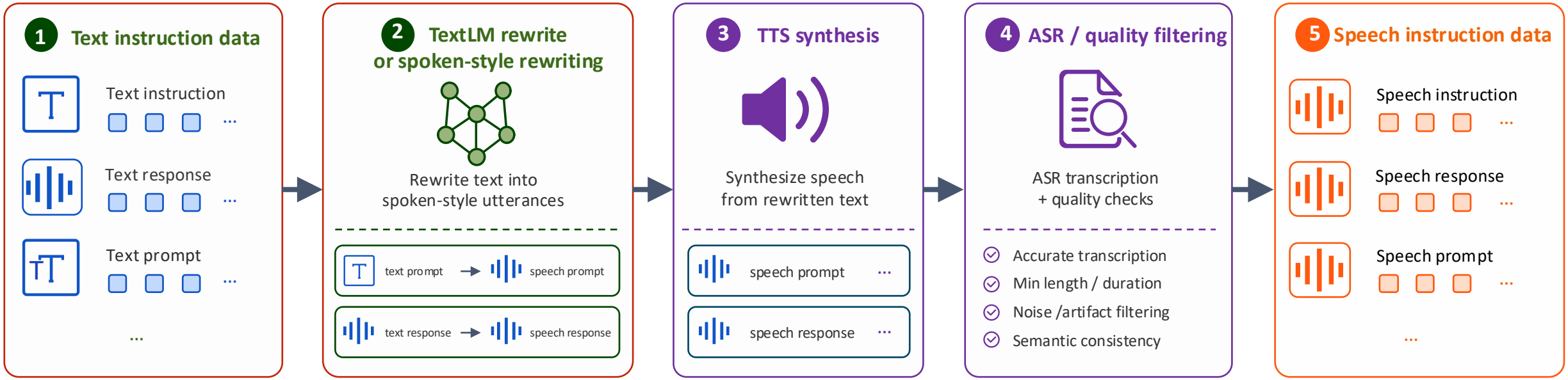


Convert instruction-following data into speech-text formats.

$$L_{\text{inst}} = -\log p_\theta \left(y^{\text{text}}, y^{\text{speech}} \mid x^{\text{speech/text}} \right)$$

Synthetic Speech Instruction Data

Many SpeechLMs create speech instruction data by converting text instruction datasets into spoken form.



Used in

- Llama-Omni
- COSMIC
- SpeechGPT-style training

Caution / potential issues

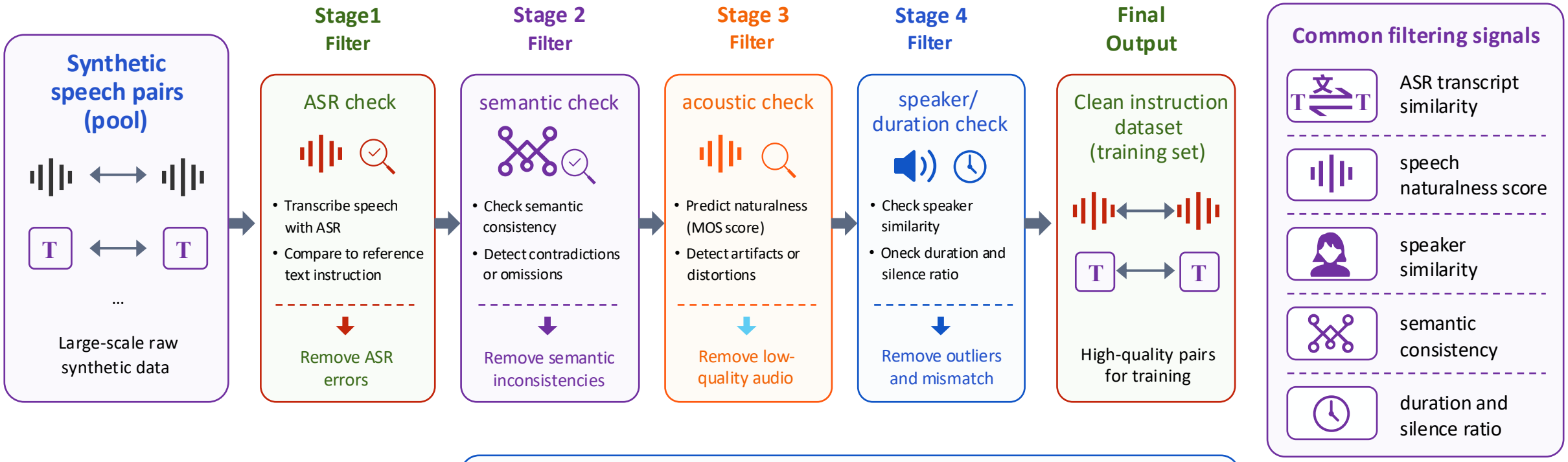
- TTS voice bias
- Synthetic speech artifacts
- Semantic mismatch

Synthetic speech conversion

$$(x^{\text{speech}}, y^{\text{speech}}) = \text{TTS}(x^{\text{text}}, y^{\text{text}})$$

Quality Control for Speech Instruction Data

Synthetic speech instruction data must be filtered to avoid propagating ASR, TTS, and semantic errors.



Important:
Important for speech-in-speech-out instruction tuning.

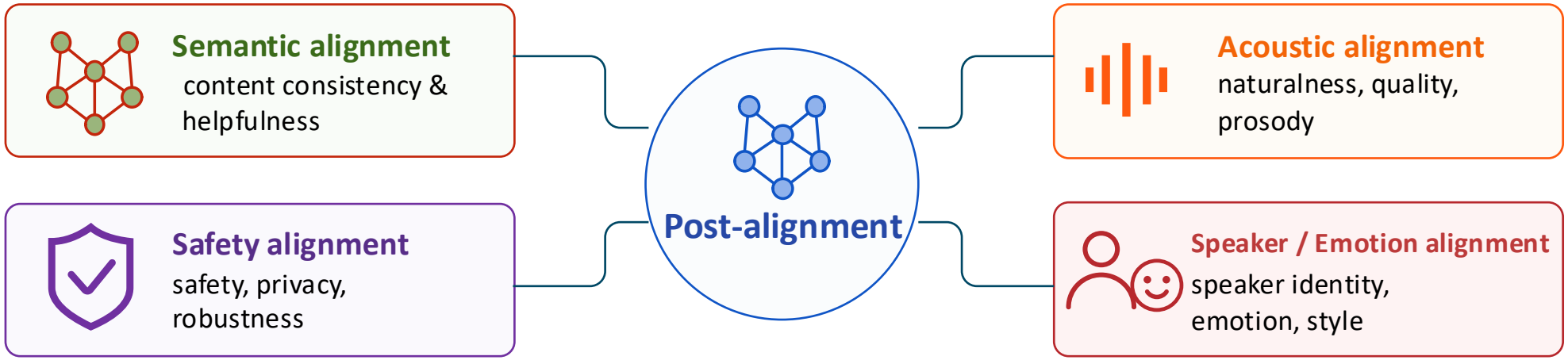
Quality score for each pair

$$score = \alpha\ \mathrm{Sim}_{sem} + \beta MOS_{pred} + \gamma\ \mathrm{SpkSim}$$

α, β, γ:weights(α+β+γ=1)
Sim_{sem}:semantic similarity score
MOS_{pred} :predicted naturalness (MOs)
$SpkSim$: speaker similarity score

Post-alignment in Speech LLMs

Post-alignment refines SpeechLM behavior after pre-training and instruction tuning, but speech introduces extra alignment challenges.



Goals

- helpful and safe responses
- semantic consistency
- acoustic quality
- speaker and emotion appropriateness

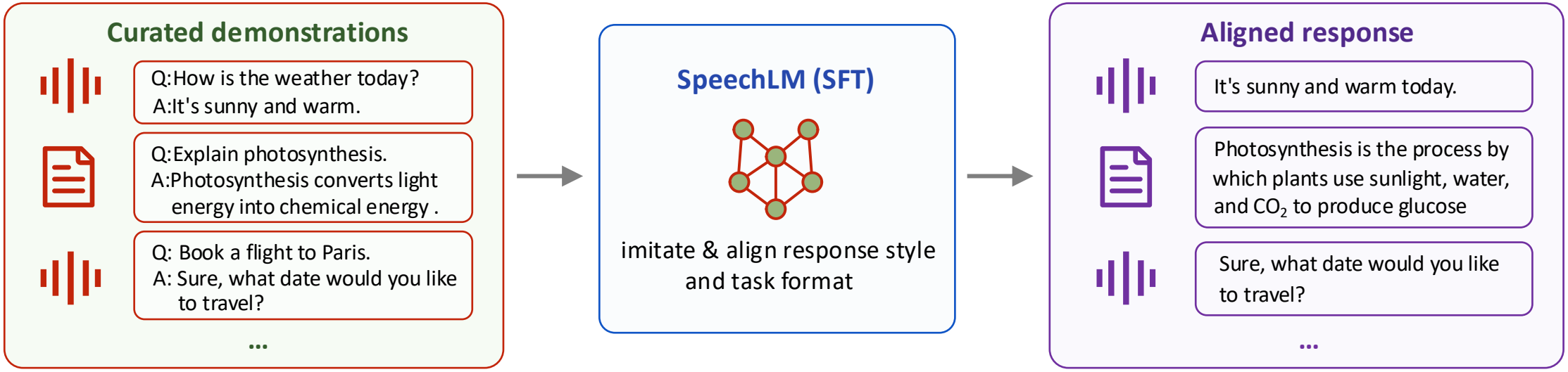
Speech-specific challenges

- generated token distribution mismatch
- ASR-based evaluation noise
- subjective acoustic preference
- privacy and voice safety

$$L_{align} = L_{sem} + \lambda_1 L_{acoustic} + \lambda_2 L_{safety} + \lambda_3 L_{speaker}$$

SFT as Alignment Baseline

Supervised fine-tuning is the simplest post-alignment method when high-quality demonstrations are available.



Goal
- imitate high-quality human or teacher responses

Input / Output
- speech/ text prompt →preferred response

Optimization
- maximize likelihood of reference response

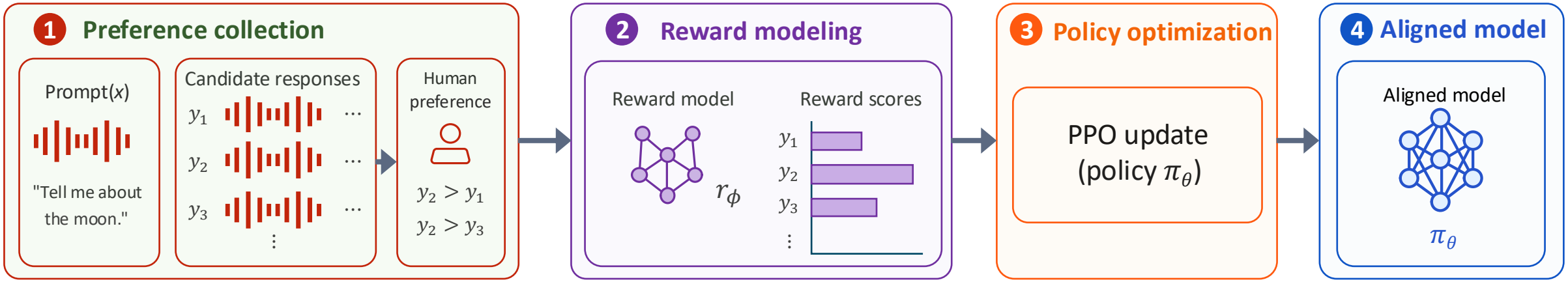
SpeechLM relation
- align spoken response style and task format

Limitation
- cannot directly learn preference among multiple valid responses

$$L_{SFT} = -\mathbb{E}_{(x,y^*)} \log \pi_\theta(y^* \mid x)$$

RLHF / PPO

RLHF uses human preferences to train a reward model, then optimizes the policy with reinforcement learning



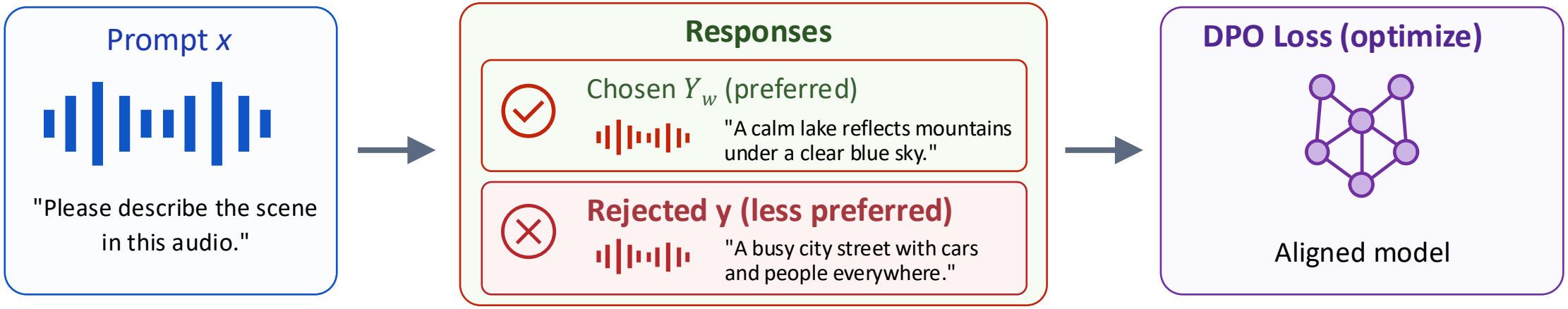
Goal:	maximize human-preferred behavior
Input / Output:	prompt →multiple responses → human preference
Optimization:	reward model and policy update with PPO
SpeechLM relation:	optimize semantic helpfulness or acoustic preference
Limitation:	expensive, unstable, and hard to scale for speech outputs

$$\max_\theta \mathbb{E}_{y\sim\pi_\theta}\left[r_\phi(x,y) - \beta D_{KL}\left(\pi_\theta \,\|\, \pi_{ref}\right)\right]$$

$$\mathcal{L}_{PPO} = \mathbb{E}[\min(r_t A_t, \text{clip}(r_t, 1-\epsilon, 1+\epsilon)A_t)]$$

Direct Preference Optimization

DPO directly optimizes preference pairs without explicitly training a reward model or running online RL.

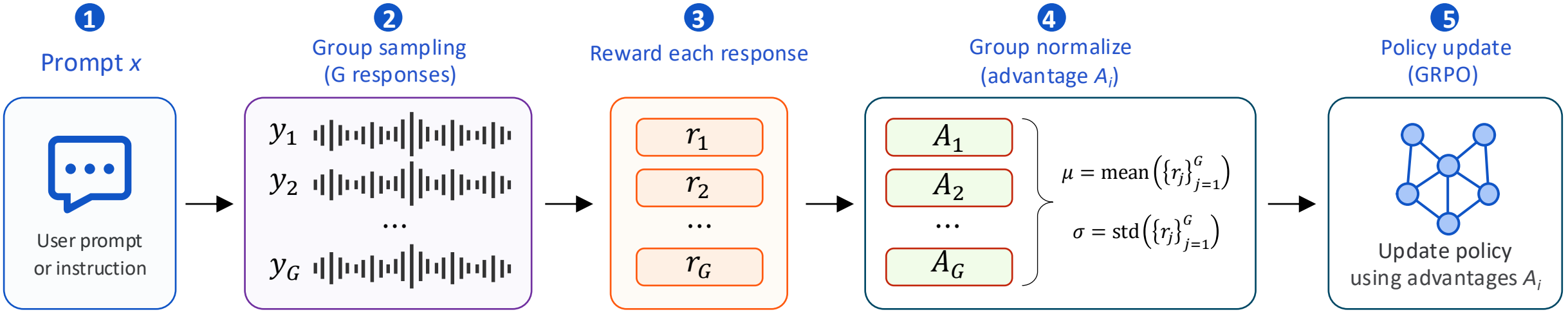


- **Goal:** make preferred response more likely than rejected response
- **Input / Output:** prompt, chosen response, rejected response
- **Optimization:** classification-style preference loss
- **SpeechLM relation:** rank speech outputs by humans, ASR, or AI judges
- **Limitation:** depends heavily on preference data quality

$$L_{DPO} = -\mathbb{E}\log\sigma\left[\beta\log\left(\frac{\pi_\theta(y_w \mid x)\pi_{\text{ref}}(y_l \mid x)}{\pi_{\text{ref}}(y_w \mid x)\pi_\theta(y_l \mid x)}\right)\right]$$

Group Relative Policy Optimization

GRPO estimates relative advantages within a group of sampled responses, avoiding a separate value model.



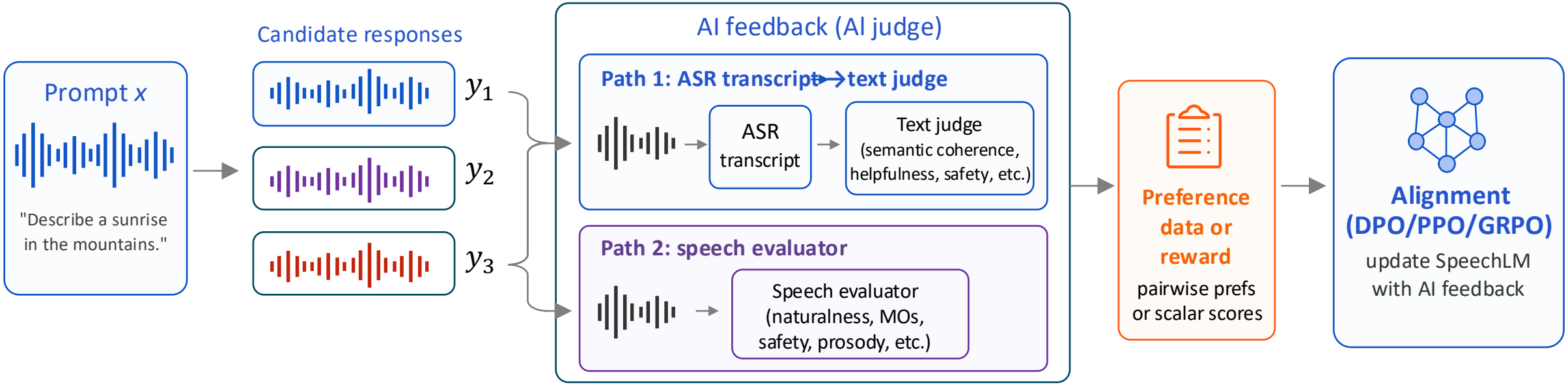
- **Goal:** improve policy using group-normalized rewards
- **Input / Output:** prompt →group of responses
- **Optimization:** compare responses within the same group
- **SpeechLM relation:** automatic rewards from ASR, speaker similarity, MOS predictor, or safety score
- **Limitation:** reward design for speech is difficult

$$A_i = \frac{r_i - \text{mean}\left(\{r_j\}_{j=1}^{G}\right)}{\text{std}\left(\{r_j\}_{j=1}^{G}\right)}$$

$$L_{\text{GRPO}} = -\frac{1}{G}\sum_{i=1}^{G} min(\rho_i A_i, \text{clip}(\rho_i, 1-\epsilon, 1+\epsilon)A_i)$$

RLAIF: Reinforcement Learning from AI Feedback

RLAIF replaces or supplements human preference labels with AI-generated feedback.

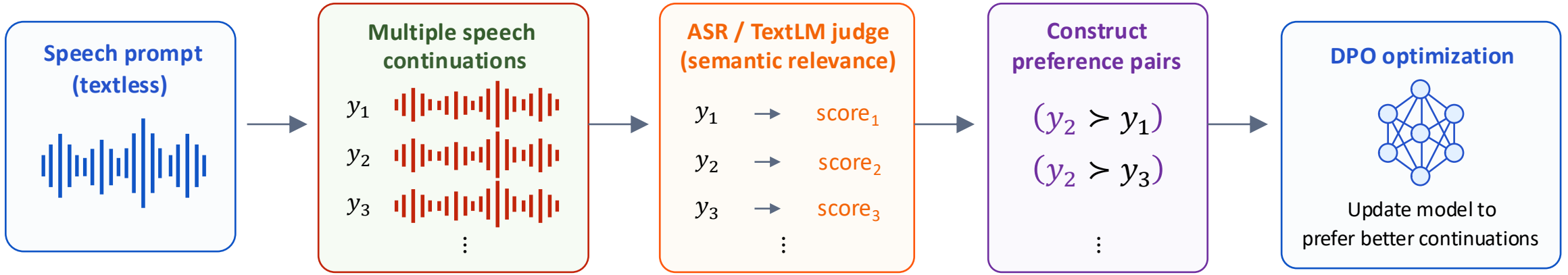


- **Goal:** reduce human annotation cost
- **Input / Output:** prompt and candidate responses →AI judge scores or preferences
- **Optimization:** reward modeling, DPO, or PPO-style update
- **SpeechLM relation:** AI judge evaluates transcribed content, safety, or semantic coherence
- **Limitation:** may ignore acoustic quality or inherit bias

$$y_w \succ y_l \leftarrow J_{AI}(x, y_w, y_l)$$

Case Study: Align-SLM

Align-SLM uses preference optimization with AI feedback to improve semantic coherence in textless spoken language models.



Problem

textless SLMs may generate semantically inconsistent continuations

Strategy

- generate multiple continuations
- transcribe or evaluate semantically
- construct preference pairs
- apply DPO-style optimization

Main target

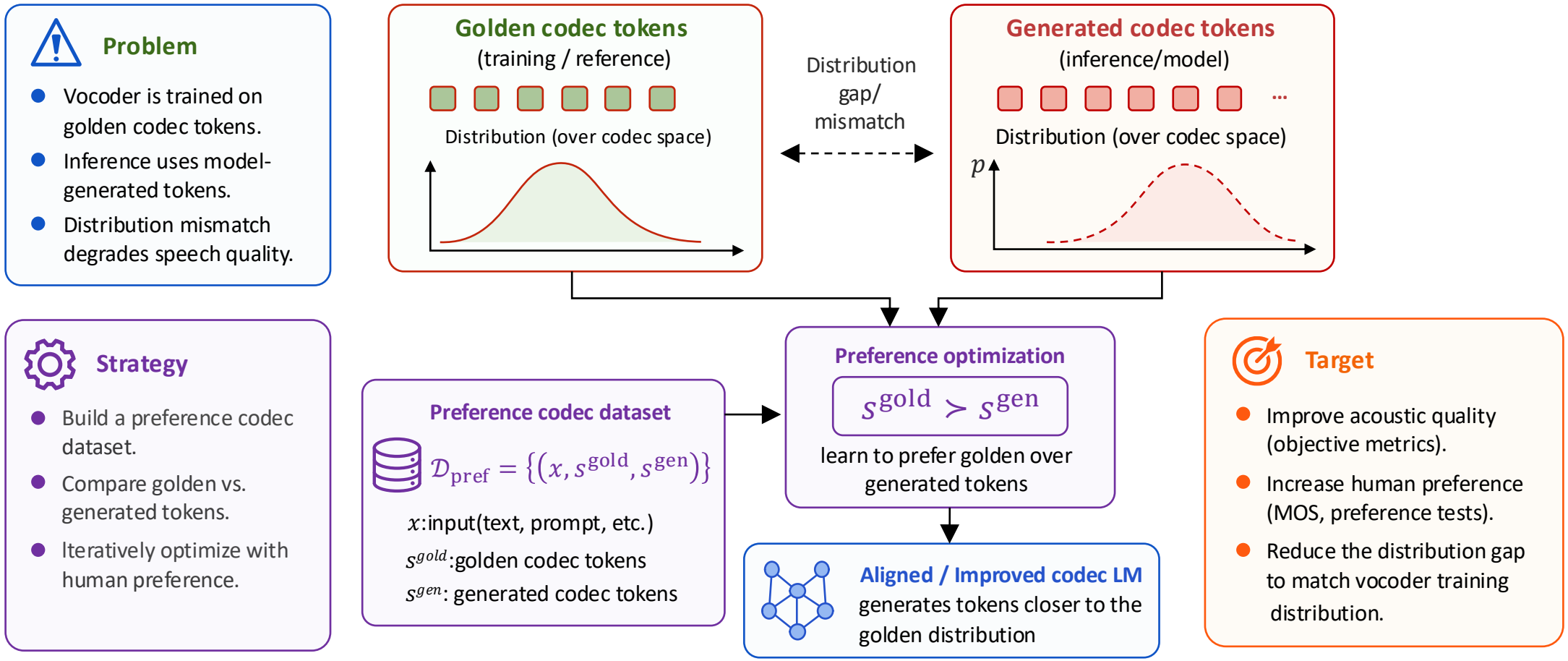
semantic relevance and coherence

$$(y_w, y_l) = \underset{y}{\text{argmax}}\ J_{sem}(x, y), \quad \underset{y}{\text{argmin}}\ J_{sem}(x, y)$$

$J_{sem}(x, y)$: semantic relevance score

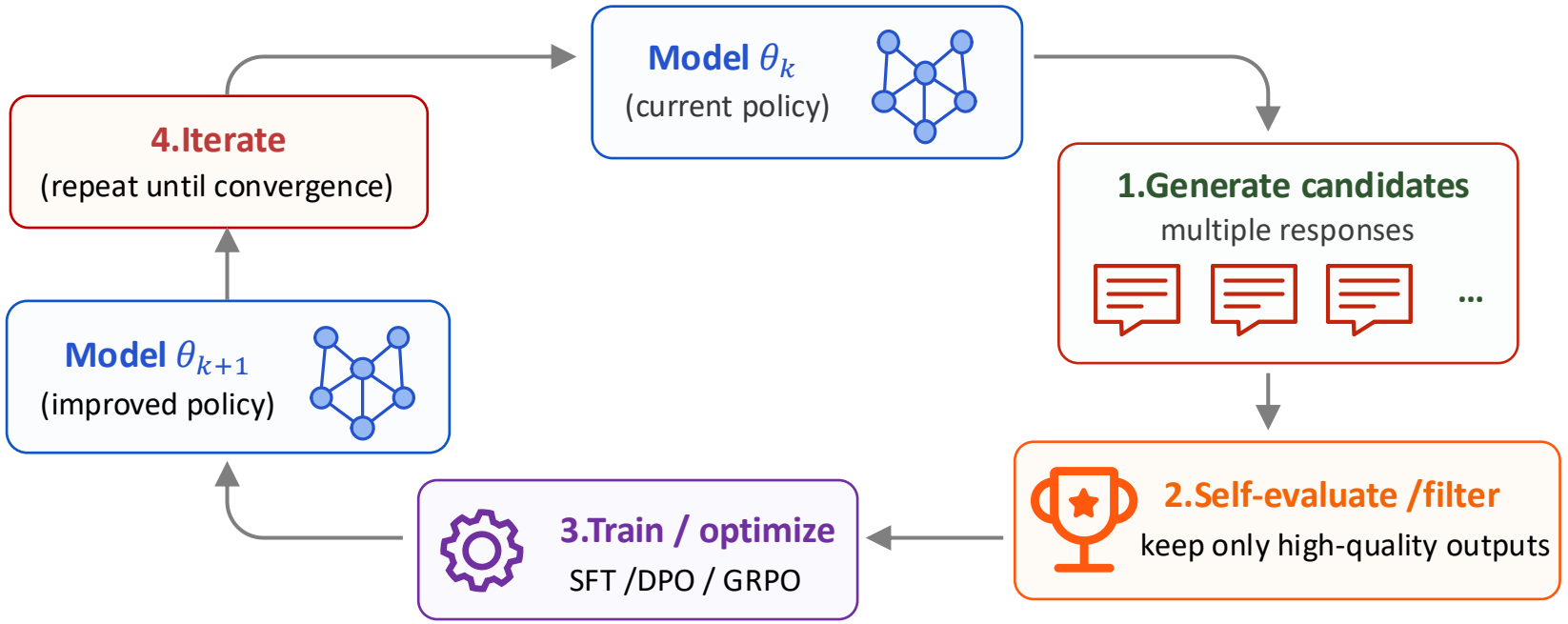
Case Study: SpeechAlign

SpeechAlign targets the distribution gap between golden codec tokens and model-generated codec tokens.



Online Self-Distillation

Online self-distillation lets the model iteratively generate, filter, and learn from its own improved outputs.



$$D_k^{\text{self}} = \{(x, \hat{y}), \mid \hat{y} \sim \pi_{\theta_k} Q(x, \hat{y}) > \tau\} \quad \theta_{k+1} = \text{argmin}_\theta \mathcal{L}(\theta; D_k^{\text{self}})$$

Goal

- continuous self-improvement without fully relying on human labels

Diff.from SFT:

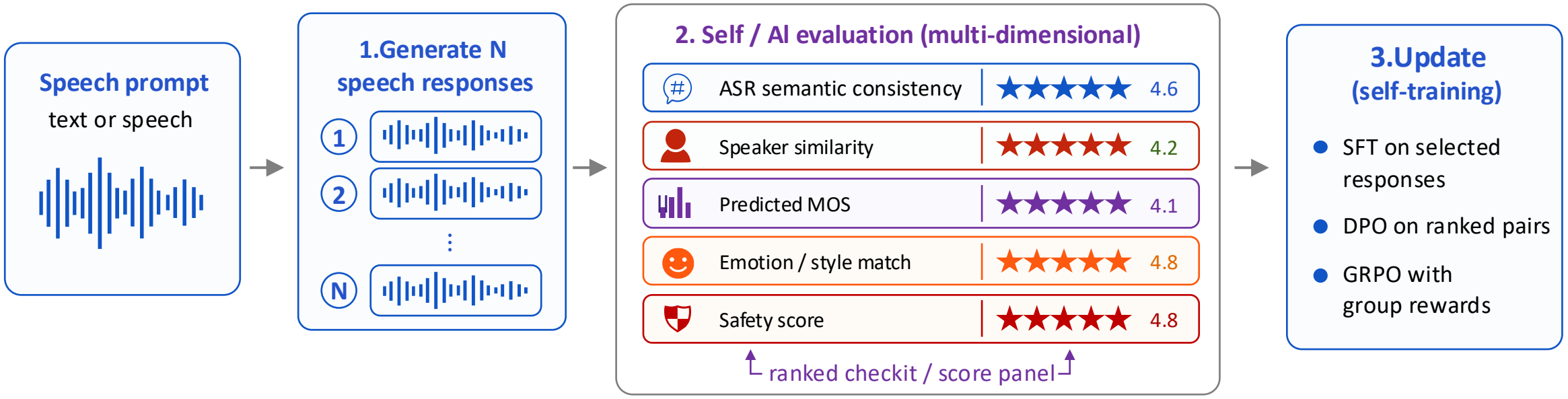
- training data is generated by the model itself

Diff.fromDPO /GRPO:

- preference or reward signal may also be self-generated

Online Self-Distillation for Speech LLMs

For Speech LLMs, self-distillation can use both semantic and acoustic feedback to improve speech responses.



$$Q(y) = \alpha Q_{\text{sem}}(y) + \beta Q_{\text{acoustic}}(y) + \gamma Q_{\text{spk}}(y) + \delta Q_{\text{safe}}(y)$$

$$\mathcal{D}^{\text{self}} = \{(x, y) | Q(y) > \tau\}$$

ASR = semantic consistency
MOS =mean opinion score
Spk = speaker similarity
Safe = safety evaluation

Comparison of Post-alignment Methods

Different alignment methods trade off supervision cost, training stability, and speech-specific evaluation difficulty.

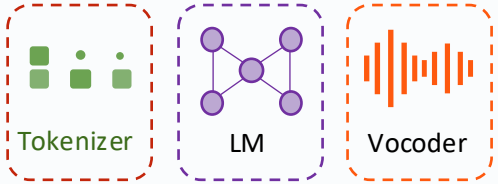
Method	Supervision	Optimization	SpeechLM Use	Limitation
SFT	high-quality demonstrations	maximum likelihood	align style, format, behavior	cannot express preference
PPO / RLHF	human preference (rank / pair)	RL with PPO + reward model	semantic quality, acoustic preference	expensive, unstable
DPO	preference pairs (chosen / rejected)	preference loss (no RL)	ranked speech responses	depends on pair quality
GRPO	group rewards (automatic)	group-relative policy update	ASR, MOS, spk sim, safety rewards	reward design is difficult
RLAIF	AI feedback (preference / score)	reward model or DPO / PPO	scalable preference collection	judge bias, misses acoustic nuance
Online Self-Distillation	self-generated + self-filtered	SFT / DPO / GRPO	continuous self-improvement	risk of self-reinforcing errors

Selection guideline: demonstrations → SFT; preference pairs → DPO;automatic rewards → GRPO; lower human cost →RLAIF / self-distillation.

Future Directions in SpeechLM Training

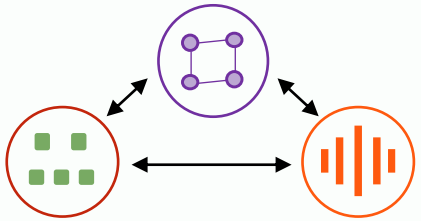
Future Speech LLM training will require more unified, efficient, and safety-aware training recipes.

Component-wise training
(current)



- tokenizer, LM, vocoder trained separately
- limited cross-objective optimization

Unified joint training



- joint tokenizer-LM-vocoder training
- semantic-acoustic co-optimization

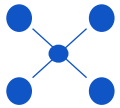
Aligned real-time SpeechLM

- full-duplex & low-latency
- rich instruction-following capability

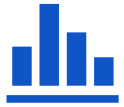
Safe speech agent
(future)

- safety, privacy, voice protection
- controllable self-improvement

$$L_{\text{total}} = L_{\text{AR}} + \lambda_1 L_{\text{align}} + \lambda_2 L_{\text{quality}} + \lambda_3 L_{\text{safety}}$$



end-to-end training



better semantic-acoustic modeling

real-time& full-duplex data

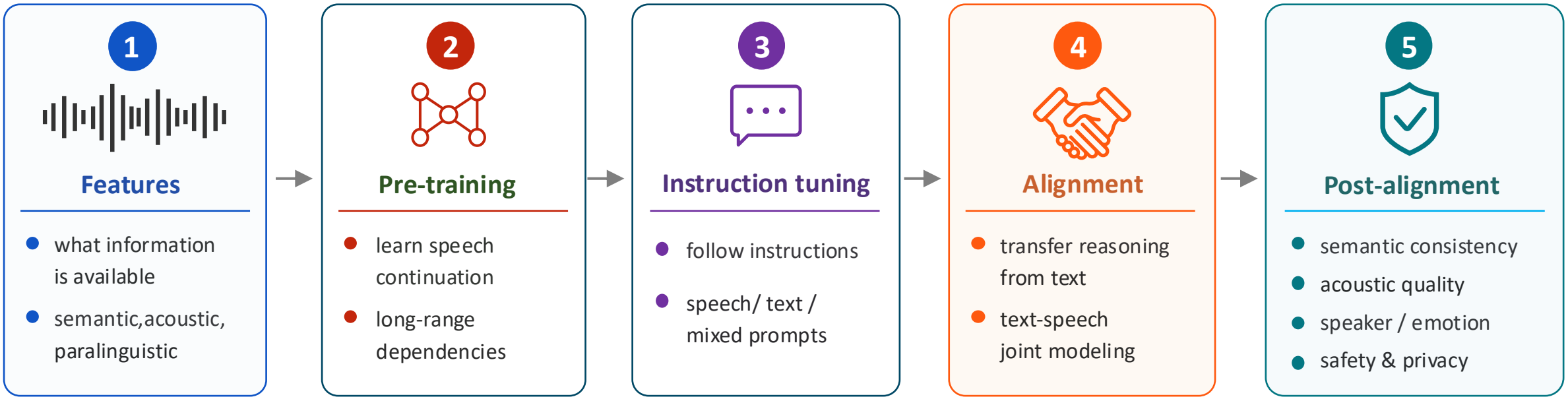
speech alignment benchmarks

voice safety & privacy

self-improving with feedback

Takeaways

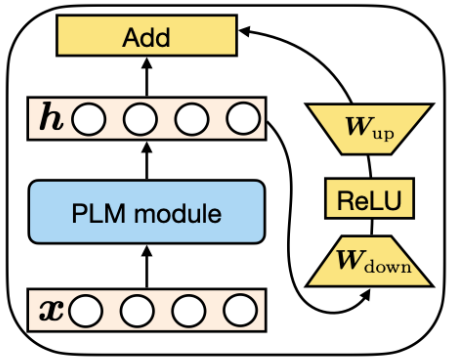
Speech LLM training recipes determine what the model can hear, understand, generate, and align with.



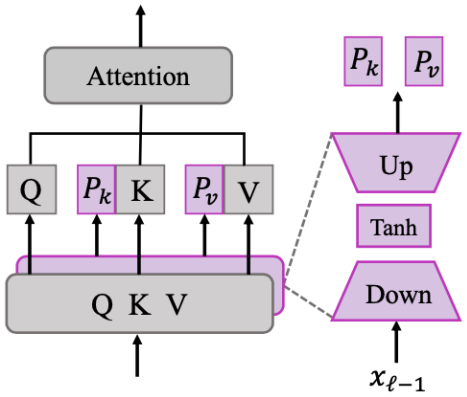
Goal: build capable, safe, and human-aligned speech agents

Parameter-efficient Fine Tuning

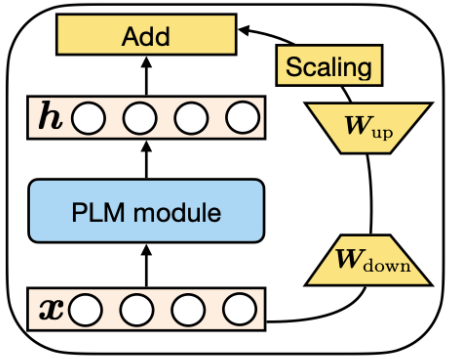
- Fine-tuning a pre-trained Transformer model (PTM) for speech applications in a parameter-efficient manner offers the dual benefits of **reducing memory** and leveraging the **rich feature representations** in massive **unlabeled datasets**.



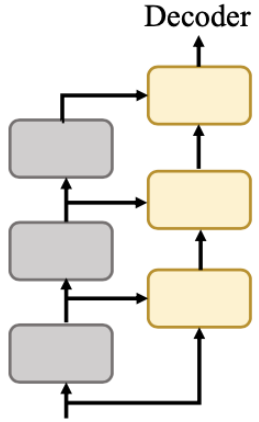
(a) Adapter



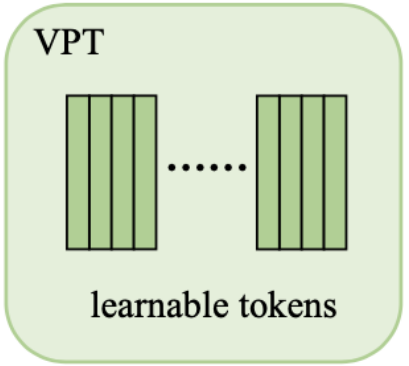
(b) Prefix tuning



(c) LoRA



(c) Side Tuning

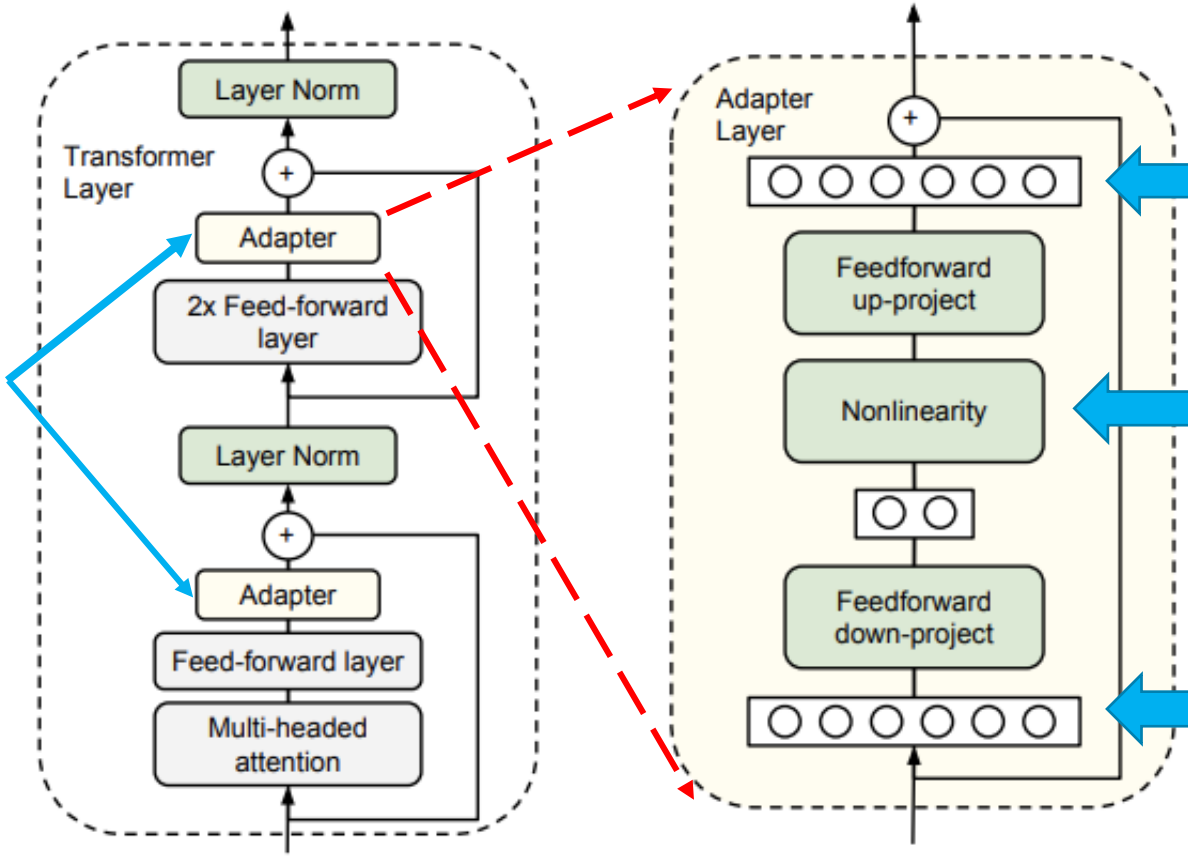


(c) Prompt Tuning

What is the best way in terms of effectiveness and efficiency to adapt speech LLM?

Adapter

Inserted serial adapter after each of the two sub-layers in the Transformer layer (attention layer, feedforward layer).



Project back to d-dimensions

Apply nonlinearity

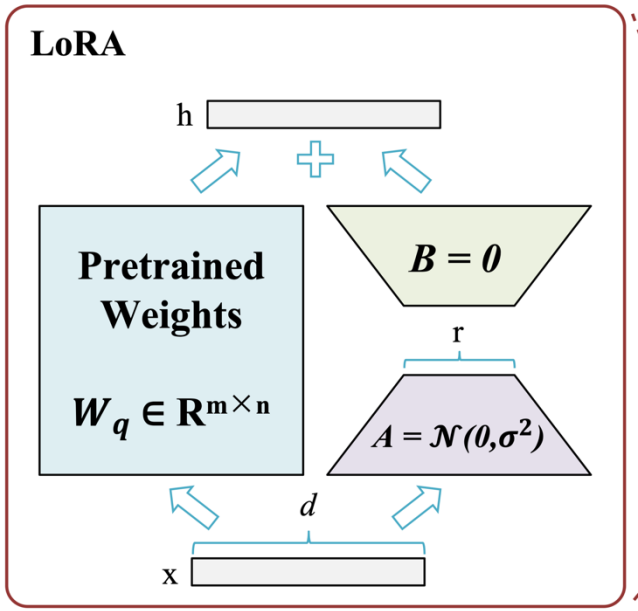
Adapters project original d-dimensional features into smaller dimension m.

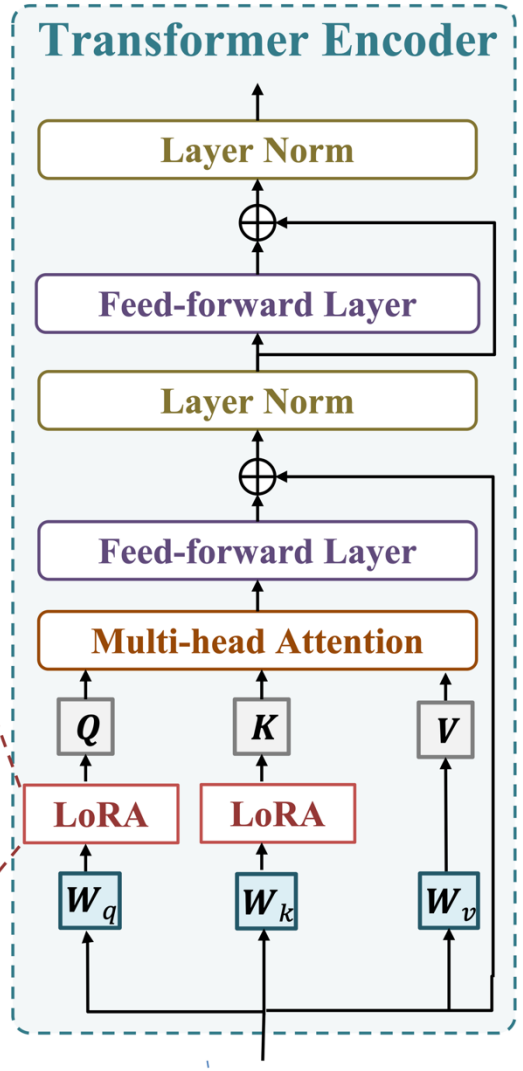
Bottleneck architecture to limit the number of parameters

LoRA

Simple and Effective

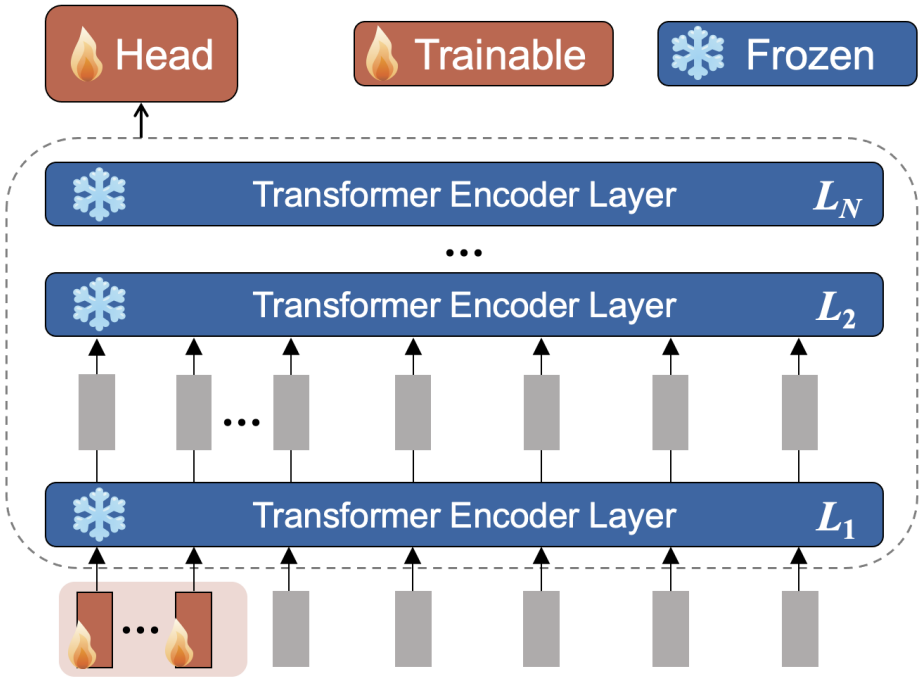
- Introduces trainable low-rank matrices to adapt pre-trained models.
- Only a small number of additional parameters are trained.
- Merged into the original weights at inference time with **zero additional latency**.





Prompt Tuning

- **Discrete Prompts (a.k.a. Hard prompts)**
  - Search for the **optimal combination** of tokens in Vocab.
  - Although it should be **human-readable and understandable**, it is difficult to perform well when searching in a discrete space.
- **Continuous Prompts (a.k.a. Soft prompts)**
  - The prompt doesn't need to be in natural language that **humans can understand**.
  - Special tokens (or virtual tokens) are created for the prompt to optimize in **continuous space.**



Soft Prompt VS. Discrete Prompts

Image Classification

Discrete Prompts	Classification Accuracy
a [CLASS].	82.68
a photo of [CLASS].	80.81
a photo of a [CLASS].	86.29
$[V]_1$ $[V]_2$ ... $[V]_M$ [CLASS].	**91.83**

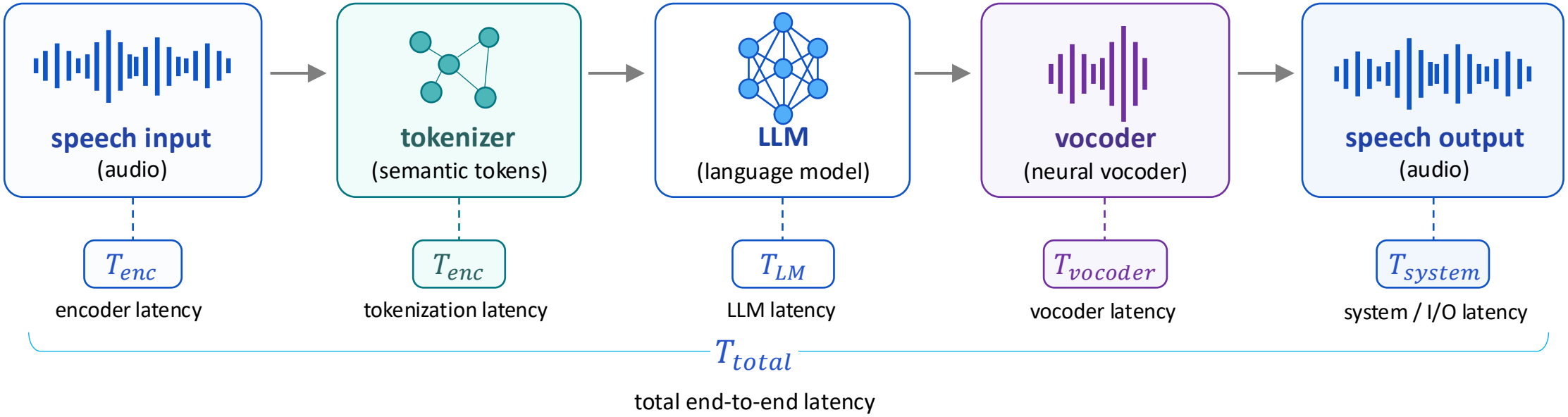
Soft Prompts

The blue block needs to use a held-out validation set for word tuning, which is inefficient; the green block automates the process and requires only a few labeled images for learning.

Zhou, K., Yang, J., Loy, C. C., & Liu, Z. (2022). Learning to prompt for vision-language models. International Journal of Computer Vision, 130(9), 2337-2348.

Why Speech LLM Acceleration Matters

Speech LLM acceleration is essential for real-time speech agents, not just faster decoding.



Real-time speech agents must listen, think, and speak.

Latency comes from multiple modules, not only the LM.

Natural turn-taking requires low response delay.

Total latency decomposition

$$T_{total} = T_{enc} + T_{LM} + T_{vocoder} + T_{system}$$

T_{enc} : encoder latency

T_{LM} :LLM latency

$T_{vocoder}$: vocoder latency

T_{system} : system / /O latency

Where the Bottlenecks Come From

Speech LLM latency is multi-stage and system-level, not just LM decoding.

Key bottleneck sources

- **Long speech token sequences**
  More tokens to generate and model.
- **Multi-codebook decoding**
  Multiple codebooks increase compute and memory.
- **Streaming constraints**
  Small chunks, causal attention, and real-time deadlines.
- **Vocoder overhead**
  High-fidelity neural vocoders are compute-intensive.
- **System scheduling**
  Queuing, batching, and resource contention add tail latency.

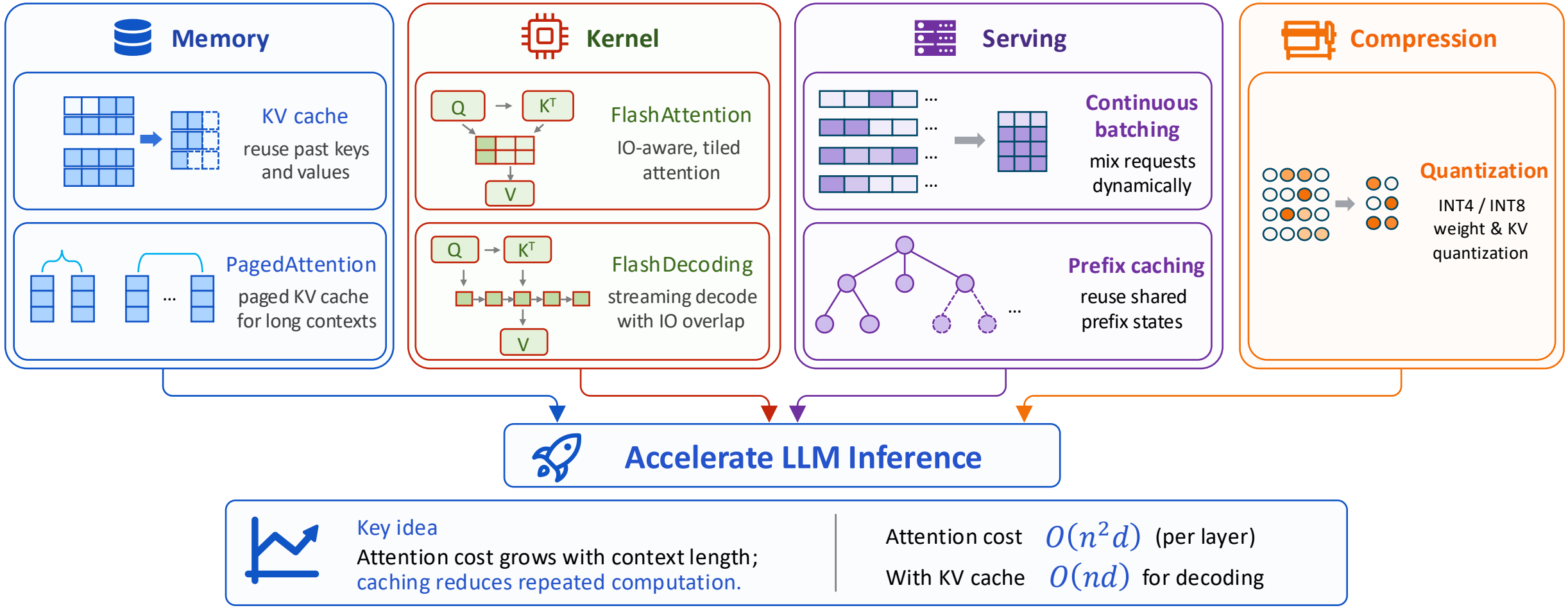
Tokenizer	Long token sequences from speech duration and multi-codebook quantization.
LM	High autoregresive cost and KV-cache pressure with long contexts.
Multi-stream	Cross-stream attention and multiple codebooks increase compute and memory.
Streaming	Small chunks, causal masks, and real-time constraints limit parallelism.
Vocoder	Neural vocoders (e.g., HiFi-GAN, SoundStream) are compute- and latency-heavy.
Serving	Queueing, batching, resource contention, and I/O overhead add tail latency.

Latency Accumulates across the entire stack

$$N_{tokens} \approx \text{duration} \times \text{token rate} \times \text{streams}$$

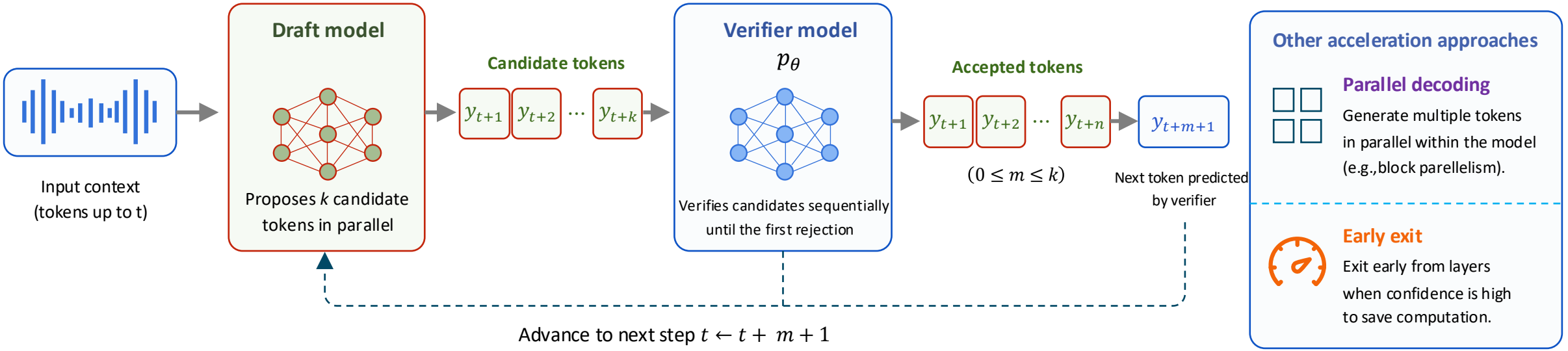
General LLM Acceleration Toolbox

LLM acceleration relies on KV cache, eficient attention, batching, and quantization.



Decoding Acceleration for Autoregressive Models

Autoregressive decoding is sequential; acceleration reduces step-by-step dependency.



- **Speculative decoding:** Draft model proposes k tokens; verifier accepts a prefix.
- **Parallel decoding:** Produce multiple tokens per step via intra-model parallelism.
- **Early exit:** Stop computation early when the next-token confidence is sufficient.
- **Draft-verify paradigm:** Balances speed (draft) and accuracy (verify).

Decoding objective

$$y_t : t + k \sim q_\phi, \ p_\theta \text{ verifies}$$

q_θ:draft (proposal)model

p_θ:verifier (target) model

k:draft length

Why Speech Breaks Text LLM Assumptions

Speech tokens are longer, multi-stream, and time-sensitive, making text-LM acceleration insufficient.

Text tokens

t_1 t_2 t_3 t_4 t_5 ⋯ t_n

Short sequence
(~10^2 tokens)

VS.

Speech tokens

s_1 s_2 s_3 ⋯ ⋯ ⋯ s_n

Very long sequence
(~104-106tokens)

VS.

Codec tokens (multi-codebook)

Level1	c_1^1	c_2^1	c_3^1	⋯	c_1^1
Level2	c_1^2	c_3^2	c_3^2	⋯	c_t^2
⋮	⋮	⋮	⋮		⋮
Level R	c_1^R	c_2^R	c_3^R	⋯	c_t^R

R parallel streams

T time steps

Shorter context

Much longer context

More streams & tighter latency

- Longer sequence length
- Codec multi-codebook streams
- Real-time constraints
- Streaming input/output coupling

Token count scaling

$$N_{codec} = T \times R$$

T : number of time steps

R : number of codebooks (streams)

- Speech needs much longer contexts.
- Multiple streams increase token throughput.
- Tighter latency budgets limit acceleration.
- Text-LM assumptions do not hold.

Tokenizer-level Acceleration

Reducing token rate is the most effective way to reduce Speech LLM latency.

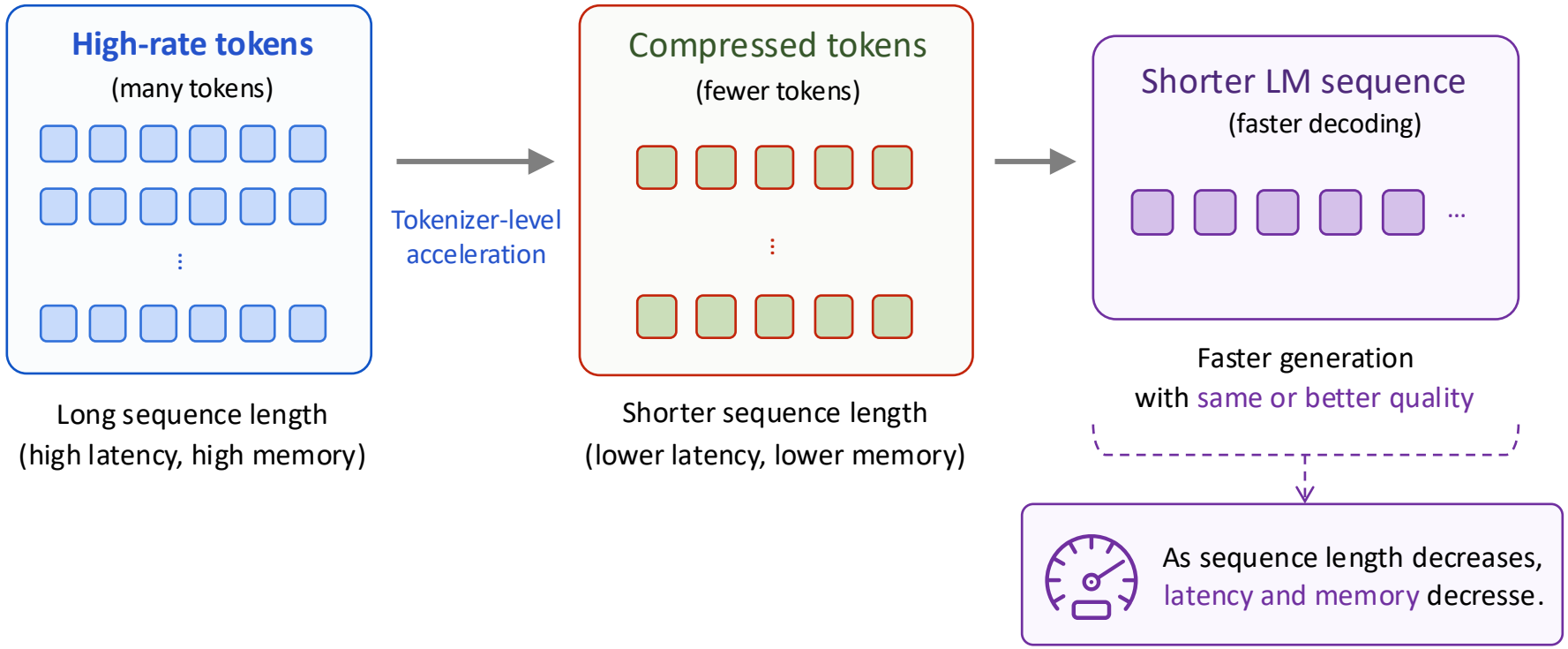
Why reduce tokens?

Lower frame rate
sample at a lower temporal frequency

Fewer codec streams
use fewer RVQ levels / codebooks

Semantic compression
retain mainly semantic information

Mixed token design
allocate more tokens to what matters, fewer to the rest



Compression ratio

$$compression\ \text{ratio} = \frac{N_{\text{original}}}{N_{\text{compressed}}}$$

$N_{original}$: number of tokens before compression
$N_{compressed}$: number of tokens after compression

Low-rate Speech Tokenizers

Modern speech tokenizers reduce latency by lowering token rate or unifying semantic and acoustic representations.

	SpeechTokenizer	Mimi	WavTokenizer
Token design (mini schematic)	Hierarchical semantic-acoustic tokens Semantic (1 token/sec) s_1 s_2 s_3 ⋯ s_T Acoustic (R tokens / sec) $a_{1,1}$ $a_{1,2}$ $a_{1,R}$ $a_{2,1}$ $a_{2,2}$	Low-latency codec tokens q_1 q_2 q_3 ⋯ q_T Codec tokens (very low rate)	Efficient acoustic tokens a_1 a_2 a_3 ⋯ a_T Acoustic tokens (low rate)
Latency orientation	Low rate via semantic-acoustic hierarchy	Low rate via discrete codec bottleneck	Low rate via efficient acoustic quantization
Real-time suitability	High (token rate ≈1-10 tokens/s)	High (token rate ≈ 1–5 tokens/s)	High (token rate ≈5-25 tokens/s)

All three approaches reduce LM sequence length (token rate) to lower latency and improve real-time usability.

- SpeechTokenizer: hierarchical semantic-acoustic design
- Mimi: low-latency real-time codec
- WavTokenizer: efficient acoustic tokenizer
- Key idea: reduce LM sequence length

Unified view (SpeechTokenizer)

$$s^{mixed} = \left[s^{sem} \; ; \; s^{acoustic}_{(1:R)} \right]$$

s^{sem} : semantic tokens (1 / sec)

$s^{acoustic}_{(1:R)}$: acoustic tokens (R / sec)

R : acoustic refinement factor

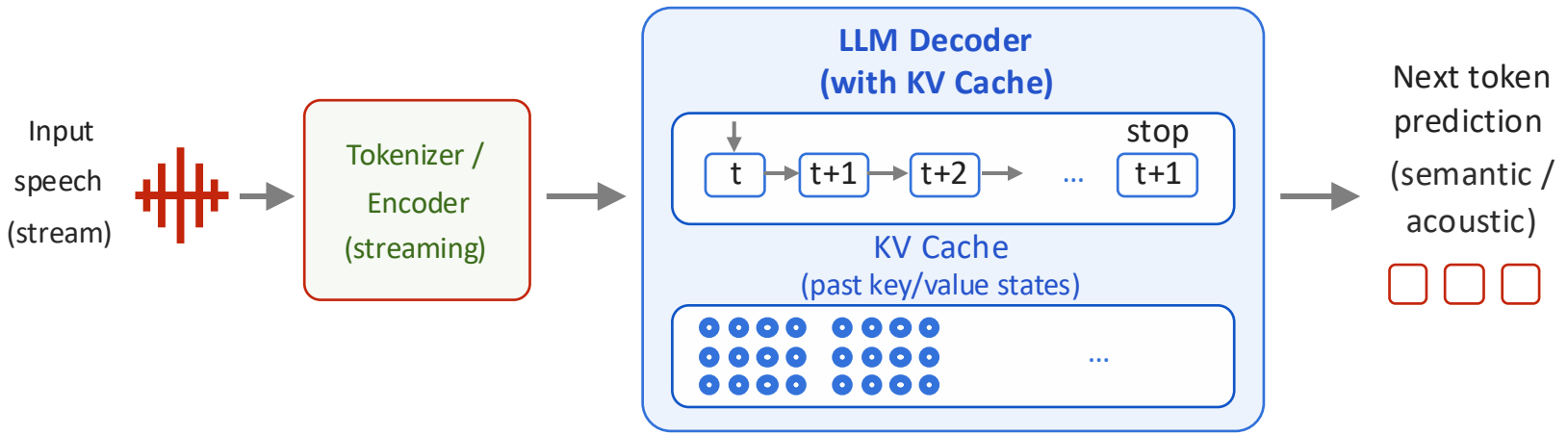
LM Decoding-level Acceleration for Speech Tokens

Speech-token generation can reuse LLM decoding acceleration, but long context and multi-stream tokens make KV cache management more complex.

Key techniques

- KV cache for streaming
- FlashAttention / FlashDecoding
- Speculative decoding adapted to speech tokens
- Multi-stream tokens require synchronization

Streaming speech-token generation witleh KV cache



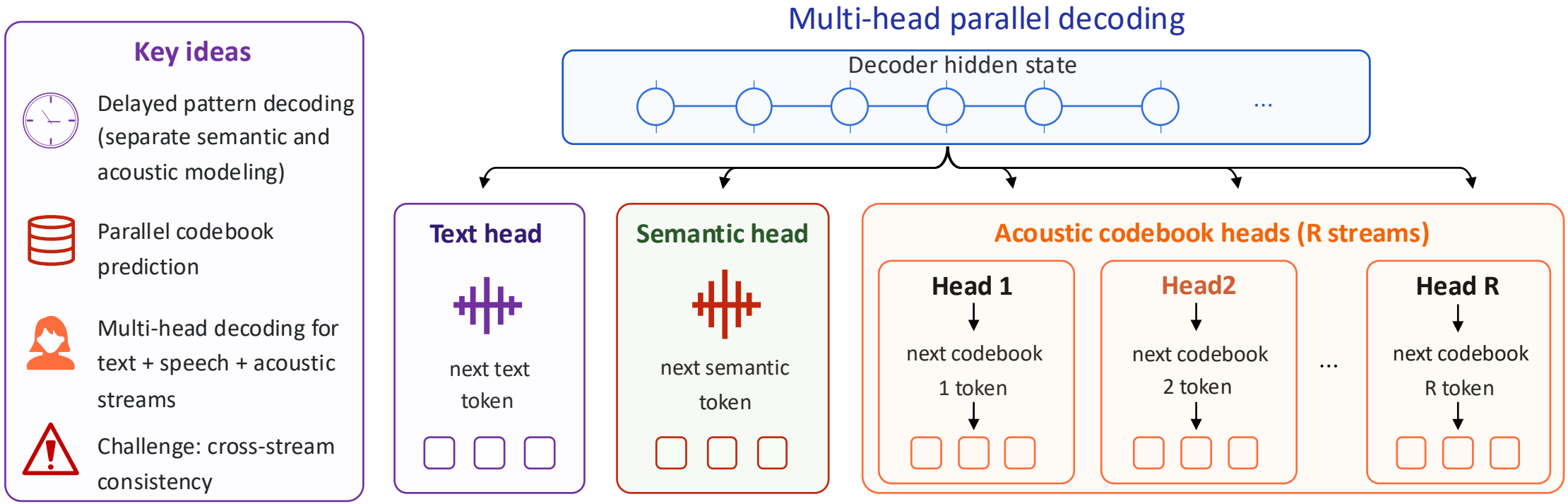
KV cache memory cost scaling

$$M_{KV} \propto L \times T \times d$$

L : number of layers
T : context length (speech tokens)
d : hidden dimension

Multi-stream Generation Acceleration

Speech LLMs can exploit structured parallelism across semantic, acoustic, and codec codebook streams to reduce sequential decoding bottlenecks.



Joint distribution factorization

$$p(y) = p(y^{text}) \cdot (y^{sem}) \cdot \prod_{r=1}^{R} p\left(y_r^{acoustic}\right)$$

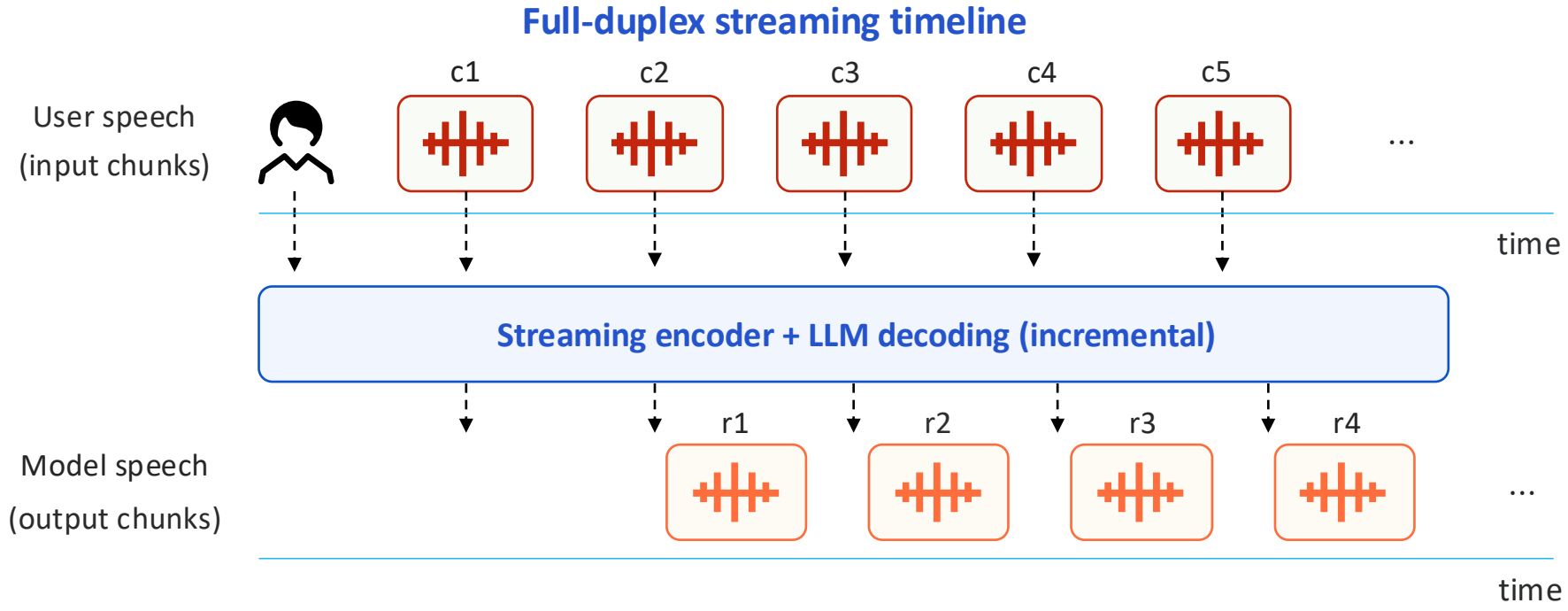
R: number of codebook streams (Parallel prediction reduces sequential dependency)

Streaming Interaction Acceleration

Real-time Speech LLMs require streaming-native inference with incremental processing and overlapping input-output.

Key components

- Chunk-wise speech encoding
- Incremental decoding for early response
- Full-duplex interaction (listen while speak)
- Overlap of input listening and output speaking



Response latency model

$$T_{response} \approx T_{chunk} + T_{decode} + T_{audio}$$

T_{chunk} : chunk processing latency
T_{decode}: decoding latency
T_{audio} : codec/vocoder latency

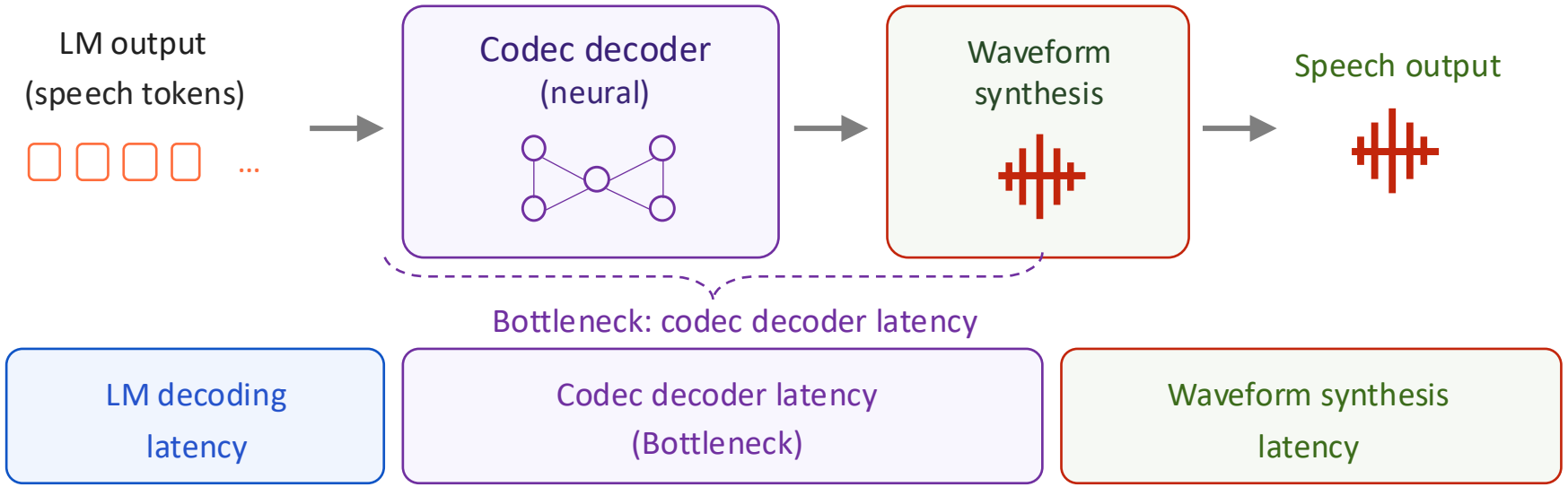
Vocoder / Codec Decoder Acceleration

Even with fast LM decoding, vocoder / codec decoding can dominate end-to-end speech latency

Key points

- Neural codec decoder is non-trivial cost
- Streaming/ causal codec enables real-time synthesis
- Non-autoregressive vocoder reduces latency
- Trade-off between audio quality and latency

Audio synthesis pipeline



Audio latency decomposition

$$T_{audio} = T_{codec} + T_{synthesis}$$

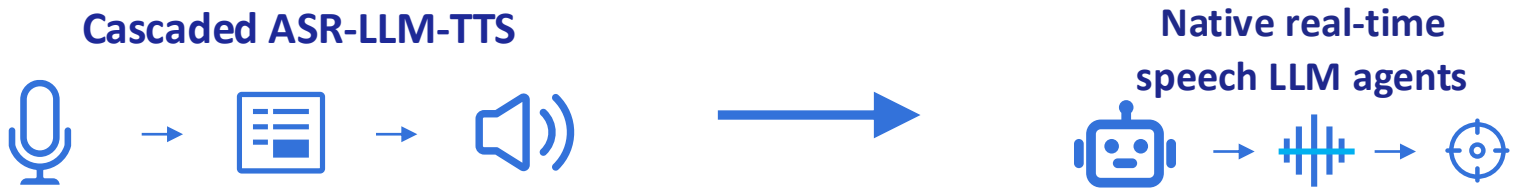
T_{codec} : codec decoding latency

$T_{synthesis}$: waveform synthesis latency

Case Study: Real-time Speech LLM Systems

Recent systems achieve low latency via streaming, low-rate tokens, and parallel decoding, moving toward native speech agents.

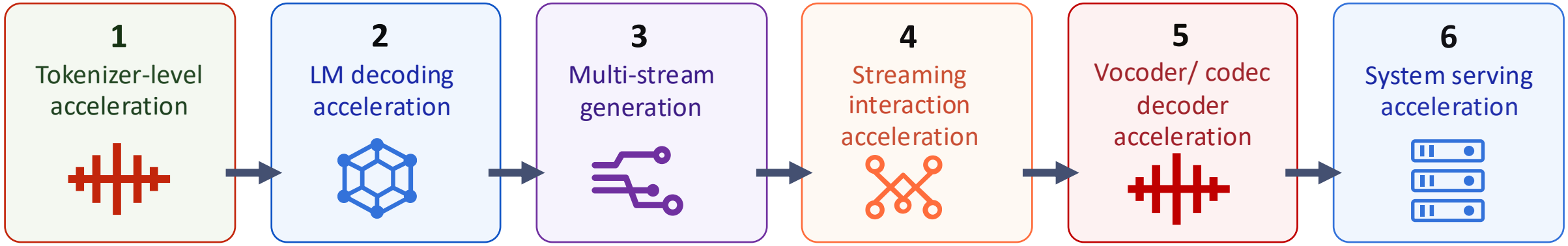
Model	Mini-Omni (2024)	LLaMA-Omni (2024)	Moshi (2024)
Streaming	✓ (streaming)	✓ (streaming)	✓ (streaming)
Full-duplex	Partial (listen→speak)	✓ (near full-duplex)	✓ (full-duplex)
Token / Codec	Discrete speech tokens	Speech-text interleaved	Mimi codec tokens (low-rate)
Key Design	Parallel text & speech generation	Inner monologue + speech decoding	Real-time dialogue Multi-stream decoding
Latency Focus	Low token rate Incremental decoding	Streaming interaction Multi-stream heads	Low-latency codec Overlapl/0



Speech LLM Acceleration Roadmap

Speech LLM acceleration requires joint optimization across multiple system layers to achieve real-time interaction.

Six-layer acceleration roadmap



1	2	3	4	5	6
Lower token rate Better compression	KV cache Fast attention / Speculative-Parallel	Parallel codebooks Multi-head decoding / Consistency	Chunking Incremental decode / Full-duplex	Causal codec Lightweight vocoder/ Audio streaming	Batching Caching / Distributed serving

End-to-end latency decomposition

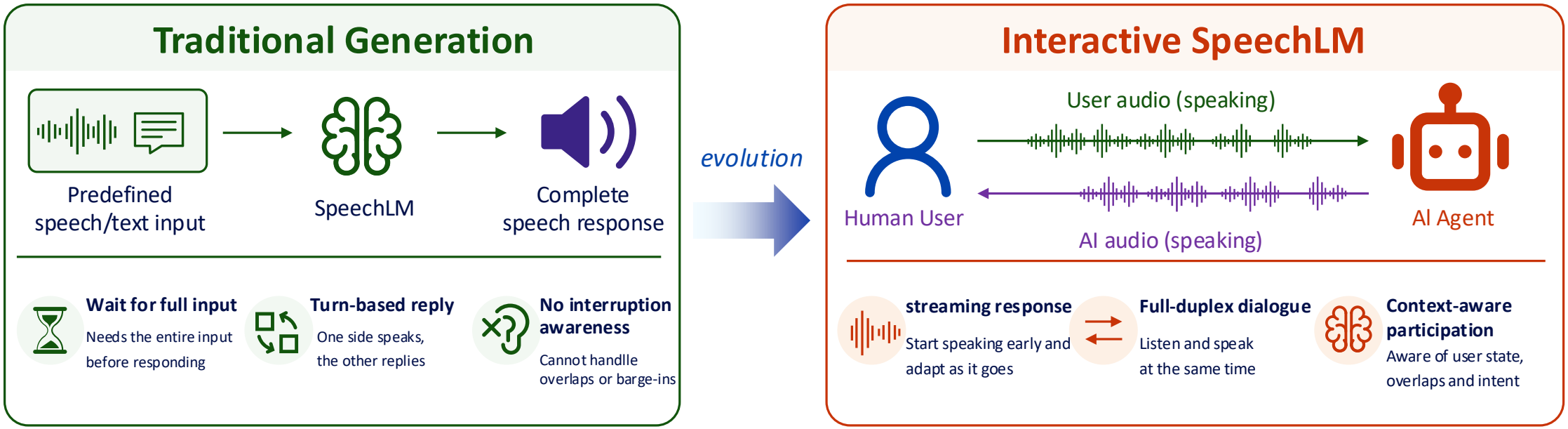
$$T_{total} = T_{tokenizer} + T_{LM} + T_{stream} + T_{vocoder} + T_{system}$$

Each layer contributes to the final response latency.

Part IV Interaction Capabilities

Speech Interaction Paradigm

From traditional speech generation to natural spoken interaction



$$\text{Response}_t = SpeechLM(Context_{<=t})$$

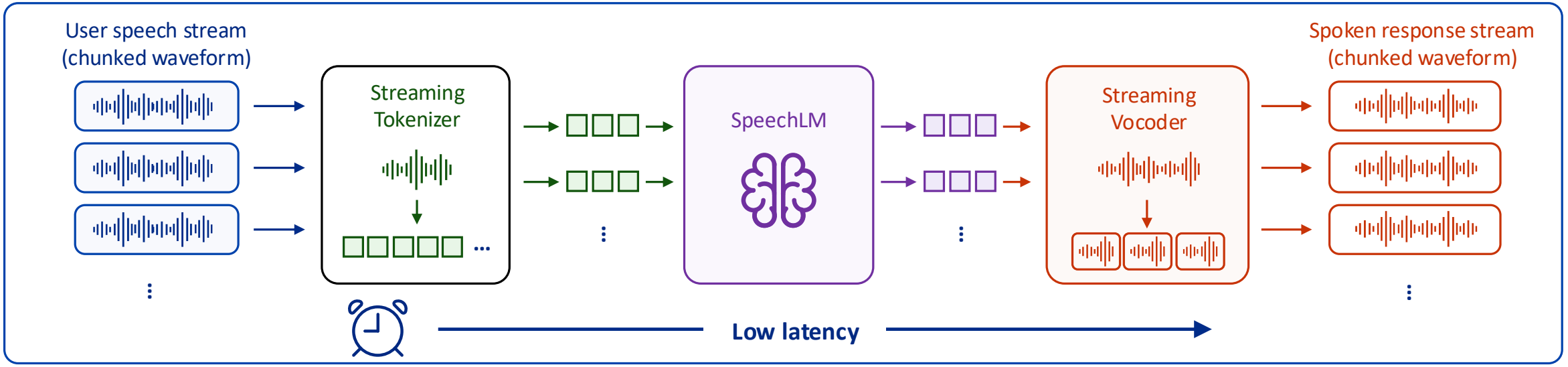
Two key capabilities

1. **Real-time Interaction**
2. **Interactive Period Recognition (IPR)**

Speech LLMs should not only generate speech, but also interact through speech in real time.

Real-time Interaction

Streaming modules enable low-latency speech response



No need to wait
for full utterance

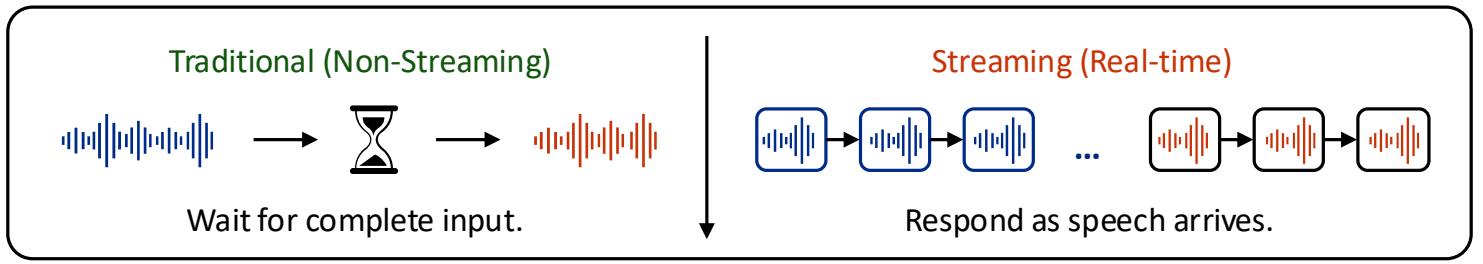
Chunk-by-chunk
encoding and synthesis

Fast, adaptive response
to ongoing speech

Streaming Computation (Conceptual)

$$a_{\leq=t} = V_O\left(LM\left(Tok\left(x_{\leq=t}\right)\right)\right)$$

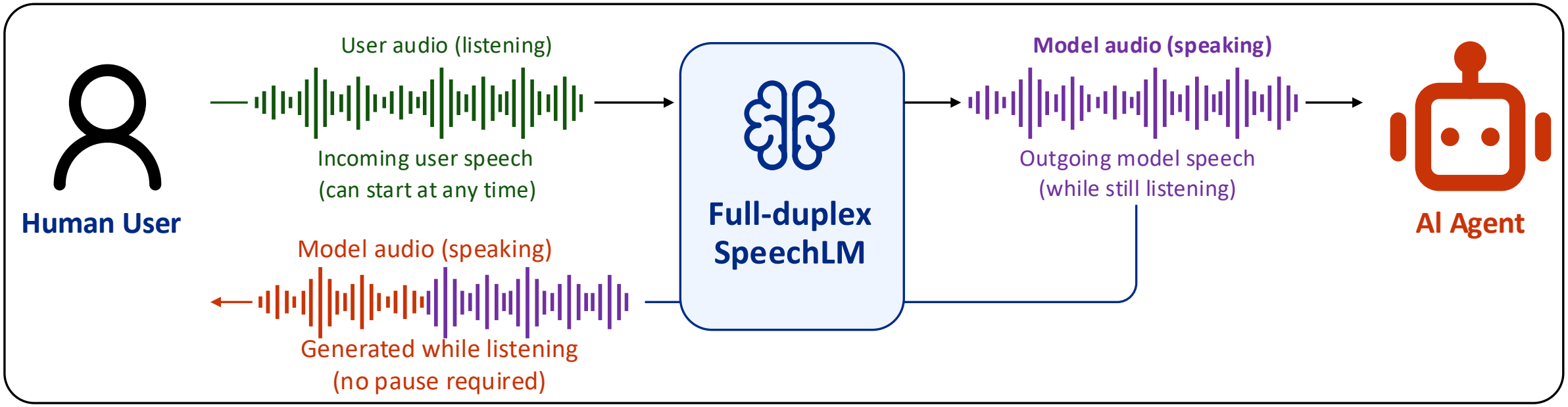
Streaming output depends on partial input context.



Streaming tokenizer and vocoder are the first step toward real-time Speech LLM interaction.

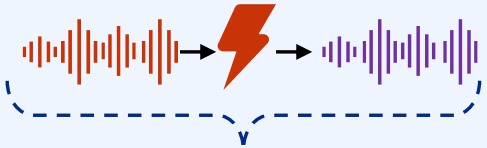
Full-duplex Interaction

A Speech LLM should listen and speak at the same time



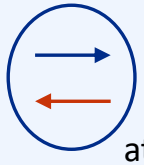
1 User Interruption

The user can interrupt the model while it is speaking.



2 Simultaneous Response

The model keeps listening to the user while generating speech.



Listen and Speak at thesame time

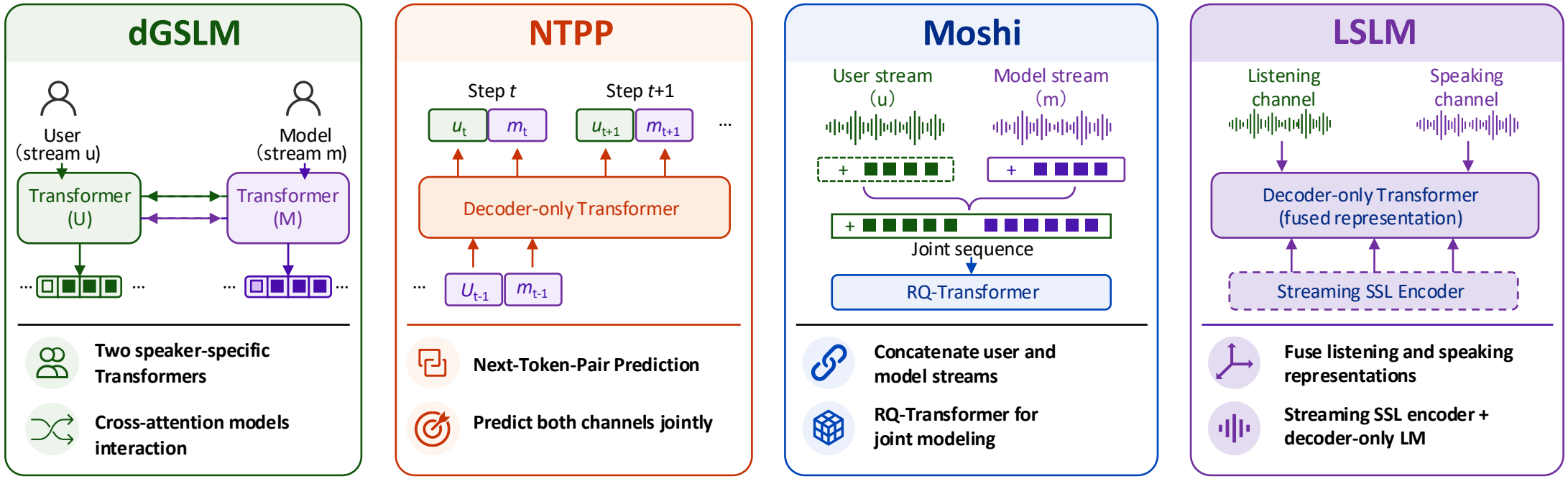
$$(u_t, m_t) \sim LM(u_{<t}, m_{<t})$$

Joint modeling of user and model audio streams

Full-duplex modeling moves Speech LLMs from turn-based response to natural conversational dynamics.

Representative Modeling Strategies

Different methods model multi-stream speech interaction in different ways



$$(s_t^u, s_t^m) = LM(context_t)$$

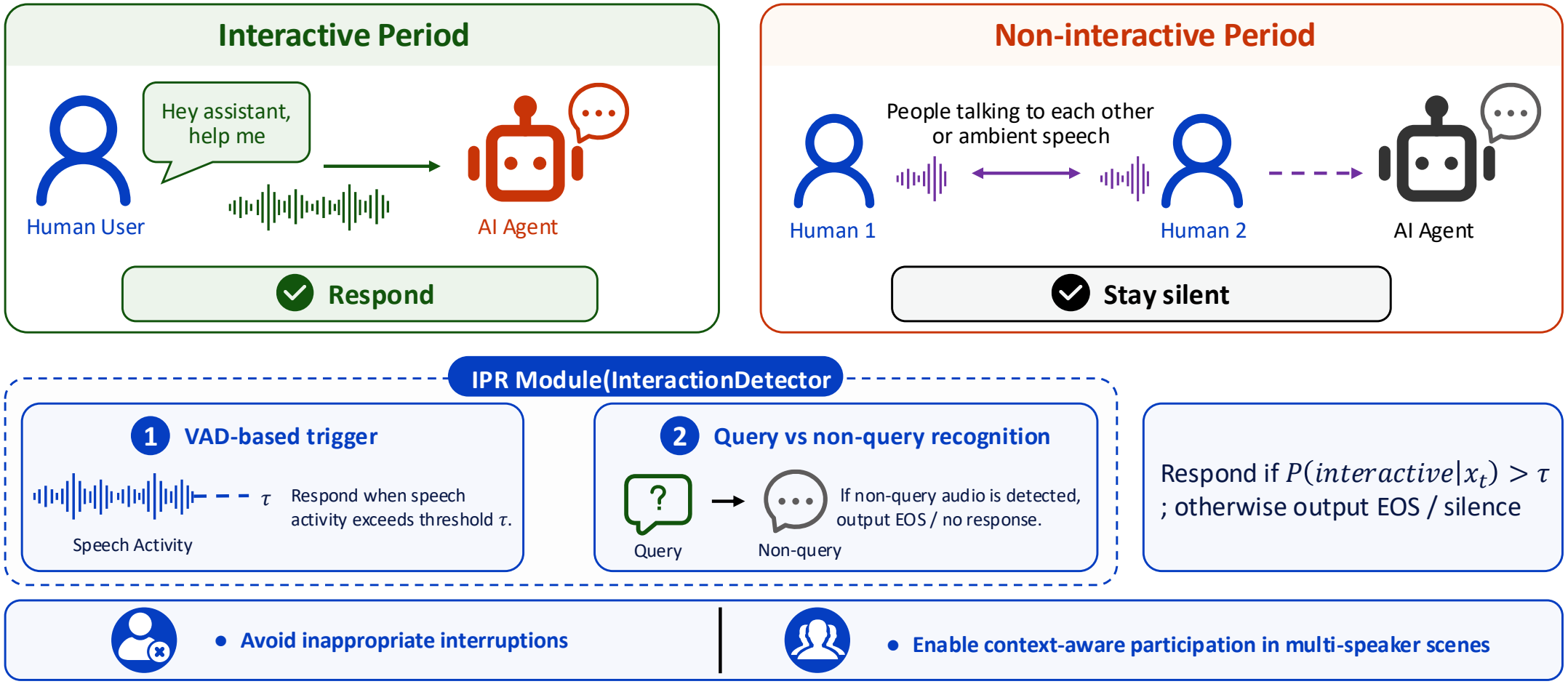
Core idea: jointly model multiple speech streams over time.

multi-stream | joint prediction | low-latency dialogue

Although the architectures differ, the common goal is joint modeling of interaction over time.

Interactive Period Recognition (IPR)

A Speech LLM should know when to respond - and when to remain silent



IPR makes Speech LLMs more natural and socially aware by speaking only at the right time.

Afternoon Tea Break — 30 minutes

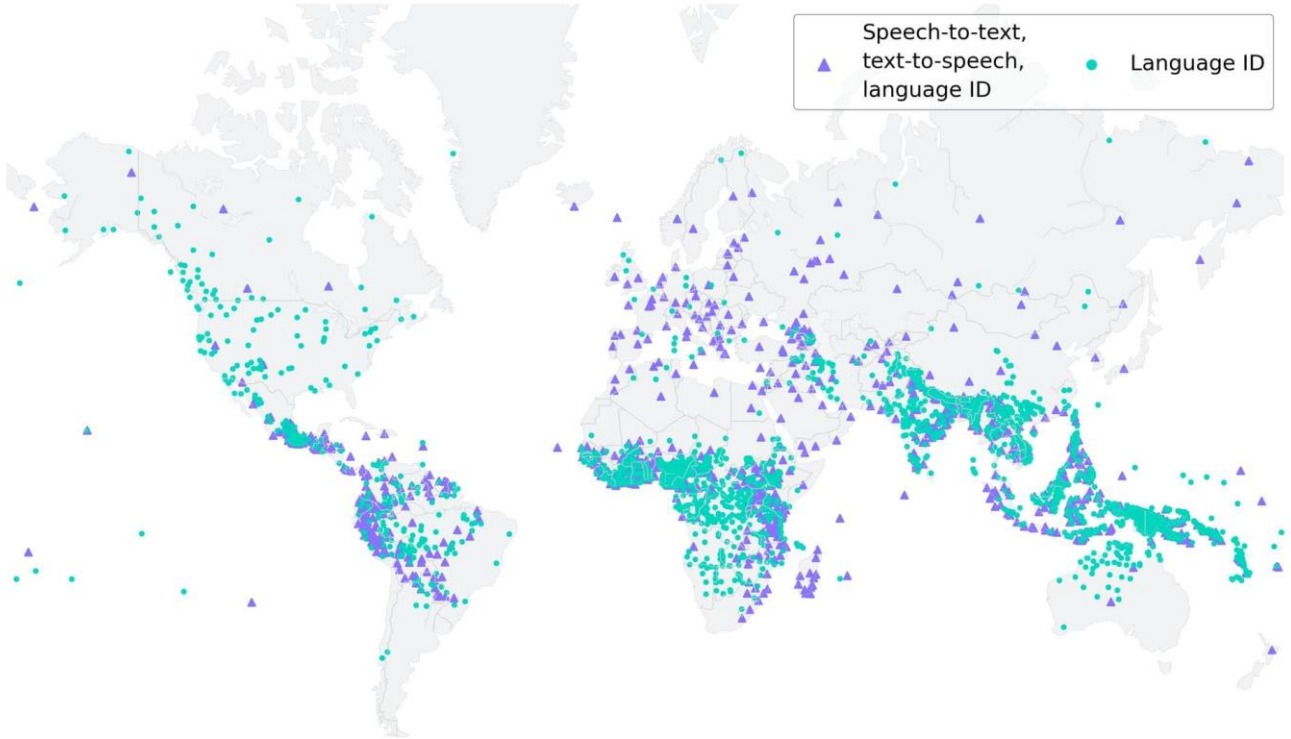
Session Resumes at 4:00 p.m.

Part V

Under-Resourced Speech LLMs

What Changes Under Scarcity?

The multilingual world is much larger than the multilingual training set.



But what happens when the target language has very little data?

Vineel Pratap, Andros Tjandra, Bowen Shi, et al., "Scaling Speech Technology to 1,000+ Languages," *Journal of Machine Learning Research*, vol. 25, no. 97, pp. 1–52, 2024.

Low Resource Is Not One Problem

When we use the term low-resource,
it is important to understand that it does not refer to only one type of scarcity.

Different types of scarcity require different solutions.

Arun Babu, Changhan Wang, Andros Tjandra, et al., "XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale," *Proc. Interspeech*, pp. 2278–2282, 2022.

Case Studies: Uyghur, Kazakh and Kyrgyz

Why Uyghur, Kazakh and Kyrgyz?

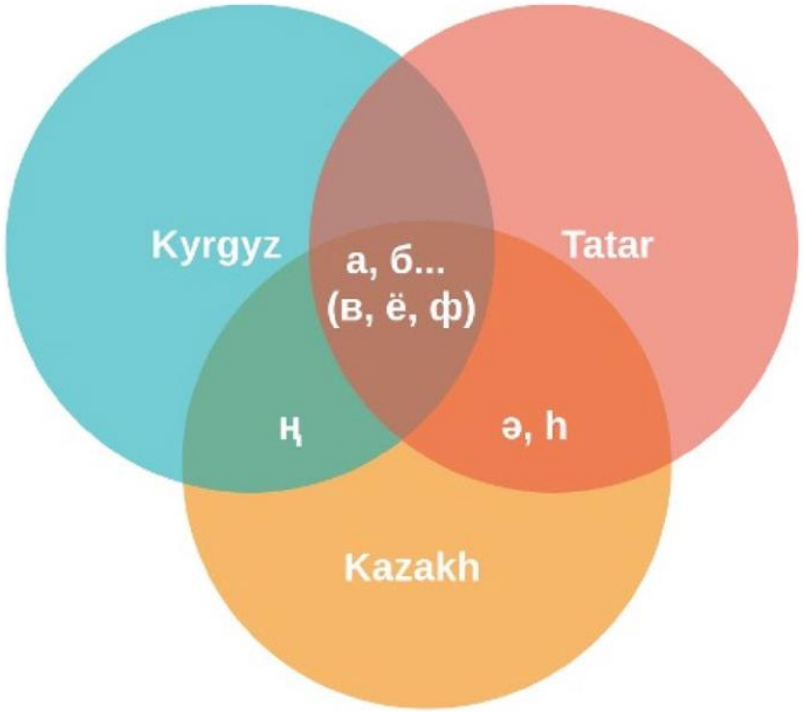


Figure 1. Example of common letters for some of the languages included in the experiments.

These languages provide useful low-resource case studies because they combine:

- Agglutinative morphology
- Rich word-form variation
- Limited annotated speech
- Multiple writing conventions or script histories
- Strong linguistic relations that enable transfer

Low-resource does not mean linguistically simple.

Akbayan Bekarystankyzy, Orken Mamyrbayev, Mateus Mendes, et al., "Multilingual End-to-End ASR for Low-Resource Turkic Languages with Common Alphabets," *Scientific Reports*, vol. 14, Article 13835, 2024.

Morphological Richness Creates Data Sparsity

One major challenge in these languages is morphological richness:

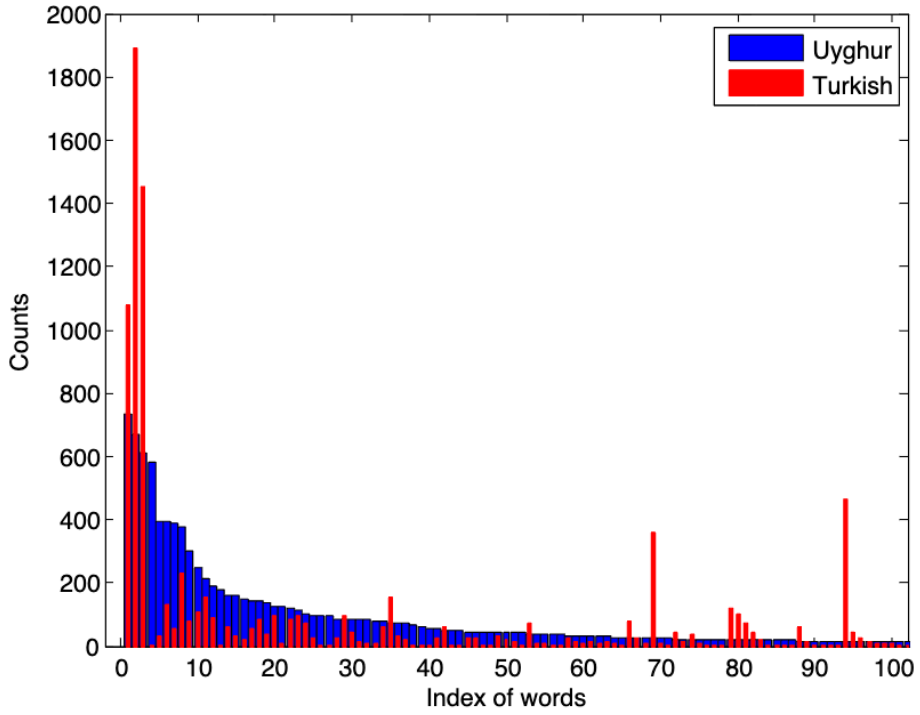


Figure 1: Counts of the 100 most frequent Uyghur words in the training data and overlap with Turkish data

Morphological Richness Creates Vocabulary Sparsity

Agglutinative languages construct words by combining:

Stem + Suffix$_1$ + Suffix$_2$ + …

This creates:

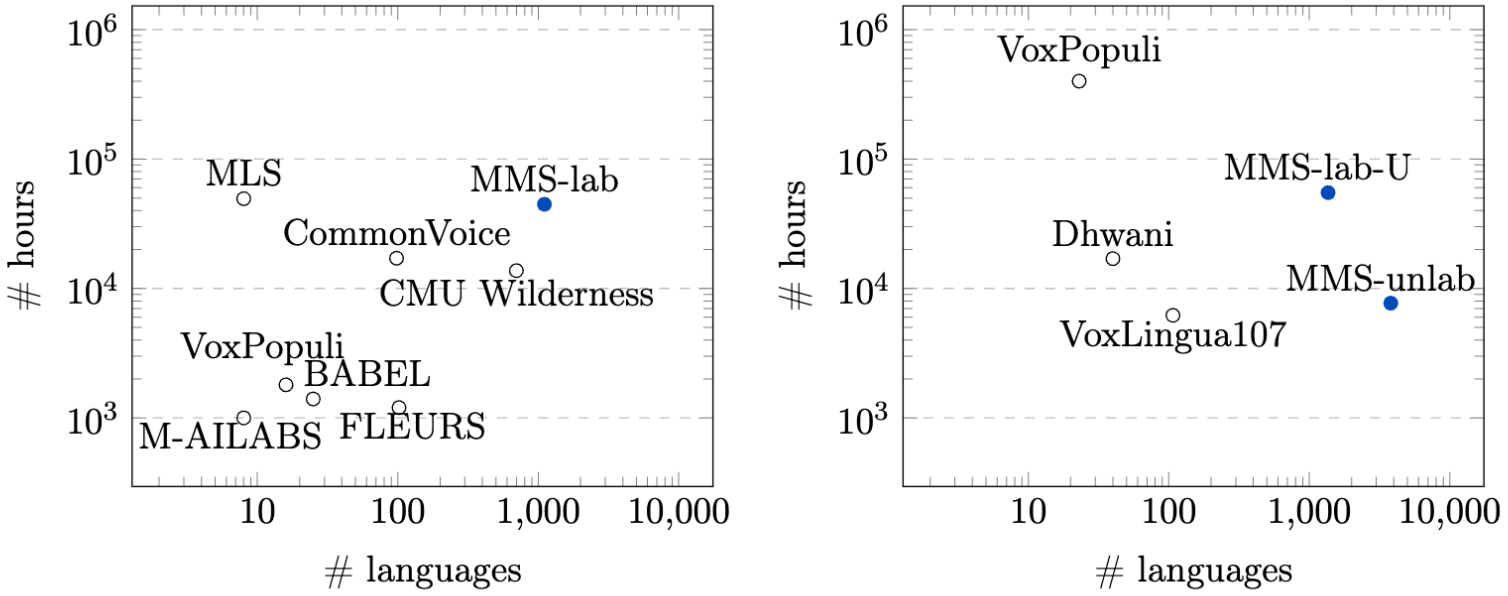
- Many valid surface forms
- High out-of-vocabulary rates
- Fewer observations per word form
- Difficulties for word-level language modeling

A small corpus must cover a much larger lexical space.

Ayimunishagu Abulimiti and Tanja Schultz, "Building Language Models for Morphological Rich Low-Resource Languages Using Data from Related Donor Languages: The Case of Uyghur," *Proc. SLTU-CCURL*, pp. 271–276, 2020.

Are Speech LLMs Automatically Better Than Cascades?

The two modalities may exist separately, but not as aligned pairs.



(a) Labeled Datasets.

(b) Unlabeled Datasets.

Figure 2: **Dataset Overview.** MMS-lab, MMS-lab-U and MMS-unlab compared to existing multilingual speech corpora in terms of the supported languages and dataset size. We compare to BABEL (Gales et al., 2014), CMU Wilderness (Black, 2019), CommonVoice (Ardila et al., 2020), Dhwani (Javed et al., 2022), FLEURS (Conneau et al., 2022), M-AILABS (M-AILABS, 2018), MLS (Pratap et al., 2020c), VoxLingua107 (Valk and Alumäe, 2020), and VoxPopuli (Wang et al., 2021) .

Vineel Pratap, Andros Tjandra, Bowen Shi, et al., "Scaling Speech Technology to 1,000+ Languages," *Journal of Machine Learning Research*, vol. 25, no. 97, pp. 1–52, 2024.

Are Speech LLMs Better in Low-Resource Settings?

Language	ASR (Whisper) + MT (NLLB)						Multimodal Speech Translation		Audio LLMs	
	Cascaded Baseline (T1)	Cascaded Mono. (T2)	Cascaded Multi. (T3)	Cascaded Multi. + Mono. (T4)	Cascaded Multi. + ASR Corrector (T5)	T4 + ASR Corrector (T6)	SeamlessM4T Baseline	SeamlessM4T Multi.	GPT-4o audio	Gemini 2.0 Flash
French	**38.30**	31.49	x	x	x	x	33.77	x	37.49	36.16
Hindi	27.79	31.18	30.85	30.90	31.08	31.48	24.62	28.71	29.28	**35.38**
Punjabi	13.87	26.59	26.65	26.68	25.38	24.62	28.71	28.10	19.15	**29.58**
Tamil	19.53	22.65	22.48	22.78	23.10	23.43	19.93	21.87	15.17	**25.14**
Telugu	17.51	25.12	25.26	25.15	27.37	27.53	23.27	24.93	19.83	**30.78**
Malayalam	1.32	27.07	27.05	27.68	27.79	28.45	21.20	25.95	23.55	**30.31**
Swahili	25.01	27.55	27.18	27.70	28.40	28.38	14.81	31.22	19.37	**31.91**
Hausa	3.36	18.45	15.90	18.34	18.04	**20.05**	1.01*	6.07	1.07	16.29
Yorùbá	2.62	11.14	10.66	11.04	10.62	11.18	12.64	**15.36**	2.31	7.35
Igbo	1.80*	11.46	9.86	11.60	13.23	**13.93**	0.19	11.65	1.63	2.19
Luganda	4.07*	11.36	11.34	11.56	13.14	13.35	5.95	**18.92**	4.95	11.93
Average	11.69	21.26	20.72	21.34	21.82	**22.24**	15.23	21.28	13.63	22.09

[a] Starred (*) BLEUs indicate that the target languages were unseen by the model. **Bolded BLEUs** indicate the best score across different finetuning strategies and baseline.

Table 4: Overview of **BLEU** scores (↑) achieved by SOTA models with different finetuning strategies. please refer to **Section** 3.3 for definitions of finetuning strategies, and **Section** 4.1 for detailed analysis.

Direct speech LLMs may have weak acoustic coverage for unseen language.
Strong specialized ASR models can remain competitive Domain mismatch can dominate architecture choice
End-to-end is not automatically synonymous with low-resource robustness.

Jonah Dauvet, Min Ma, Jessica Ojo, and David Ifeoluwa Adelani, "Reassessing Speech Translation for Low-Resource Languages: Do LLMs Redefine the State-of-the-Art Against Cascaded Models?," *Proc. MRL*, pp. 149–160, 2025.

Self-Supervised Learning: Learn Before Transcription

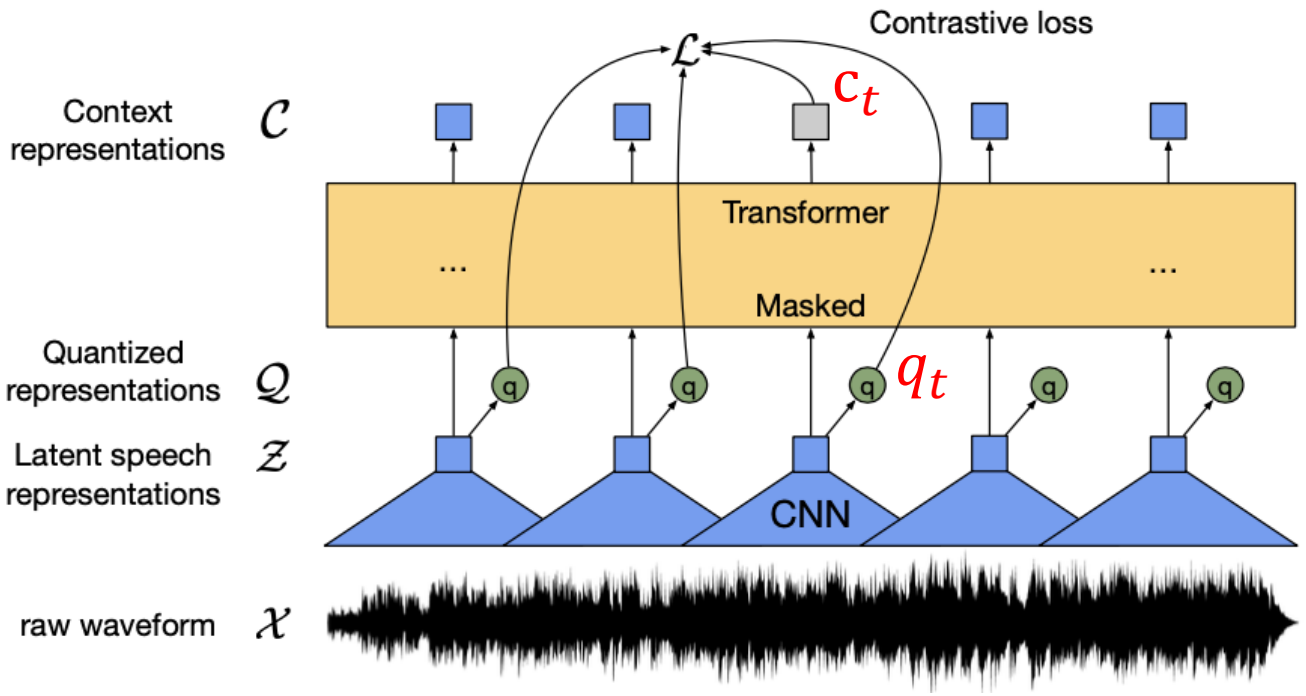


Fig . Illustration of Wav2Vec 2.0.

Mask-based Contrastive Loss:

Context representation

Discrete latent speech representation

$$\mathcal{L}_m = -\log \frac{\exp(sim(\mathbf{c}_t, \mathbf{q}_t)/\kappa)}{\sum_{\tilde{\mathbf{q}} \sim \mathbf{Q}_t} \exp(sim(\mathbf{c}_t, \tilde{\mathbf{q}})/\kappa)}$$

Distractors $\tilde{q} \in Q_t$

Diversity Loss:

$$\mathcal{L}_d = \frac{1}{GV} \sum_{g=1}^{G} -H(\bar{p}_g) = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} \bar{p}_{g,v} \log \bar{p}_{g,v}$$

Entropy of $p_{g,v}$

$$p_{g,v} = \frac{\exp(l_{g,v} + n_v)/\tau}{\sum_{k=1}^{V} \exp(l_{g,k} + n_k)/\tau}$$

It reduces the need for labels, but it still requires **raw speech**

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli, "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

From Monolingual SSL to Cross-Lingual SSL

XLS-R pre-trains one shared representation model on:

1. 128 languages 2. Nearly half a million hours of speech 3. Highly diverse resource

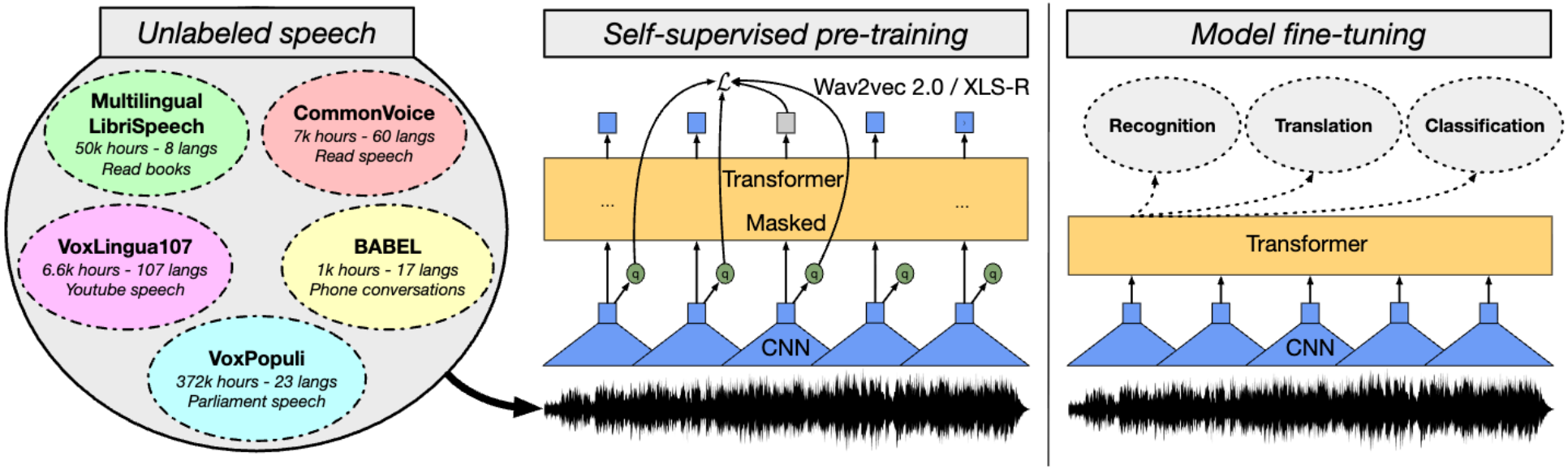


Figure 1: **Self-supervised cross-lingual representation learning.** We pre-train a large multilingual wav2vec 2.0 Transformer (XLS-R) on 436K hours of unannotated speech data in 128 languages. The training data is from different public speech corpora and we fine-tune the resulting model for several multilingual speech tasks.

A low-resource language no longer needs to learn every representation from scratch.

Arun Babu, Changhan Wang, Andros Tjandra, et al., "XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale," *Proc. Interspeech*, pp. 2278–2282, 2022.

Multilingual Transfer Is Also a Sampling Problem

Multilingual ≠ Balanced

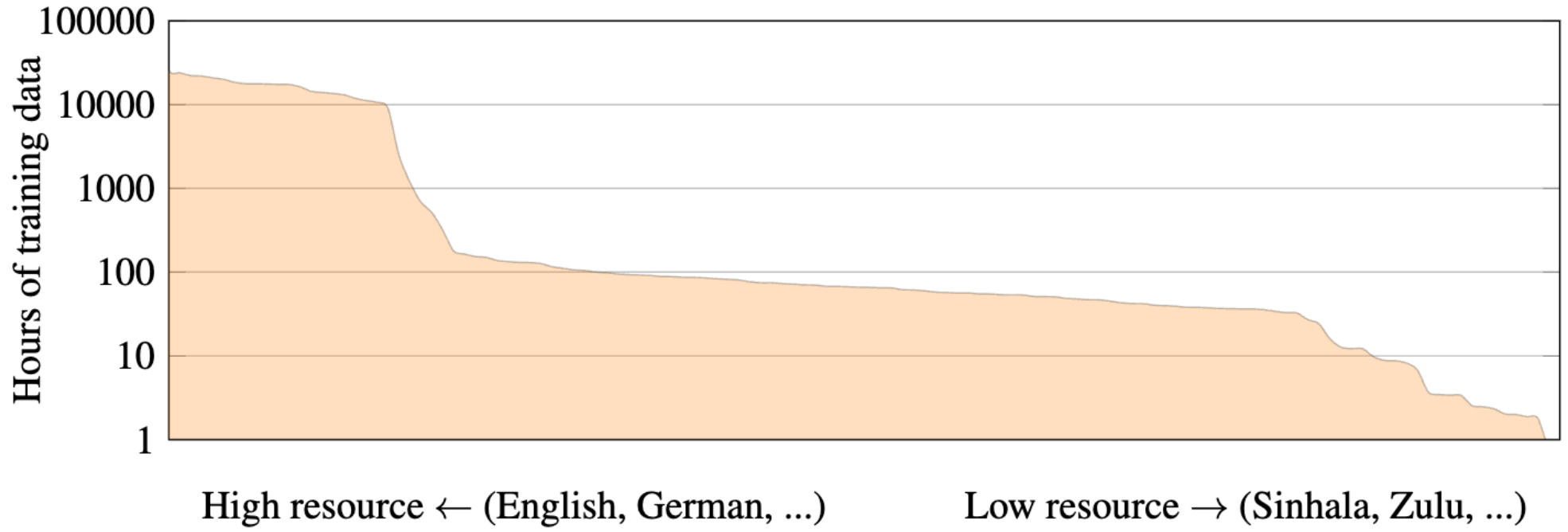


Figure 2: Illustration of the unlabeled training data distribution across the 128 languages of XLS-R.

Language coverage is not the same as effective language representation.

High-resource languages may dominate the gradients, so sampling and balancing are also required

Vineel Pratap, Andros Tjandra, Bowen Shi, et al., "Scaling Speech Technology to 1,000+ Languages," *Journal of Machine Learning Research*, vol. 25, no. 97, pp. 1–52, 2024.

Scaling Language Coverage: MMS

MMS: Can Speech Technology Scale to 1,000+ Languages?

Massively Multilingual Speech (MMS):

SSL pre-training over **1,406 languages**
One multilingual ASR model for **1,107 languages**
TTS support for **1,107 languages**
Language identification for **4,017 languages**

On 54 FLEURS languages,
MMS substantially reduced average WER relative to Whisper while using far less labeled training data.

Scale the language space, not only the parameter count.

Vineel Pratap, Andros Tjandra, Bowen Shi, et al., "Scaling Speech Technology to 1,000+ Languages," *Journal of Machine Learning Research*, vol. 25, no. 97, pp. 1–52, 2024.

Case Study I: Related Languages Can Help

Case Study I — Multilingual Transfer across Turkic Languages

Joint ASR training across related languages produced particularly large gains for critically low-resource languages.

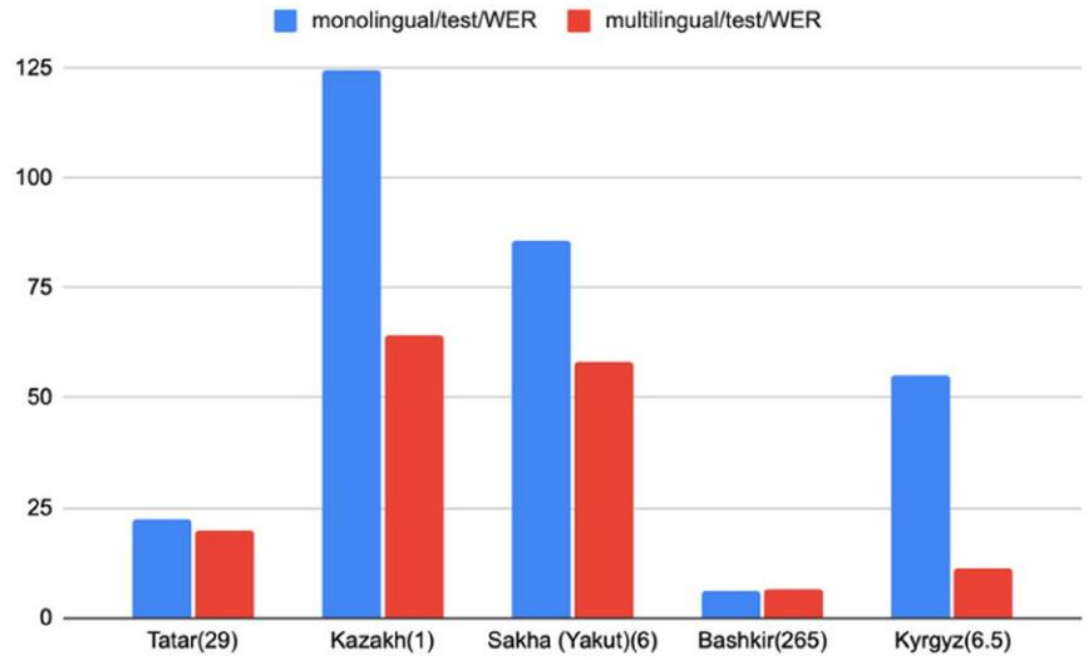


Figure 3. Monolingual ASR model WER comparison with multilingual training.

Related-language Transfer is strong when useful linguistic overlap meets severe data scarcity.

Akbayan Bekarystankyzy, Orken Mamyrbayev, Mateus Mendes, et al., "Multilingual End-to-End ASR for Low-Resource Turkic Languages with Common Alphabets," *Scientific Reports*, vol. 14, Article 13835, 2024.

Case Study II — Use Text Even When It Is Not Paired with Speech

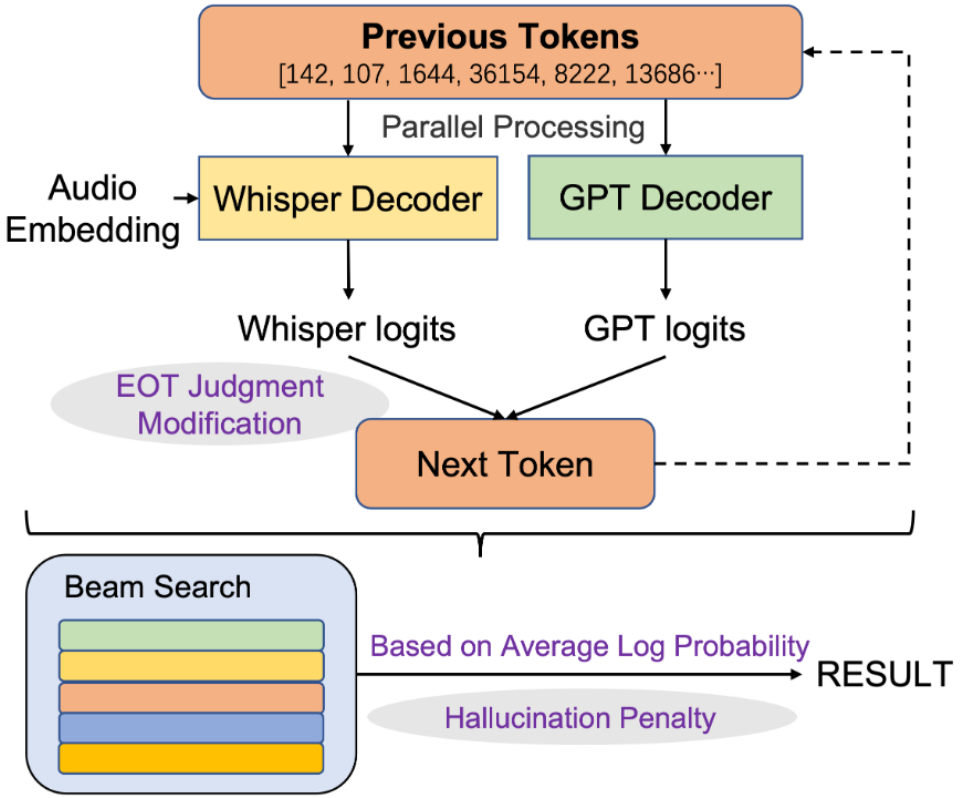


Figure 1: *Integrating GPT into the decoding process of Whisper.*

What if we have:

$$\text{Speech}_A \quad + \quad \text{Text}_B$$

but no alignment?

Li et al. combine:

- Whisper acoustic predictions
- A Kazakh GPT language model
- Modified decoding
- Hallucination penalties

Unpaired text can improve linguistic decoding without requiring new speech transcription.

Jinpeng Li, Yu Pu, Qi Sun, and Wei-Qiang Zhang, "Improving Whisper's Recognition Performance for Under-Represented Language Kazakh Leveraging Unpaired Speech and Text," *Proc. Interspeech*, pp. 2514–2518, 2024.

Case Study III — Pseudo-Labeling Untranscribed Speech

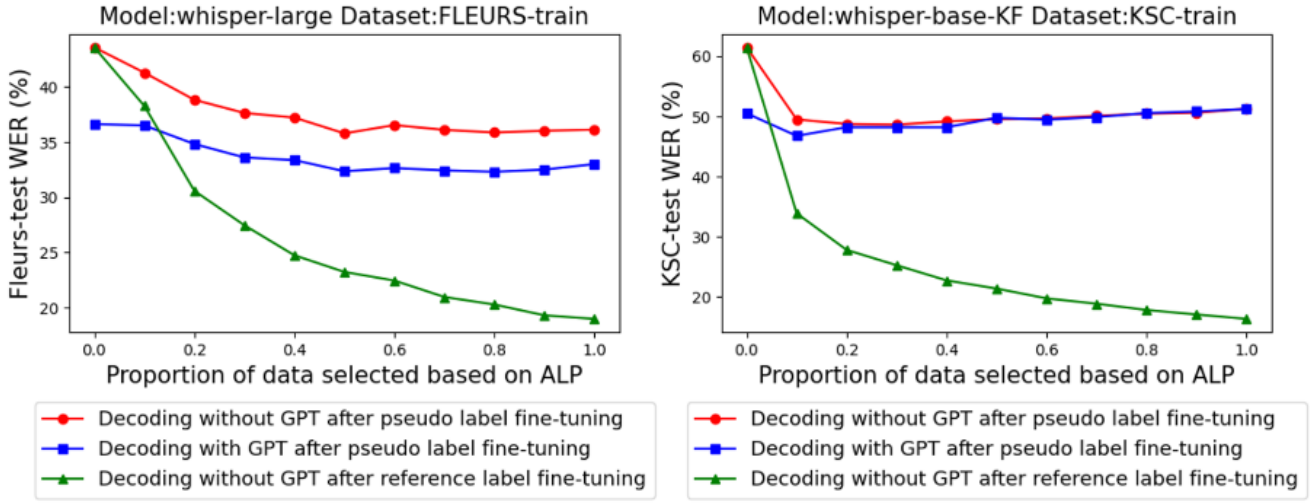


Figure 3: *Relationship between the proportion of data selected based on the average token log probability and the WER of the corresponding domain test set.*

- Use the current ASR model to
- decode unlabeled speech
- Estimate prediction confidence
- Keep reliable pseudo-transcriptions
- Fine-tune on selected pseudo-labeled data

Li et al. use **Average Token Log Probability (ALP)** for data selection.
They report more than **10 % absolute WER reduction** in multiple Kazakh experiments.

Arun Babu, Changhan Wang, Andros Tjandra, et al., "XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale," *Proc. Interspeech*, pp. 2278–2282, 2022.

From Speech Representations to Unified Speech–Text Models

The supervision can now flow across both languages and modalities.

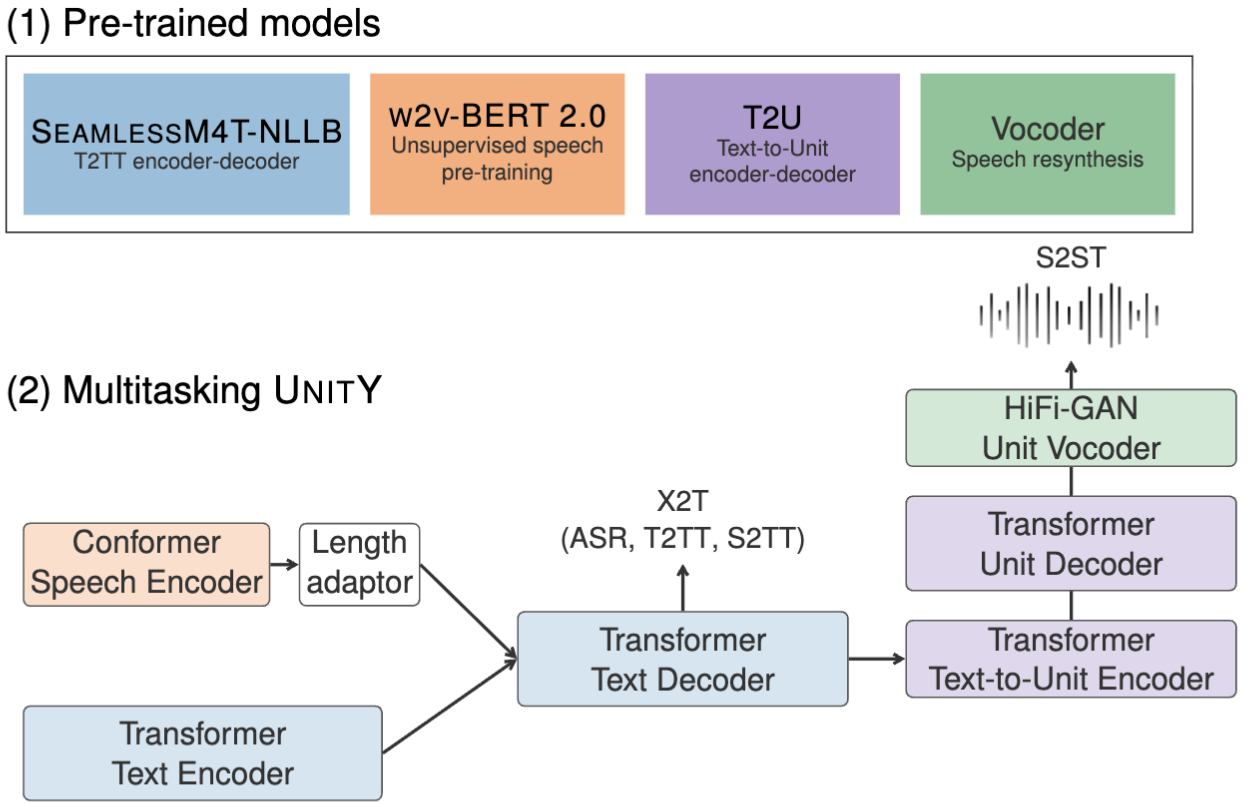


Figure 4: Overview of SeamlessM4T. (1) shows the pre-trained models used when finetuning multitasking UnitY. (2) outlines multitasking UnitY with its two encoders, text decoder, T2U encoder-decoder, and the supporting vocoders for synthesizing output speech in S2ST.

A unified model makes it possible to combine these heterogeneous resources in one training framework.

Seamless Communication, Loïc Barrault, Yu-An Chung, et al., "SeamlessM4T: Massively Multilingual & Multimodal Machine Translation," *arXiv:2308.11596*, 2023.

Shared Embedding Spaces Enable Data Mining

Cross-Modal Transfer Requires Cross-Modal Alignment

SeamlessM4T uses multilingual speech–text representations to mine aligned data from large unstructured corpora.

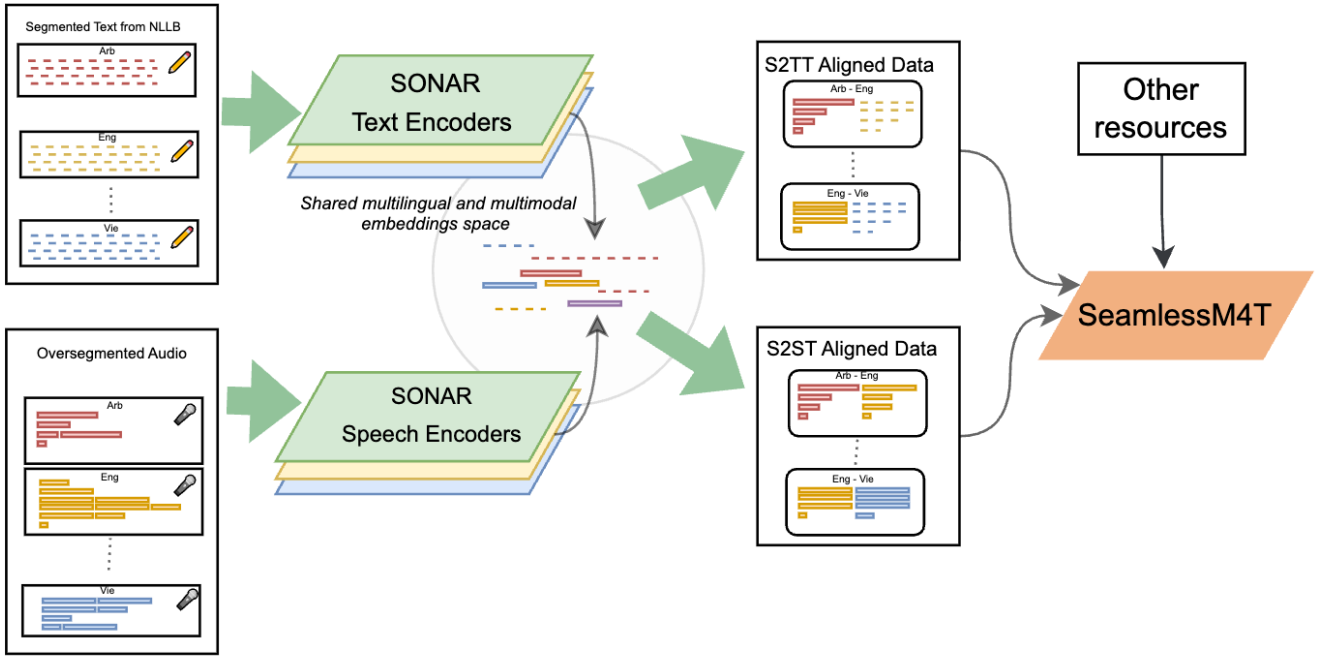


Figure 2: Workflow of the SONAR encoding and mining processes.

Shared embedding converts previously unpaired resources into useful training signals.

Seamless Communication, Loïc Barrault, Yu-An Chung, et al., "SeamlessM4T: Massively Multilingual & Multimodal Machine Translation," *arXiv:2308.11596*, 2023.

Parameter-Efficient Adaptation Under Scarcity

When Data and Compute Are Both Limited

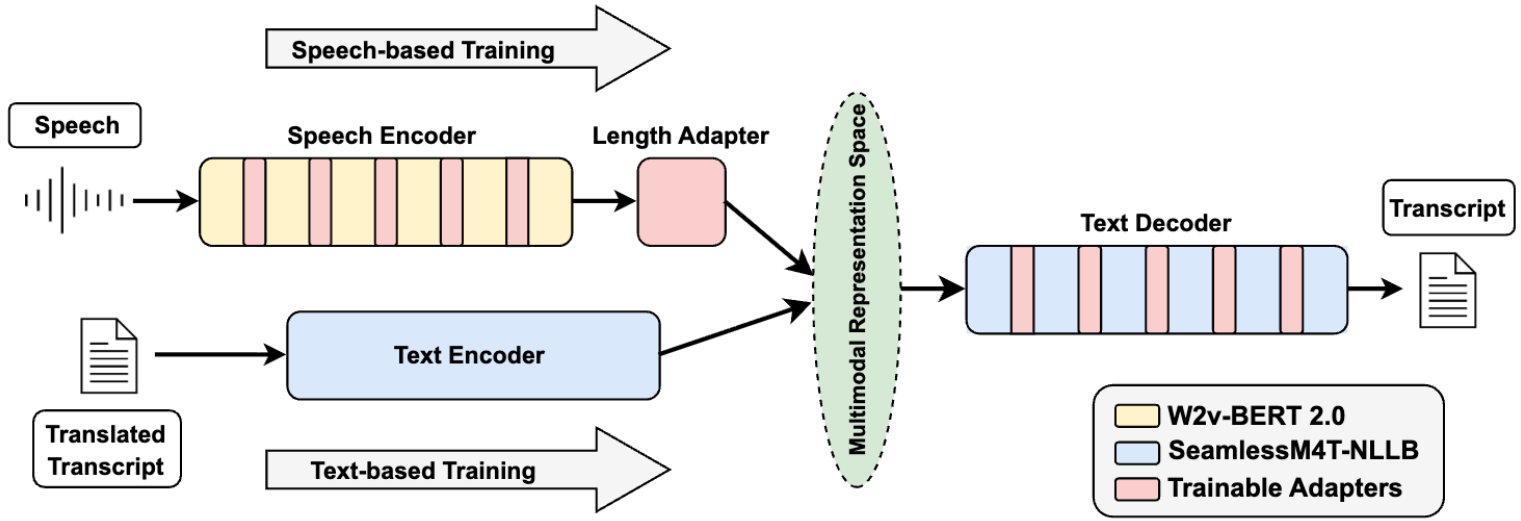


Figure 1: **Parameter-efficient Adaptations for SeamlessM4T:** A multimodal ASR model such as SeamlessM4T can be fine-tuned in a parameter-efficient manner through either speech-based adaptations or text-only adaptation.

Gupta et al. combine **text-only adaptation + PEFT + multilingual transfer** using SeamlessM4T. Cross-lingual transfer achieved up to **17% relative WER reduction** in an extremely low-resource setting without target-language labeled speech.

Abhishek Gupta, Amruta Parulekar, Sameep Chattopadhyay, and Preethi Jyothi, "Parameter-Efficient Adaptation of Multilingual Multimodal Models for Low-Resource ASR," *Proc. MRL*, pp. 175–185, 2024.

PEFT Should Also Be Resource-Aware

Not Every Layer Needs the Same Amount of Adaptation

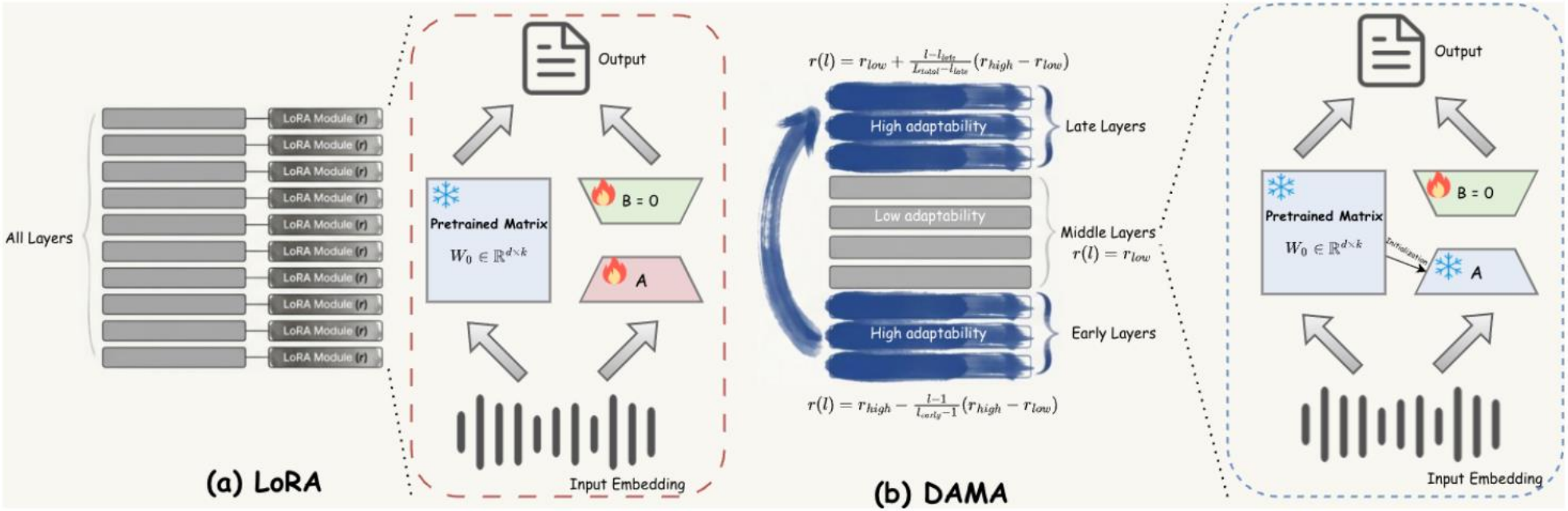


Figure 2: Overview of the DAMA Framework compared with LoRA. (a) The standard LoRA with uniform rank. (b) The DAMA Framework. Specifically, the Depth-Aware Rank schedule allocates high plasticity to the early and late layers, while the Basis-Protected Projection physically locks the middle layers to protect "Semantic Valley". All layers mean all the layers from the decoder. The decoder processes acoustic embeddings from the encoder and transcribes them into output tokens.

DAMA allocates adaptation capacity according to layer depth.

Yang Xiao, Eun-Jung Holden, and Ting Dang, "Adapting Where It Matters: Depth-Aware Adaptation for Efficient Multilingual Speech Recognition in Low-Resource Languages," *Findings of ACL*, pp. 16776–16788, 2026.

More Fine-Tuning Data Is Not Always Better

How Much Target-Language Data Is Enough?

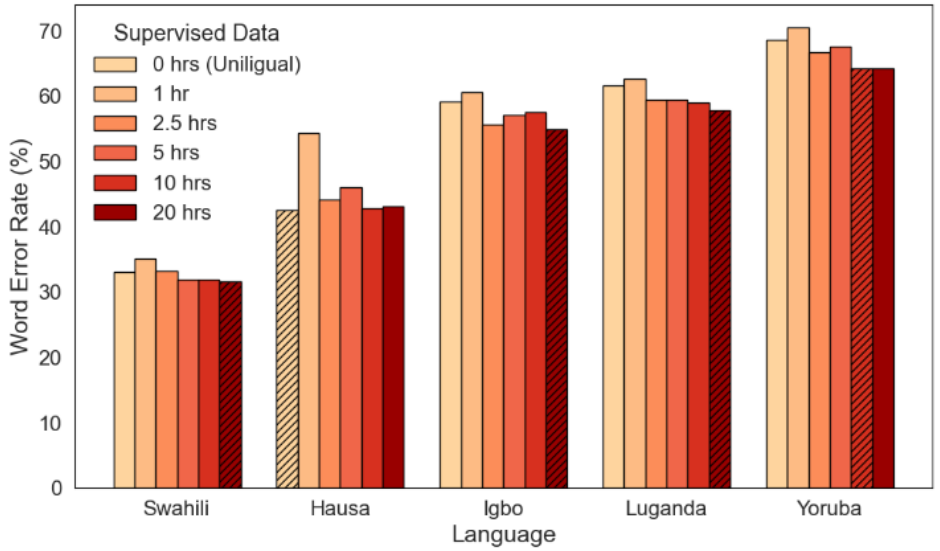


Figure 1: Sample efficiency measured by ASR **WER** (%) scores (↓) with varying amounts of finetuning hours; dashed bars indicate the best system for each language. Please refer to **Section 4.4** for details.

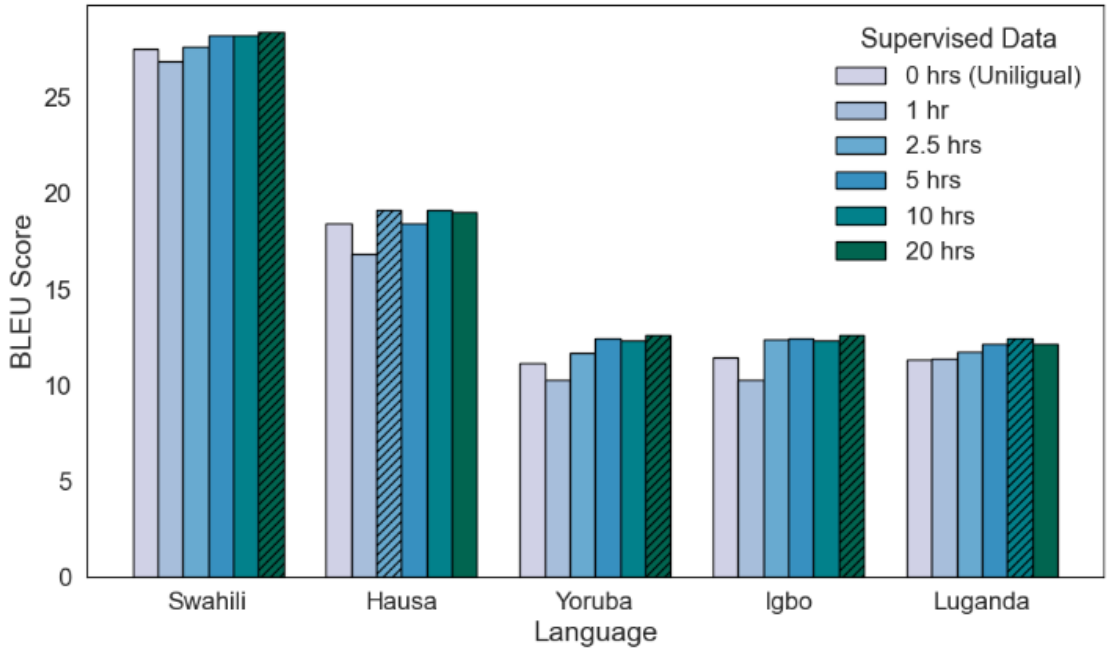


Figure 2: Sample efficiency measured by MT **BLEU** scores (↑) with varying amounts of finetuning hours; dashed bars indicate the best system for each language. Please refer to **Section 4.4** for details.

Data quality and domain match can matter as much as data quantity.

Jonah Dauvet, Min Ma, Jessica Ojo, and David Ifeoluwa Adelani, "Reassessing Speech Translation for Low-Resource Languages: Do LLMs Redefine the State-of-the-Art Against Cascaded Models?," *Proc. MRL*, pp. 149–160, 2025. ACL Anthology

Can Speech LLMs Learn an Unseen Language

A New Question: Can We Adapt Without Updating Parameters?
Multimodal in-context learning (MICL):

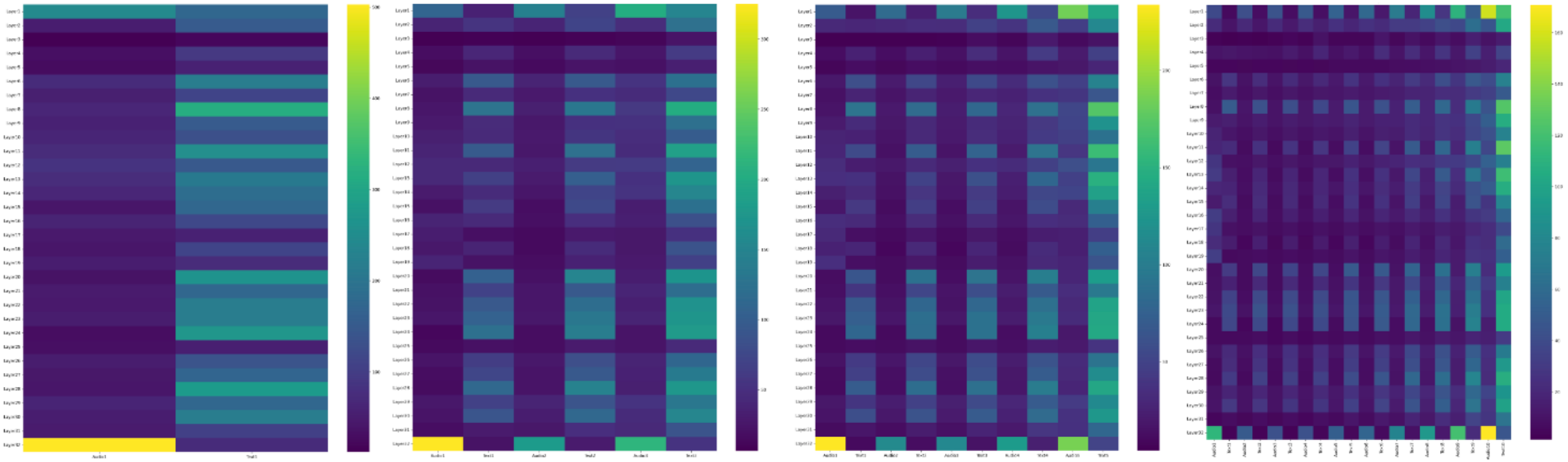


Figure 1: Layer-wise attention allocation across demonstrations of 5 and 10 samples using Phi4, averaged over all attention heads on the Khinalug dataset.

- Speech LLMs can exploit paired speech–text demonstrations
- A stronger acoustic model + Speech LLM hypothesis selection works better

Zhaolin Li and Jan Niehues, "Multimodal In-Context Learning for ASR of Low-Resource Languages," *Findings of ACL*, pp. 24745–24760, 2026.

Match the Method to the Missing Resource

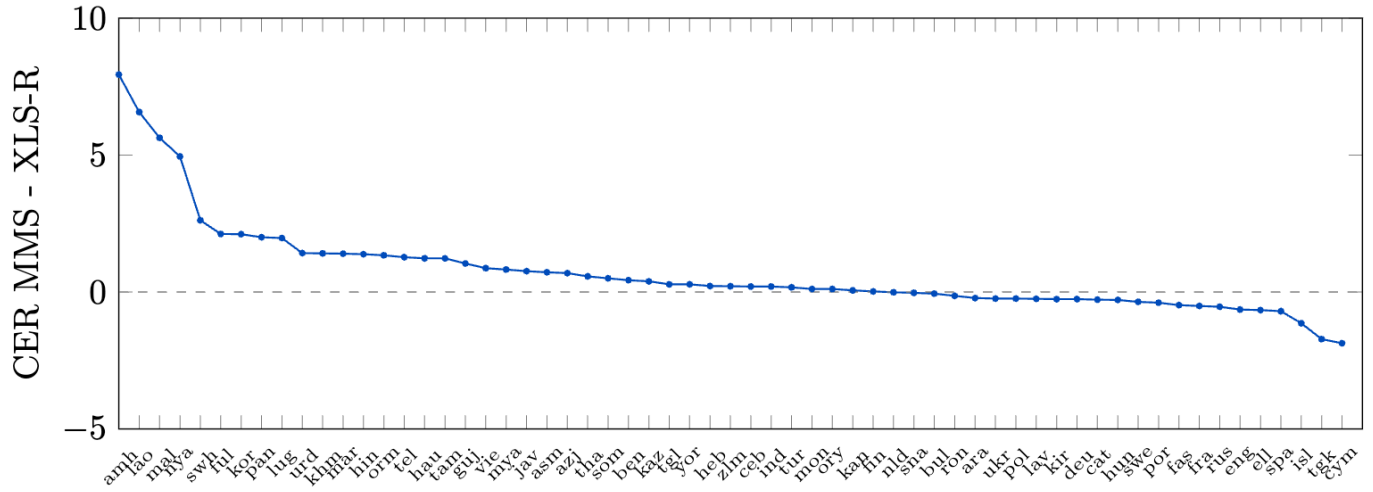


Figure 11: **MMS vs. XLS-R Breakdown.** We show the absolute character error rate difference between multilingual ASR models based on XLS-R and MMS models with 1B parameters. Positive values indicate better performance of MMS and negative values better performance of XLS-R. Models are fine-tuned on MMS-lab data and evaluated on the development sets of 61 FLEURS languages.

- Low-resource is multidimensional
- Speech foundation models change where **supervision** can come from
- More languages and larger models do not automatically guarantee better low-resource performance

Vineel Pratap, Andros Tjandra, Bowen Shi, et al., "Scaling Speech Technology to 1,000+ Languages," *Journal of Machine Learning Research*, vol. 25, no. 97, pp. 1–52, 2024.

Thanks for Your Attention