Moshi:
a speech-text foundation model for real-time dialogue

Authors: Alexandre Defossez, Laurent Mazare, Manu Orsini, Amelie Royer, Patrick Perez, Herve Jegou, Edouard Grave, Neil Zeghidour, **Kyutai**

Presented by: Shrey Pandit, Gautham Krishna Gudur, Ashwin Vinod

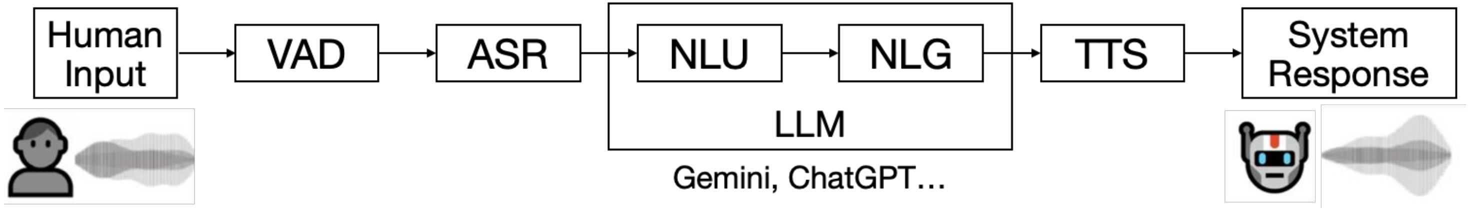
Introduction

Motivation

Current Voice AIs:

- Siri, Google Assistant, Alexa
- How do you feel about them?

Modular **pipeline**:



- Voice Activity Detection (VAD): Detects when speech starts and ends
- Automatic Speech Recognition (ASR): Converts speech into text
- Natural Language Understanding (NLU): Understands the user's intent
- Natural Language Generation (NLG): Produces a text response
- Text-to-Speech (TTS): Converts the text response back into speech

The Three Key Problems with Traditional Systems

High Latency

- latency compounds along the components of pipelines
- total response times of several **seconds**
- human-to-human conversations: a few hundred **milliseconds**

Information Bottleneck

- Work primarily in the text domain
- Ignore **paralinguistic** information like **emotions**, **accents**, and even **background sounds** that are crucial in natural conversations

Turn-Based Limitations

- Fundamentally **turn-based**: assuming that dialogue is a sequence
- Fail to handle **interruptions**, **overlapping speech** (amounts for 10 to 20% of spoken time), or non-interrupting **interjections** (real-time feedback like “uh-huh”, “I see” or “okay”)

Moshi's Breakthroughs

Removed Information Bottleneck

- Understands inputs and generates outputs directly in the audio domain (compress audio into "**pseudo-words**", predict next audio segment from previous audio)
- Capture both linguistic and non-linguistic cues like tone, emotion, and even silence.

Multi-stream: Full-Duplex Real-Time Interaction

- Listens and speaks simultaneously
- Allowing for arbitrary **conversational dynamics** including overlap and interruptions

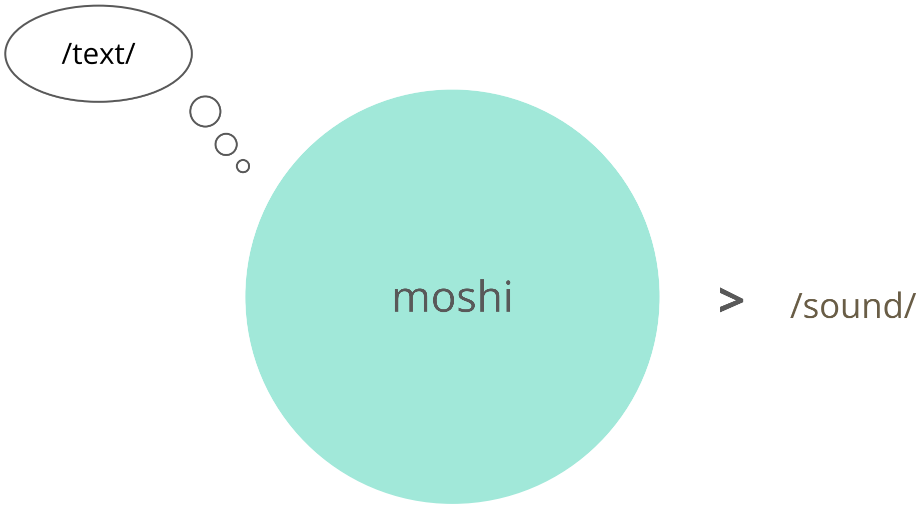
Reduced Latency

- A theoretical latency of **160ms** and a practical latency of **200ms**
- Average latency in natural conversations: **230ms**

Moshi's Breakthroughs

Multi-modality

- Moshi benefits from thinking as it speaks
- Textual thoughts help train the model

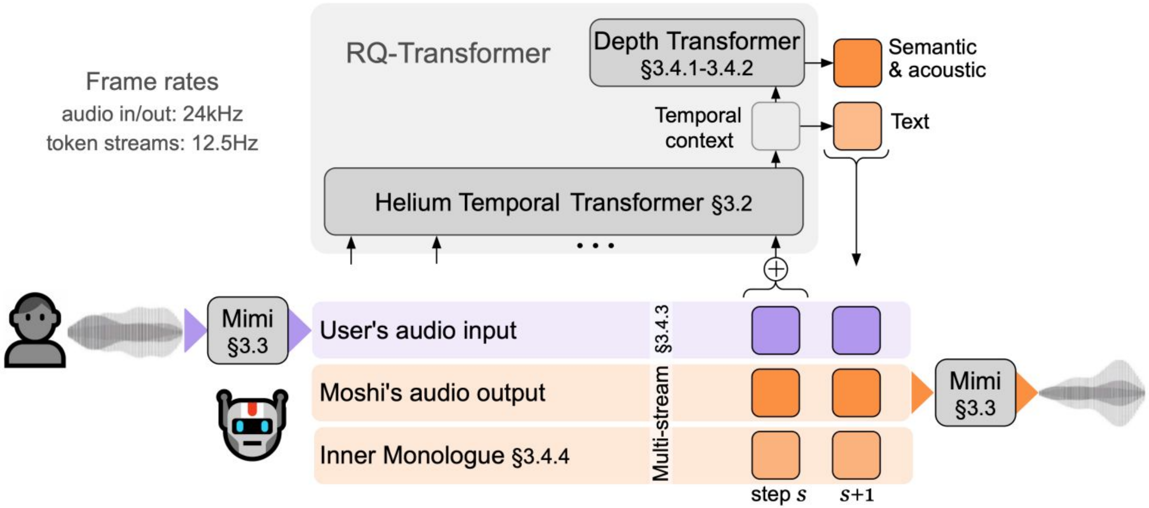


Video Demo

- Live Moshi demo: https://moshi-ai.com/
- Youtube Demo Video: https://www.youtube.com/watch?v=hm2IJSKcYvo

Model

Architecture Overview



Helium: A High-Performance Text Language Model

Mimi: A Neural Audio Codec with Residual Vector Quantization (RVQ)

Hierarchical and Streaming Audio Modeling

Inner Monologue: Enhancing Quality through Text-Audio Alignment

Helium 7B LLM

Helium LLM

What is Helium?

7B-parameter text language model

Foundation model for Moshi dialogue system

Pretrained on 2.1T tokens of public **English** data

Built from scratch

Architecture of Helium

Based on Transformer architecture

RMS normalization at:

- Input of attention blocks
- Feed-forward blocks
- Output linear layer

Rotation positional embeddings (RoPE)
Context length of 4,096 tokens

Flash Attention for efficient training

Gated Linear Units with SiLU activation

My question - Why not the latest flash attention 2?

Some Details

1. Helium is a part of Moshi, but is also trained separately on text only data first
2. This pre-trained Helium is then used as initialization for the temporal transformer

---

3. **Pre-training of Moshi**, on unsupervised data
4. **Post-training** with simulated multi-stream based on diarization
5. Fine-tuning on the **Fisher dataset** to gain its fully duplex capabilities
6. Final fine tuning on custom internal dataset

Hyper-parameter	Helium training pre-training	Moshi training pre-training	Moshi training post-training	Moshi training fisher	Moshi training fine
Temporal Transformer					
Model dimension	4096	same			
MLP dimension	11264	same			
Number of heads	32	same			
Number of layers	32	same			
Context size	4096	3000 steps, e.g. 4 min.			
Learning rate	$3 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$3 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	$2 \cdot 10^{-6}$
Depth Transformer					
Model dimension	-	1024			
MLP dimension	-	4096			
Number of heads	-	16			
Number of layers	-	6			
Learning rate	-	$2 \cdot 10^{-4}$	$5 \cdot 10^{-5}$	$4 \cdot 10^{-6}$	$2 \cdot 10^{-6}$
Input / Output space					
Text cardinality	32000	32000			
Audio cardinality	-	2048			
Frame rate	-	12.5 Hz			
Common parameters					
Batch size (text)	4.2M tok.	1.2M tok.	1.2M tok.	-	-
Batch size (audio)	-	16h	8h	40min	2.7h
Training steps	500k	1M	100k	10k	30k
LR Schedule	cosine	cosine	-	-	-
Acoustic delay	-	2	1	1	1
Text delay	-	± 0.6	0	0	0

Some Observations

1. Helium pre-training was done on only Text data and no audio

2. Moshi audio batch size was kept pretty high at 16h

3. Acoustic delay of 1 or 2 step between semantic and acoustic tokens to improve quality (more in further section)

My question - Why keep 4k context size? Why not the current standard of 128k?

Hyper-parameter	Helium training pre-training	Moshi training pre-training	Moshi training post-training	Moshi training fisher	Moshi training fine
Temporal Transformer					
Model dimension	4096	same			
MLP dimension	11264	same			
Number of heads	32	same			
Number of layers	32	same			
Context size	4096	3000 steps, e.g. 4 min.			
Learning rate	$3 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$3 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	$2 \cdot 10^{-6}$
Depth Transformer					
Model dimension	-	1024			
MLP dimension	-	4096			
Number of heads	-	16			
Number of layers	-	6			
Learning rate	-	$2 \cdot 10^{-4}$	$5 \cdot 10^{-5}$	$4 \cdot 10^{-6}$	$2 \cdot 10^{-6}$
Input / Output space					
Text cardinality	32000	32000			
Audio cardinality	-	2048			
Frame rate	-	12.5 Hz			
Common parameters					
Batch size (text)	4.2M tok.	1.2M tok.	1.2M tok.	-	-
Batch size (audio)	-	16h	8h	40min	2.7h
Training steps	500k	1M	100k	10k	30k
LR Schedule	cosine	cosine	-	-	-
Acoustic delay	-	2	1	1	1
Text delay	-	± 0.6	0	0	0

Data used to train Moshi

Text Data (Website scrapings)

- 12.5% of the dataset is from the following curated sources: Wikipedia,9 Wikibooks, Wikisource, Wikinews, StackExchange
- The remaining 87.5% of the dataset is from Common Crawl

Filtration

- Deduplication: removing of duplicate data content
- Language Identification: Use fasttext to keep only english language content.
- Quality filtering: Use fasttext classifier

Audio data

Unsupervised Dataset

- 7M hours of English speech
- Single-stream audio (24kHz, mono)
- Transcribed with Whisper large-v3

Fisher Dataset

- 2K hours of phone conversations
- Separate channels for speakers
- Upsampled from 8kHz to 24kHz using AudioSR

Supervised Multi-stream Dataset

- 170 hours of natural/scripted conversations
- Used for TTS model training
- Not used directly for Moshi training

Helium performance on Text datasets

The helium text models is competitive and performs well on standard benchmark tasks

Performs well on QA tasks when compared to models of larger size (Mistral and Gemma)

	ARCe	ARCc	OBQA	HS	WG	PIQA	SIQA	TQA	NQ	MMLU
Helium	**79.6**	**55.9**	53.6	76.3	**70.0**	79.4	**51.0**	**59.9/72.6**	23.3	**54.3**
MPT	70.5	46.5	51.4	**77.6**	69.9	**80.6**	48.5	-/61.2	20.8	30.8
Falcon	73.7	47.5	53.0	76.3	68.9	80.3	47.2	-/64.6	21.0	28.0
Llama 2	75.2	45.9	**58.6**	77.2	69.2	78.8	48.3	-/72.1	**25.7**	45.3
OLMo	67.2	42.5	50.0	75.5	69.8	77.5	-	-/-	-	52.0
Mistral	80.5	54.9	52.2	81.0	74.2	82.2	47.0*	62.5/-	23.2	62.5
Gemma 1	81.5	53.2	52.8	81.2	72.3	81.2	51.8	63.4/-	23.0	64.3

Why not compare it to Gemma 2 9B?

Moshi Pre-training

– Temporal Transformer in Moshi with Helium, while the Depth Transformer is randomly initialized

– We first train on the unsupervised audio dataset, using a single stream of audio (this is the first time audio data is introduced)

– In order to prevent catastrophic forgetting, also train half of the time on batches of text only data from the same dataset as used for Helium

Moshi Post training

Main aim of post training – **To gain its multi-stream ability**

Speaker Diarization using PyAnnote

- Creates binary mask (1: active, 0: inactive)
- Separates into two streams:
  - Main speaker waveform
  - Residual speakers waveform

Training Parameters:

- Text stream aligned with main speaker
- No delay between text/audio tokens

```
Original Audio:
[Speaker A]: "Hello how are you?"
[Speaker B]: "I'm good, thanks"
[Speaker A]: "Great to hear"

If Speaker A is chosen as main speaker:
Mask:        1 1 1 1 0 0 0 0 1 1 1
Main Wave:   "Hello how are you?" ... "Great to hear"
Residual:    [silence] "I'm good, thanks" [silence]
```

Moshi Fine tuning

Multi-stream Training (Fisher Dataset) 2000 hours of Overlapping Phone conversation between users

Problem: Simulated data lacks natural **overlaps**

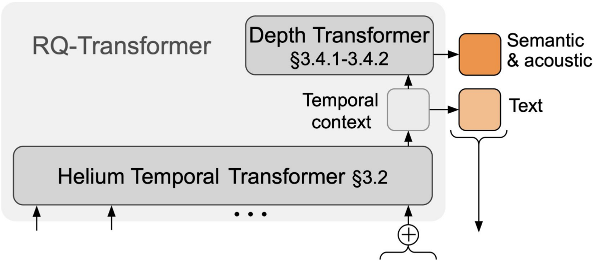
- Solution: Real conversation data with overlapping speech
- Implementation:
  - 10K batches of 40 min conversations
  - Learns from genuine two-speaker interactions
  - Separate tuning for both transformers

Robust Audio Processing (Creates synthetic data for this)

Challenge: Real-world audio is messy

- Dynamic Gain (-24dB to +15dB)
  - Handles varying user volumes
  - Applies to 50% of training data
- Environmental Noise
  - Uses Deep Noise Suppression challenge data
  - Varies noise levels (-30dB to +6dB)
  - Includes strategic silence periods
- Echo & Reverb Simulation
  - Mimics real-room acoustics
  - Controlled echo delay (100-500ms)
  - Combined effects in 30% of cases

Moshi Training Loss



Gives the same importance to the text token (k=1), and the combined audio tokens.

Alpha_k is set to 100 for semantic tokens, and 1 for acoustic ones.

$$L(V, l) = \frac{1}{S} \sum_{s=1}^{S} \left(\mathrm{CE}(l_{s,1}, V_{s,1}) + \frac{1}{\sum_{k=2}^{K} \alpha_k} \sum_{k=2}^{K} \alpha_k \mathrm{CE}(l_{s,k}, V_{s,k}) \right)$$

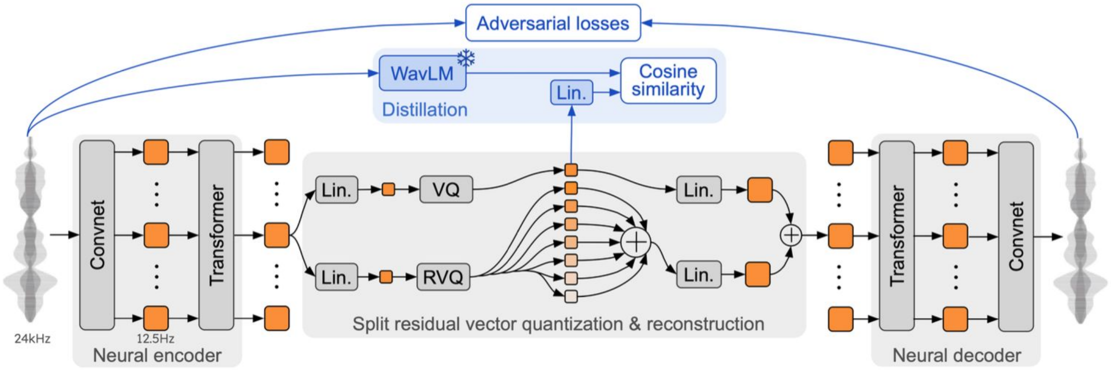
Cross entropy on the text token

Weighting on different tokens

l_(s,k) is estimated logits from the depth transformer, V_(s,k) is GT discrete token

Mimi: A hybrid audio tokenizer

Mimi: A hybrid audio tokenizer

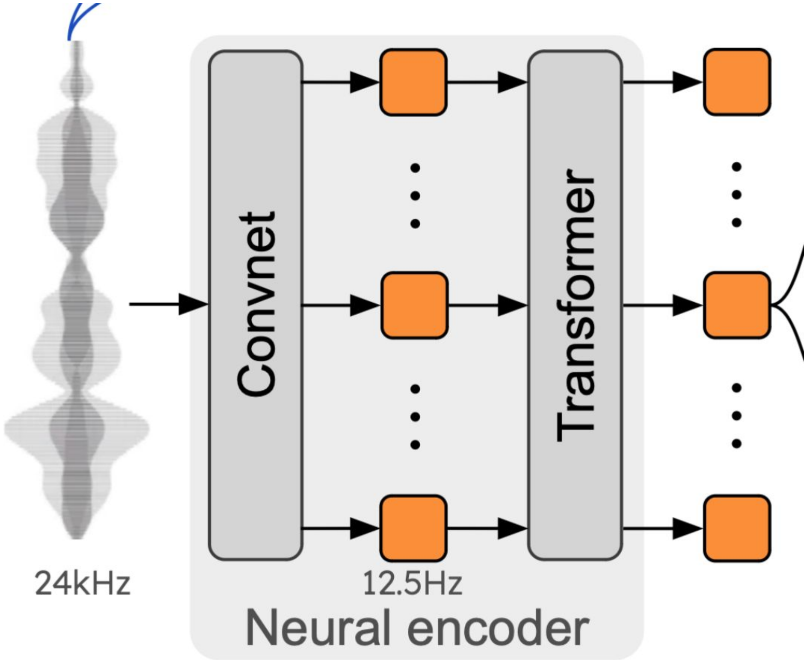


Feature	Acoustic Tokens	Semantic Tokens
Audio Reconstruction Quality Ability to reconstruct high-quality audio from tokens	✓	✗
Linguistic Correlation Strength of correlation with linguistic content	✗	✓

- Acoustic tokens model fine audio details and are optimized for high-quality reconstruction.
  - Conditioned text-to-audio models
    - Text-to- speech
    - Text-to-music
- Unconditioned speech generation requires combining them with semantic tokens extracted from self-supervised speech models.

- Semantic tokens are not causal.
  - computed in an offline manner.
- Generating acoustic and semantic tokens with separate encoders is a computational burden.
- Uses distillation to transfer non-causal, high-level semantic information into the tokens produced by a causal model.
  - Allowing for streaming encoding and decoding of semantic-acoustic tokens.

Encoder Block



Input: Single-channel waveform (24Khz: $x \in \mathbb{R}^L$

Output: Latent representation: $\mathrm{enc}(x) \in \mathbb{R}^{S \times D}$

L samples of audio, S frames, each with D dimensions

Inspiration:

- SoundStream (Zeghidour et al., 2022)
- Encodec (Défossez et al., 2023)

Components:

- SeaNet (Tagliasacchi et al., 2020) autoencoder
- Residual Vector Quantizer (RVQ) (Zeghidour et al., 2022)

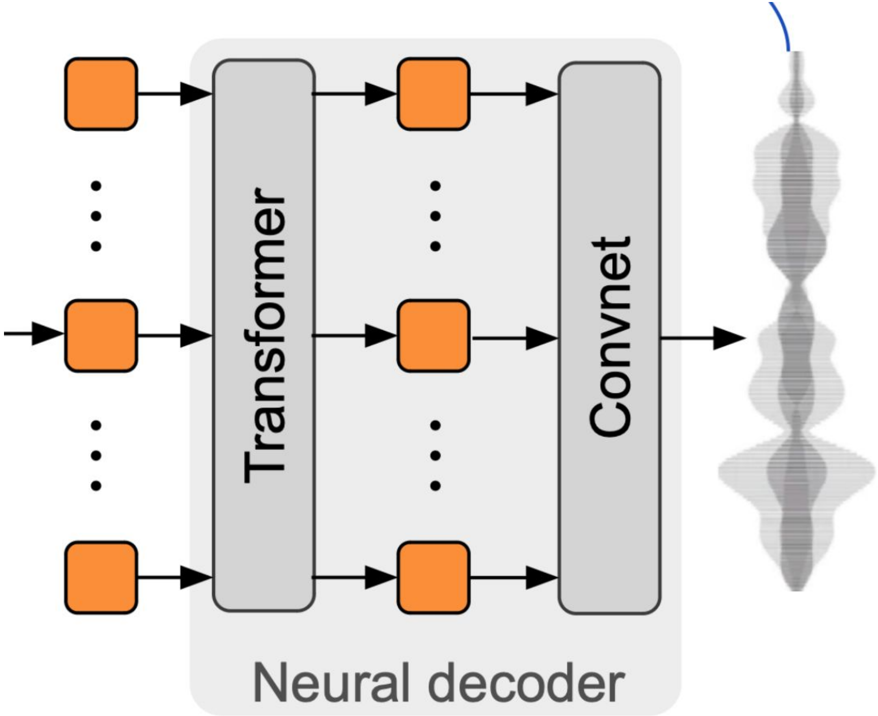
Residual convolutional blocks with:

- **Combined Dilated** and **strided** convolutions :
  - **Dilated Convolutions**: Increase the receptive field..
  - **Strided Convolutions**: Downsample the input, reducing the temporal dimension.
- **Activation Function: ELU** non-linearities (Clevert et al., 2016)
- **Weight Normalization** (Salimans and Kingma, 20.

Causal convolutions for streaming capability

- 4 blocks with strides: (4,5,6,8) followed by 1D convolution with stride 2.
- **Output**: 24 kHz waveform to latent representation of 12.5 FPS and dimension 512.

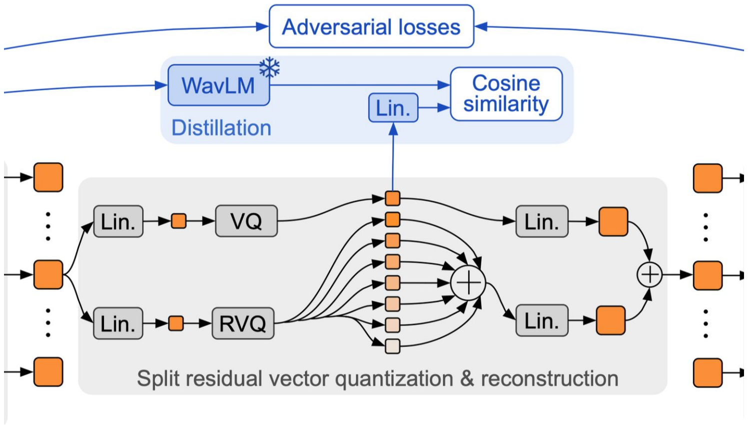
Decoder Block



Mirrors the encoder but uses **transposed convolutions** to upsample the latent representation back to the original audio sampling rate.

Input: Quantized latent representation.
Output: Reconstructed waveform.

Split RVQ



Split RVQ:

- Replace single RVQ (8 levels) with:
    - **Plain VQ**: Dedicated to semantic information.
    - **RVQ (7 levels)**: Operates in parallel on acoustic residuals.
- Outputs from both summed for reconstruction.
- Decouples semantic and acoustic information, improving trade-off.

Initial Setup of distilling semantic information.

- Inspired by **SpeechTokenizer** (Zhang et al., 2024b).
- Distills semantic info from **WavLM** (Chen et al., 2022) into the first RVQ level.
- Mimi: Projects 24 kHz waveform to 512-d embeddings at 12.5 Hz
- WavLM: Projects 16 kHz waveform to 1024-d embeddings at 50 Hz.
- Targets for distillation
    - Downsample input to 16 kHz for WavLM embeddings.
    - Apply **non-causal average pooling** (stride = 4, kernel = 8) to align embeddings at 12.5 Hz.
- Compute cosine distance between the first quantizer output and transformed WavLM embeddings.
- Linear projection (output dim: 1024) applied parallel to embeddings going into the decoder.

Trade-offs Observed:

- **Benefit**: Improves **phonetic discriminability** (measured by **ABX** scores).
- **Drawback**: Negatively impacts audio quality due to conflicts with reconstruction and adversarial losses

Improving Encoding ability of **Mimi**

Feature	Details
Transformer Bottleneck	• Enhances compactness of encoded speech and reconstructs high-quality audio. • Placed One Transformer before and one after quantization. • 8 layers, 8 heads, RoPE position encodings, finite context (250 frames = 20 sec). • Model dimension = 512, MLP dimension = 2048. • **Stabilization**: LayerScale with diagonal initialization at 0.01. • **Causality**: Uses causal masking for streaming compatibility. • **Benefits**: Improves perceived audio quality and semantic information distillation.
Optimization	• **Optimizer**: AdamW with weight decay (0.05) for Transformer parameters. • **Hyperparameters**: Learning rate = $8 \cdot 10^{-4}$, momentum decay = 0.5, squared gradient decay = 0.9. • **Batch Size**: 128 (random 12s windows). • **Steps**: 4M training steps with Transformer context limited to 10s.

Quantization Rate	• **Quantizers**: $Q = 8$, Codebook size $N_A = 2048$. • **Bitrate**: 1.1 kbps at 12.5 Hz. • **Embedding Projection**: $512 \to 256$ (before RVQ), $256 \to 512$ (after RVQ). • **Quantizer Dropout**: 50% probability during training for improved bitrate scalability. • **Observation**: Dropout benefits increase with lower bitrates.
Adversarial Training	• **Baseline Losses**: Reconstruction + adversarial (multi-scale mel-spectrogram & STFT). • **Pure Adversarial**: Feature loss + discriminator loss only. • **Results**: Objective metrics degraded but subjective evaluations showed superior quality. • **References**: Table 3 (quantization dropout effects) and Table 4 (adversarial-only training).

Results

Quantization Rate	Transformer in encoder	Transformer in decoder	WavLM distillation	Split quantizer	ABX (↓)	VisQOL (↑)	MOSNet (↑)	MUSHRA (↑)
✓	✓	✓			23.3%	2.91	2.89	65.9±1.7
✓	✓	✓	✓		6.5%	2.22	2.87	57.8±1.8
✓		✓	✓	✓	10.8%	2.79	2.85	59.7±1.7
✓	✓		✓	✓	8.1%	2.59	2.72	48.4±1.7
	✓	✓	✓	✓	8.0%	2.45	2.88	68.3±1.7
✓	✓	✓	✓	✓	8.1%	2.82	2.89	64.0±1.7

Model	f_s	f_r	bitrate	causal	ABX (↓)	VisQOL (↑)	MOSNet (↑)	MUSHRA (↑)
Ground Truth	24kHz	-	-	-	-	-	3.08	90.6±1.0
RVQGAN	24kHz	75Hz	1.5kbps		-	1.74	2.74	31.3±1.3
SemantiCodec	16kHz	50Hz	1.3kbps		42.2%	2.43	3.12	64.8±1.5
SpeechTokenizer	16kHz	50Hz	1.5kbps		**3.3%**	1.53	2.67	45.1±1.5
SpeechTokenizer	16kHz	50Hz	4.0kbps		**3.3%**	3.07	3.10	74.3±1.5
Mimi, adv. loss only	24kHz	12.5Hz	1.1kbps	✓	8.7%	1.84	3.10	**81.0**±1.3
Same, downsampled at 16kHz	16kHz	12.5Hz	1.1kbps	✓	-	-	-	77.7±1.4
Mimi, non adv. only	24kHz	12.5Hz	1.1kbps	✓	8.1%	2.82	2.89	58.8±1.8

ABX (↓): Error rate on phonetic discriminability (lower is better), VisQOL (↑): Audio quality metric (higher is better) , MOSNet (↑): Another audio quality metric (higher is better), MUSHRA (↑): Human judgment scores (higher is better)

- Adding a Transformer to the decoder significantly improves **MUSHRA scores**.
- Using a 50% quantization rate improves **VisQOL** scores.
- **VisQOL and MOSNet** show poor correlation with perceived audio quality.
- Adversarial-only training achieves a **MUSHRA score of 81.0**, compared to **58.8** with mixed loss functions (Encodec).

Comparison with Baselines:

- Outperforms **RVQGAN** (Kumar et al., 2023):
  - Achieves higher perceived quality despite lower bitrate and semantic modeling.
- Surpasses **SemantiCodec** (Liu et al., 2024):
  - Offers higher reconstruction quality while operating at **4× lower framerate**.

Key Advantage:

- Mimi balances **semantic modeling** and **acoustic reconstruction**, crucial for high-quality audio generation at low bitrates

Generative Audio Modeling – Hierarchical Autoregressive Modeling

- Interested in modeling multiple sub-sequences, e.g. different audio codebooks, along with an optional text stream. Stack those sub-sequences at step s and k^{th} subsequence as $V_{s,k}$ for $1 \leq s \leq S$ and $1 \leq k \leq K$.

- **RQ-Transformer –**
  – For a given sequence step s, Temporal Transformer (Tr_{Temp}) maps to a temporal context vector (z_s)
  – For a given sub-sequence index k, Depth Transformer (Tr_{Depth}) maps z_s and V_s to logits $l_{s,k}$
  – Further, define $l_{s,1} = \mathrm{Lin}(z_s) \in \mathbb{R}^{N_1}$ with a dedicated linear layer
  – Train Tr_{Temp}, Tr_{Depth}, Lin, so that softmax is a good approximation of $V_{s,k}$ conditioned on prev sub-sequences

$$\begin{cases} \mathrm{softmax}(l_{s,1}) & \approx \mathbb{P}\left[V_{s,1} | V_0, \ldots, V_{s-1}\right] \\ \mathrm{softmax}(l_{s,k}) & \approx \mathbb{P}\left[V_{s,k} | V_0, \ldots, V_{s-1}, V_{s,1}, \ldots V_{s,k-1}\right] \quad \text{if } k > 1. \end{cases}$$

- # steps – Temporal Transformer is always S, rather than K.S – Depth Transformer is always K

- Depth Transformer has 6 layers, 1024 dimensions, 16 attention heads

Generative Audio Modeling – Hierarchical Autoregressive Modeling

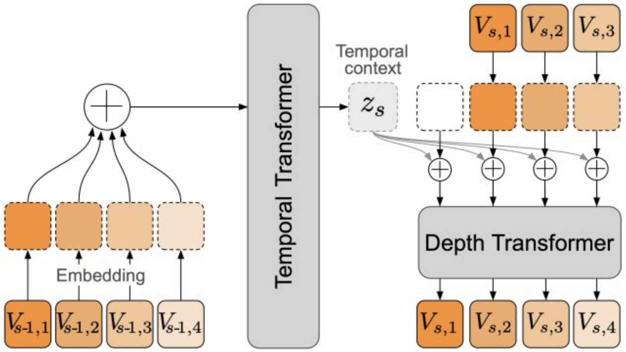


Figure 3: **Architecture of the RQ-Transformer.** The RQ-Transformer breaks down a flattened sequence of length $K \cdot S$ into S timesteps for a large Temporal Transformer which produces a context embedding used to condition a smaller Depth Transformer over K steps. This allows scaling to longer sequences by increasing S—or to a higher depth by increasing K— than modeling the flattened sequence with a single model. In this figure, we use $K = 4$ for the sake of illustration.

Generative Audio Modeling

- **Acoustic delay** – Introduce slight delay (1 or 2 steps) between semantic and acoustic features, allowing the Temporal Transformer to improve quality to model inter-dependence

- Insert $A_{t,q}$ (Q = 8, T = 12.5) sub-sequences from audio codec Mimi into multi-sequence V modeled by RQ-Transformer

  Formally, given a delay $\tau \in \mathbb{N}$, we have, for all steps s

$$\begin{cases} V_{s,1} = A_{s,1} & \\ V_{s,q} = A_{s-\tau,q} & \text{if} \quad s \geq \tau + 1, q > 1 \\ V_{s,q} = 0 & \text{if} \quad s < \tau + 1, q > 1. \end{cases}$$

- Multi-stream Modeling – Can be extended to modeling a two-speaker conversation – concat two streams of audio $A_{t,q}$ (Moshi) and $A'_{t,q}$ (user) with acoustic delay into V

Ablations on Generative Modeling – RQ-Transformer & Delay Patterns

Table 5: **Ablation study on the use of the RQ-Transformer**. All models are initialized with Helium and pretrained on audio. When not using RQ-Transformer, we predict the 8 levels of tokens with independent classification heads, following Copet et al. (2023). Note that perplexities are only comparable between models with a given delay, as the classification task is easier with more delay for higher tokens.

Acoustic Delay	RQ-Transformer	Perplexity
[0, 1, 2, 3, 4, 5, 6, 7]		42.2
[0, 1, 2, 3, 4, 5, 6, 7]	✓	40.3
[0, 2, 2, 2, 2, 2, 2, 2]		135.4
[0, 2, 2, 2, 2, 2, 2, 2]	✓	36.8

Table 6: **Ablation study on delay patterns, weight of the semantic token and Inner Monologue**. All models are initialized with Helium, pretrained on audio and use the RQ-Transformer. We vary the weight of the semantic token while keeping the weight of other tokens (including the text token when using Inner Monologue) to 1. As different delay patterns cannot be compared in terms of perplexity, we generate continuations from 3s prompts on the valid set, convert them into transcripts with Whisper (Radford et al., 2023) and report their negative log-likelihood with LiteLlama-460M-1T[13] along with their length (in characters) as proxies for linguistic quality.

Acoustic Delay	Semantic Token Weight	Depthwise Parametrization	Inner Monologue	Transcript NLL (↓)	Transcript Length (↑)
[0, 0, 0, 0, 0, 0, 0, 0]	1.0	✓		4.36	486
[0, 1, 1, 1, 1, 1, 1, 1]	1.0	✓		4.12	529
[0, 2, 2, 2, 2, 2, 2, 2]	1.0	✓		4.09	519
[0, 2, 2, 2, 2, 2, 2, 2]	100.0			3.75	538
[0, 2, 2, 2, 2, 2, 2, 2]	100.0	✓		3.65	602
[0, 2, 2, 2, 2, 2, 2, 2]	100.0	✓	✓	**2.77**	**1920**

Generative Audio Modeling – Inner Monologue

- Moshi also models the textual representation of its own speech – increases linguistic quality over pure audio domain

- Text stream W – apply SentencePiece tokenizer to audio transcriptions corresponding to Moshi (with Whisper) to obtain sequence of text tokens. Insert W as first sub-sequence in V (acts as prefix to generation of semantic tokens)

- Do not use text representation from user stream (real-time is challenging; not rely on external ASR system)

- **Align text with audio tokens –** align with constant framerate (12.5Hz). Leverage word-level timestamp from Whisper.
  – Special tokens – PAD and EPAD (never appear). About 65% padding tokens in English conversational speech.
  – W_t initialized with PAD tokens until next word. EPAD inserted before next word to indicate end of padding.
  – Do not insert an EPAD token if it would overwrite a text token from a previous word

- Add more delay between text sequence (W_t) and audio tokens ($A_{t,q}$). Controls which modality the language model will take decision about the content of generated audio.

Generative Audio Modeling – Inner Monologue

Joint sequence modeling for Moshi. Putting together the multi-stream and inner monologue, we have the final set V of sequences to model defined as

$$\begin{cases} V_{s,1} & = W_s & & \text{aligned text tokens.} \\ V_{s,2} & = A_{s,1} & & \text{semantic tokens of Moshi.} \\ V_{s,1+q} & = A_{s-\tau,q} & \text{if} \quad s \geq \tau + 1, 1 < q \leq Q & \text{delayed acoustic tok. of Moshi.} \\ V_{s,1+Q+1} & = A'_{s,1} & & \text{semantic tokens of } \textit{other}. \\ V_{s,1+Q+q} & = A'_{s-\tau,q} & \text{if} \quad s \geq \tau + 1, 1 < q \leq Q & \text{delayed acoustic tok. of } \textit{other}, \end{cases} \quad (6)$$

Inference of Moshi

The joint sequence (Eq. 6) is the target for our modeling task at **train time**
– At any time step s, the model is input with $V_s(0, V_1, \ldots, V_{s-1})$ and output an estimated probability distribution $\hat{V}_s(0, V_1, \ldots, V_{s-1})$

At **inference time**, sample from $\hat{V}_{s,k}$ for all sub-sequence indexes of Moshi's outputs.
– For k = 1 for the text tokens corresponding to Moshi's speech
– For k ∈ {2, . . . , 2+Q} for Moshi's audio tokens

In an application setting, prediction for the audio coming from the user (k > 2+Q) is actually ignored, as the actual user audio is used instead

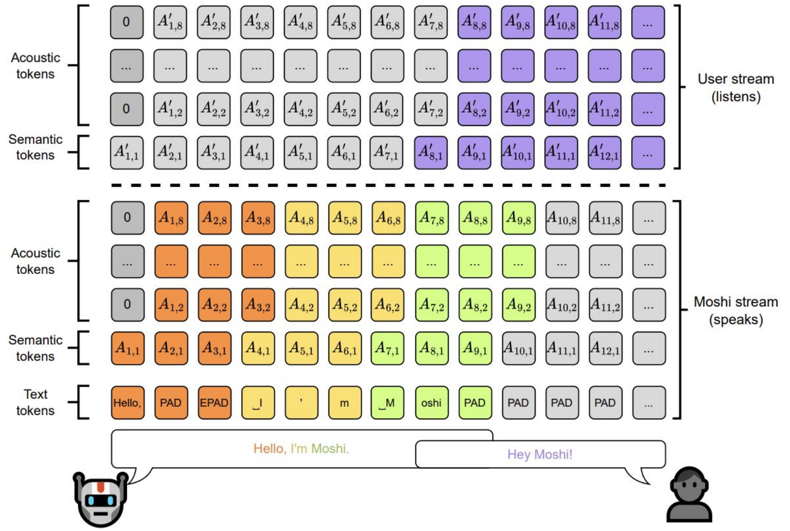


Figure 4: **Representation of the joint sequence modeled by Moshi**. Each column represents the tokens for a given step in the joint sequence ($V_{s,k}$) described in Equation 6 with an acoustic delay $\tau = 1$, e.g. the input of the Temporal Transformer for this step. Tokens are predicted from bottom to top in the Depth Transformer. At inference time, tokens under the dashed line (corresponding to Moshi) are sampled, while those above are fed from the user. This design allows for our model to handle overlapping speech turns.

Audio Language Modeling – Ablations

Initialized with Helium for Temporal Transformer, pretrained on audio data.

Metrics (NLL)
– sWUGGY – model's ability to learn from a lexicon – comparing likelihood of an existing word and an invalid variant (e.g. "oxidation", "accidation")

– sBLIMP evaluates syntactic contrasts

– Spoken StoryCloze evaluates semantic contrasts by comparing commonsense five-sentence stories, with last one being either coherent with context or incoherent

– Spoken Topic-StoryCloze, a variant of the above where the negative continuation is randomly sampled among unrelated sentences (resulting in higher scores)

– MMLU for text understanding evaluation

Baselines
Audio-only models
– GSLM, AudioLM, TWIST-1.3B

Audio-only warm start from a pre-trained text LM
– TWIST 13B, VoxtLM, Spirit-LM

Multimodal – joint training on text and audio
– VoxtLM, SpiritLM

Table 7: **Performance of audio and text language modeling**. We report accuracies based on scoring with negative log-likelihood, normalized by sequence length. MMLU is evaluated in a 5-shot setting. Reusing the terminology of Nguyen et al. (2024), ∅ represents unsupported modalities while - represents unreported numbers.

	Audio metrics				Text metrics
Model	sWUGGY	sBLIMP	sTopic-StoryCloze	sStoryCloze	MMLU
Audio only - Cold Start					
GSLM (Lakhotia et al., 2021)	64.8	54.2	66.6	53.3	∅
AudioLM (Borsos et al., 2022)	71.5	**64.7**	-	-	∅
TWIST (Hassid et al., 2023)	72.2	56.5	-	-	∅
Moshi	**74.8**	59.9	80.9	56.9	∅
Audio only - Warm Start					
TWIST (Hassid et al., 2023)	74.5	59.2	76.4	55.4	∅
VoxtLM (Maiti et al., 2023)	62.9	53.9	-	-	∅
Spirit-LM (Nguyen et al., 2024)	69.5	58.0	72.9	54.8	∅
Moshi	74.3	58.9	81.8	58.7	∅
Text and audio - Warm Start					
VoxtLM (Maiti et al., 2023)	66.1	57.1	-	-	∅
Spirit-LM (Nguyen et al., 2024)	69.0	58.3	82.9	61.0	36.9
Moshi after single-stream pretraining	72.6	58.8	83.0	60.8	**49.8**
Moshi after multi-stream instruct	63.0	55.2	**83.6**	**62.7**	49.7
Moshi after multi-stream instruct, synthetic voice	60.9	54.6	82.5	60.9	48.7

Spoken Question Answering

- Metric: Spoken Web Questions and Llama Questions
- Baselines: GSLM, AudioLM, TWIST, SpeechGPT
- Results:

Model	Web Q.	LlaMA Q.	Audio Trivia QA
Audio only			
GSLM (Lakhotia et al., 2021)	1.5	4.0	-
AudioLM (Borsos et al., 2022)	2.3	7.0	-
TWIST (7B) (Hassid et al., 2023)	1.1	0.5	-
Moshi (w/o Inner Monologue)	**9.2**	**21.0**	7.3
Text and audio			
SpeechGPT (7B) (Zhang et al., 2024a)	6.5	21.6	14.8
Spectron (1B) (Nachmani et al., 2024)	6.1	22.9	-
Moshi	**26.6**	**62.3**	**22.8**
Text			
Helium (text)	32.3	75.0	56.4

Quality and Statistics of Generated Dialogues

- Metric: samples, temp,cond, PPL, IPU, Pause, Gap, Overlap
- Results:

Model	samples	temp	cond. PPL	IPU	Pause	Gap	Overlap
Best non-cascaded (Nguyen et al., 2023)	50	1.0	195.9	41.4s	13.8s	10.7s	6.1s
Cascaded (Nguyen et al., 2023)	50	1.0	45.9	54.8s	0.0s	5.3s	0.0s
Ground Truth (Nguyen et al., 2023)	50	∅	65.0	53.5s	5.5s	4.4s	3.6s
Moshi	1000	0.8	**41.9**	35.1s	13.2s	12.5s	1.2s
Moshi	1000	0.9	56.7	44.7s	9.1s	7.5s	2.2s
Moshi	1000	1.0	79.3	50.8s	7.0s	4.5s	4.1s
Ground Truth	1000	∅	59.6	51.1s	6.4s	4.2s	3.3s

Compressing Moshi and Impact on Speech Quality

- Linguistic impact of model compression on Helium

BF16A8 ($\sim$ 15GB)	54.3		
Bitwidth vs Block size	per-channel	256	32
W8A8 (7.66GB)	53.96	54.09	53.81
W6A8 (6.02GB)	53.50	53.55	53.86
W5A8 (5.20GB)	52.80	53.22	52.76
W4A8 (4.37GB)	49.29	50.84	52.97
W3A8 (3.55GB)	25.49	44.15	50.85
W2A8 (2.73GB)	23.87	23.00	24.27

Compressing Moshi and Impact on Speech Quality

- Linguistic impact of model compression on Moshi

a) Moshi after single-stream pretraining

BF16A8 (15.24GB)	49.8	
Bitwidth vs Block size	256	32
W8A8 (8.33GB)	48.8	48.5
W6A8 (6.95GB)	48.5	49.1
W5A8 (6.02GB)	47.4	48.5
W4A8 (4.64GB)	44.7	45.7
W3A8 (3.72GB)	26.1	35.7
W2A8 (2.80GB)	23.4	24.4

b) Moshi after multi-stream instruct

BF16A8 (16.74GB)	49.7	
Bitwidth vs Block size	256	32
W8A8 (9.20GB)	47.6	47.6
W6A8 (7.70GB)	48.1	48.3
W5A8 (6.69GB)	46.7	47.2
W4A8 (5.18GB)	39.8	42.2
W3A8 (4.18GB)	27.7	29.9
W2A8 (3.17GB)	24.5	24.9

Compressing Moshi and Impact on Speech Quality

- Distribution of audio artifacts caused by model compression

Model / Artifacts	Gibberish audio	Noisy audio	Background noise	Repetitive text	No artifacts
unquant		4.1	0.1	0.1	95.8
W4A8, block=32		3.8	0.1	0.4	95.7
W4A8, block=256	0.1	3.7		2.2	94.0
W3A8, block=32	0.5	4.7	5.9	8.1	80.7
W3A8, block=256	0.2	12.2	3.1	21.9	62.7
W2A8, block=32	12.7	40.9	0.5	0.4	45.4
W2A8, block=256	83.1			11.0	5.9

Safety

Safety

In parallel with the development of Moshi, the paper also explores and evaluates the safety of the AI generated content from the following aspects.

- Toxicity Analysis
- Regurgitation Analysis
- System Voice Consistency
- Identification of the Content Generated by Moshi: Watermarking

Toxicity Analysis

- Adopts the **ALERT benchmark** (Tedeschi et al., 2024), which evaluates safety under multiple categories (hate, self-harm, weapon, crime, sex, substance), to evaluate the text content produced by the model

Category	Moshi	GPT-3.5	GPT-4	Llama 2	Alpaca	Vicuna	Falcon	Mistral	Mixtral	Zephyr	OLMo
Overall Safety Score	83.05	96.95	99.18	99.98	62.13	95.75	88.11	75.45	98.22	77.86	85.90

Results

- Moshi falls into the middle of this table in terms of rank
- acceptable safety standards
- Room for Growth: possibly by incorporating larger datasets, more feedback loops, and more specialized training...

Regurgitation Analysis

Regurgitation refers to a model **reproducing** exact segments or subsequences from its training data during content generation

Potential Risks:

- **Copyright Issues:** Reproducing copyrighted or unauthorized content (e.g., music, speech excerpts).
- **Privacy Concerns:** Regenerating speech content that contains sensitive or confidential information.
- **Credibility Problems:** If users find the generated content overly "familiar," they may question the model's originality.

Unlike text models, speech models face unique risks: They can regurgitate not only **text** but also **voice pitch, tone**, and even **background audio elements**.

Regurgitation Analysis

The paper evaluates the **regurgitation rate** under 4 conditions: the presence or absence of prompts, whether the data was deduplicated, whether the model was fine-tuned, and different sampling temperatures. The results are shown in the table below.

	prompted (3s)	deduplicated	fine-tuned	temp.	regurgitation rate (%)
single-stream				0	0.00
				0.6	0.13
				0.8	0.19
				1.0	0.16
	✓			0	100.00
	✓			0.8	98.40
		✓		0	0.00
		✓		0.8	0.00
	✓	✓		0	0.00
	✓	✓		0.8	0.00
multi-stream			✓	0.8	0.00
	✓		✓	0.8	0.00
		✓	✓	0.8	0.00
	✓	✓	✓	0.8	0.00

Insights from the result:

- Sampling temperature: the values typically employed for generation (0.6–1.0) are more prone to regurgitation.

- Unconditioned and prompted generation: prompted scenario is more likely to suffer from regurgitation.

- Fine-tuning: decrease regurgitation rate to some extent, but it might be overridden.

- Deduplication: A Critical Step to avoid regurgitation.

Voice Consistency

A potential risk for a speech-to-speech model is unauthorized voice generation. The model should use its **target voice** and not potentially **mimic the user's voice**.

Evaluation Method:

- generated 100 hours of conversations between Moshi and a second synthetic speaker
- used a **speaker verification model (WavLM)** to measure how closely Moshi's voice aligns with its reference voice.

Insights from the result:

- Over the generated datasets, there are 10,249 occurrences (98.7%) where the voice of moshi is closer to itself and 133 occurrences (1.3%) where the voice is closer to the other speaker.
- speaker consistency remains **stable along time**: no drift as the conversation goes on.

segment start time (seconds)	20–25	25–30	30–35	35–40	40–45
samples	2034	2006	1998	2019	1994
main > other	98.4%	99.2%	99.1%	99.2%	99.3%

Identification of Moshi-Generated Content: Watermarking

As AI-generated content becomes more pervasive, it is critical to ensure **accountability** and **traceability**. Watermarking provides a way to distinguish Moshi-generated audio from human speech or other sources, which helps prevent misuse, such as passing AI-generated speech as human or removing ethical safeguards.

Evaluation

The paper investigated if the two types of watermarking can be detected:

- **Signal-based watermarking**: this approach embeds subtle, **inaudible patterns** into the generated audio signal using the Audioseal method.
- **Generative-based watermarking** for audio: this approach modifies the probabilities during audio generation, embedding marks directly into the **token generation process**. (watermarking the generation process itself)

Evaluation of signal-based watermarking

Table 15: **Evaluation of Audioseal** (San Roman et al., 2024b) for watermarking the speech produced by Moshi. Each detection score is averaged over 1000 generations.

			average detection score	
	↓ audio post-processing	audio duration →	10 seconds	1 minute
No mark	none		0.0855	0.2474
Watermarked	none		0.9999	0.9999
Watermarked	pink-noise (noise std $\sigma = 0.2$)		0.7093	0.9019
Watermarked	RVQGAN compression & decompression		0.1101	0.2662
Watermarked	Mimi compression & decompression		0.0805	0.2404

Insights:

- **Unmodified audio** achieves near-perfect detection (0.9999).
- Adding pink noise reduces detection accuracy but still yields reasonable results for longer audio (0.7093 for 10 seconds, 0.9019 for 1 minute).
- Detection drops with **compression** (e.g., RVQGAN, Mimi), making marks indistinguishable.

Exploration on generative-based watermarking

Table 16: **Idempotence of tokens**. Probabilities that quantization indices remain identical after decoding and re-encoding the waveform back to tokens, depending on the residual quantizer level. We consider two optional audio post-processing attacks: audio shifted by a time offset of up to half the sampling period (ΔT=40ms), and re-encoding with RVQGAN. All results are averaged over 1000 generated sequences of 1 minute.

↓ codec	attacks ΔT	attacks RVQGAN	RQ level →	$k=1$ (semantic)	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
Basic	0			0.798	0.783	0.560	0.483	0.421	0.407	0.369	0.404
	10ms			0.766	0.495	0.255	0.206	0.180	0.173	0.144	0.193
	20ms			0.682	0.390	0.220	0.180	0.158	0.154	0.129	0.172
	40ms			0.503	0.329	0.182	0.146	0.128	0.125	0.107	0.156
Mimi	0			0.766	0.550	0.372	0.352	0.293	0.297	0.264	0.303
	10ms			0.731	0.376	0.206	0.176	0.152	0.154	0.132	0.182
	20ms			0.653	0.307	0.171	0.146	0.121	0.126	0.106	0.159
	40ms			0.483	0.267	0.160	0.137	0.116	0.121	0.102	0.150
	0	✓		0.741	0.409	0.221	0.198	0.150	0.154	0.134	0.173
	10ms	✓		0.702	0.281	0.148	0.133	0.118	0.117	0.100	0.136
	20ms	✓		0.633	0.228	0.126	0.114	0.098	0.097	0.084	0.119
	40ms	✓		0.450	0.197	0.120	0.113	0.104	0.102	0.086	0.112

Insights:

- **Idempotence issues:** Audio tokens change after re-encoding, reducing detection reliability.
- **Temporal shifts:** Even moderate time offsets escalate the change.
- Generative watermarking shows potential but **requires more robust token stability**. Future work on resilient watermarking techniques is critical to ensure trust and ethical AI usage.

Thank You!