

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

Alexandre Défossez*

ALEX@KYUTAI.ORG

Laurent Mazaré*

Manu Orsini

Amélie Royer

Patrick Pérez

Hervé Jégou

Edouard Grave*

Neil Zeghidour*

NEIL@KYUTAI.ORG

Kyutai

*Equal contribution

Abstract

We introduce Moshi, a speech-text foundation model and full-duplex spoken dialogue framework. Current systems for spoken dialogue rely on pipelines of independent components, namely voice activity detection, speech recognition, textual dialogue and text-to-speech. Such frameworks cannot emulate the experience of real conversations. First, their complexity induces a latency of several seconds between interactions. Second, text being the intermediate modality for dialogue, non-linguistic information that modifies meaning—such as emotion or non-speech sounds—is lost in the interaction. Finally, they rely on a segmentation into speaker turns, which does not take into account overlapping speech, interruptions and interjections. Moshi solves these independent issues altogether by casting spoken dialogue as speech-to-speech generation. Starting from a text language model backbone, Moshi generates speech as tokens from the residual quantizer of a neural audio codec, while modeling separately its own speech and that of the user into parallel streams. This allows for the removal of explicit speaker turns, and the modeling of arbitrary conversational dynamics. We moreover extend the hierarchical semantic-to-acoustic token generation of previous work to first predict time-aligned text tokens as a prefix to audio tokens. Not only this “Inner Monologue” method significantly improves the linguistic quality of generated speech, but we also illustrate how it can provide streaming speech recognition and text-to-speech. Our resulting model is the first real-time full-duplex spoken large language model, with a theoretical latency of 160ms, 200ms in practice, and is available at github.com/kyutai-labs/moshi.

Keywords: speech, text, multimodal, foundation, spoken dialogue

1 Introduction

Voice has provided a convenient interface to early conversational systems, from Alexa¹ to Siri² and Google Assistant.³ In this context, a “wake word” spoken by the user typically triggers an automatic speech recognition (ASR) system which transcribes the subsequent user’s request. Then, a natural language understanding (NLU) pipeline converts this query to a structured format used to produce a text answer through natural language generation (NLG). Eventually, a text-to-speech (TTS) system tells the answer back to the user. While this process can handle short, constrained interactions (e.g. triggering an action or retrieving a fact), the rise of large language models (LLMs) (Brown et al., 2020; Hoffmann et al., 2022; Touvron et al., 2023a) has called for a consequent extension of voice interfaces to multi-turn, open-ended conversations. A solution to this challenge is handling the NLU and NLG with an LLM, while the ASR and TTS provide the voice interface during the user’s and the system’s turn respectively (Llama, 2024). This framework supports the current generation of spoken dialogue systems such as Gemini (Gemini et al., 2023) or ChatGPT.⁴

Yet, the experience offered by these interfaces remains far from natural conversations. First, latency compounds along the many components of these pipelines, resulting in a typical global latency of several seconds. This is unlike natural conversations which demonstrate response times of a few hundred milliseconds. Second, as language understanding and generation happens in the textual domain, any non-written information is ignored by the model. This goes from paralinguistic information, such as emotion and accent, to non-speech audio, such as surrounding acoustic events. Finally, these models remain fundamentally turn-based, assuming that dialogue is a sequence of well-defined single-speaker segments. While this paradigm is suited to text dialogue, it falls short in modeling aspects of spoken conversations such as interruptions, overlapping speech—which amounts for 10 to 20% of spoken time (Çetin and Shriberg, 2006)—and backchanneling (i.e. non-interrupting interjections such as “OK” or “I see”).

In this work we introduce Moshi, a speech-text foundation model and real-time spoken dialogue system that aims at solving the aforementioned limitations: latency, textual information bottleneck and turn-based modeling. Moshi augments a text LLM backbone with a smaller audio language model (Borsos et al., 2022; Yang et al., 2023) that ingests and predicts discrete audio units. This removes the information bottleneck of text by understanding inputs and generating outputs directly in the audio domain, while benefiting from the knowledge and reasoning abilities of the underlying text LLM. We extend previous work on audio language models and design a streaming, hierarchical architecture, with a theoretical latency of 160 ms—lower than the 230 ms average in natural conversations measured over 10 languages (Stivers et al., 2009). We furthermore introduce the first multi-stream audio language model, i.e. a model that explicitly processes the input and output audio streams jointly into two autoregressive token streams. This altogether removes the concept of speaker turn and thus allows training the model on natural conversations with arbitrary dynamics including overlap and interruptions. Our resulting model is the first *full-duplex*—

1. <https://www.alexa.com>

2. <https://www.apple.com/siri>

3. <https://assistant.google.com/>

4. <https://openai.com/index/chatgpt-can-now-see-hear-and-speak/>

it always listens and always generates sound, either speech or silence—real-time conversational LLM. We summarize our contributions below:

- We present Helium, a 7B-parameter text LLM that we pretrain on 2.1T tokens of public English data. [Section 3.2](#) describes the architecture and training of the model, while [Section 4.1](#) provides details on the pretraining data collection and filtering.
- We train Mimi, a neural audio codec ([Zeghidour et al., 2022](#); [Défossez et al., 2023](#)) that converts audio into the discrete tokens predicted by Moshi and back, using residual vector quantization (RVQ). Audio language models typically combine such *acoustic* tokens with *semantic* tokens from a self-supervised speech model as it is necessary to produce intelligible speech in absence of text conditioning ([Borsos et al., 2022](#)). We rather extend the approach of [Zhang et al. \(2024b\)](#) by distilling semantic information into the first level of acoustic tokens and introduce improved training tricks. [Section 3.3](#) describes the architecture and training of Mimi while [Section 5.2](#) details ablation studies.
- We propose Moshi, a new architecture for audio language modeling, which combines Helium with a smaller Transformer ([Vaswani et al., 2017](#)) model to predict audio tokens in a hierarchical and streaming fashion. We show how challenging it is for such unconditioned audio language models to generate intelligible speech, and we provide solutions that outperform the intelligibility and audio quality of non-streaming models while generating audio in a streaming fashion. We furthermore extend this architecture to model several audio streams in parallel, allowing for a conceptually and practically simple handling of full-duplex dialogues with arbitrary dynamics. [Section 3.4](#) describes this architecture.
- In [Section 3.4.4](#), we introduce Inner Monologue, a new training and inference setup for audio language models that significantly improves the factuality and linguistic quality of generated speech by predicting time-aligned text tokens before audio tokens. Moshi is a speech-to-speech model as it allows reasoning about non-linguistic information, both from the user audio and from Moshi’s audio. Yet, this is not incompatible with Moshi producing text along its speech output. Based on the past observation ([Borsos et al., 2022](#); [Zhang et al., 2024b](#)) that coarse-to-fine generation (from semantic to acoustic tokens) is critical to generating consistent speech, we extend this hierarchy to using text tokens as a per-timestep prefix to the semantic token. Our experiments show that not only this drastically improves the length and quality of generated speech, but we also show how forcing a delay between text and audio tokens allows deriving streaming ASR and streaming TTS from a Moshi model.
- We evaluate all components of Moshi along several axes, including text understanding, speech intelligibility and consistency, audio quality and spoken question answering. Our experiments, reported in [Section 5](#), show that our model is state of the art among existing speech-text models for speech modeling and spoken question answering while being streaming compatible and able to model several minutes of context (5 min in our experiments).

We encourage the reader to talk to Moshi using our web demo.⁵

5. <https://moshi.chat/>

2 Related Work

Audio Language Modeling. Early developments in speech foundation models have improved speech understanding across many discriminative tasks, from automatic speech recognition (ASR) (Baevski et al., 2020; Radford et al., 2023; Zhang et al., 2023b) to speaker verification (Chen et al., 2022) and speech classification (Yang et al., 2021). A key factor in this development is self-supervised learning (Hsu et al., 2021; Baevski et al., 2020; Chen et al., 2022) which allows learning generic, discriminative speech representations. As these speech understanding models build on previous work done on masked language modeling for text (Devlin et al., 2019), generative text pretraining (Radford et al., 2018) has similarly inspired a large family of speech generation models. In particular, Lakhota et al. (2021) propose quantizing aforementioned self-supervised representations. The resulting discrete *audio tokens* represent a speech segment as a sequence of categorical variables, thus casting speech generation as a language modeling task. AudioLM (Borsos et al., 2022) furthermore combines these *semantic* tokens with *acoustic* tokens from a neural audio codec (Zeghidour et al., 2022), which allows for modeling arbitrary voices, recording conditions and non-speech sounds. These audio language models have redefined the state of the art in speech generation, from text-to-speech (Wang et al., 2023; Kharitonov et al., 2023) to speech-to-speech translation (Rubenstein et al., 2023; Reid et al., 2024) and speech enhancement (Yang et al., 2023). Beyond these supervised tasks, a parallel line of work has explored training and scaling unsupervised audio-only models, trained for autoregressive speech generation (Dunbar et al., 2021; Lakhota et al., 2021; Borsos et al., 2022). The abilities of these models have progressively expanded, from generating short sentences in a single speaker voice (Lakhota et al., 2021) to producing meaningful and consistent speech continuations across dozens of seconds in arbitrary voices and conditions (Borsos et al., 2022), thanks to a hierarchical modeling of semantic and acoustic tokens. A main challenge is that audio requires the modeling of long sequences, up to a few minutes, to produce meaningful and exploitable outputs. However, latent representations for audio are typically less compact than equivalent representations for text. Thus, discrete representations from neural audio codecs require multiple predictions per timestep when modeled autoregressively. (Liu et al., 2023b) and (Evans et al., 2024) use latent diffusion (Ho et al., 2020) for general audio and music modeling to alleviate the need for hierarchical discrete tokens. However, these methods cannot be used in a streaming fashion, and it is unclear whether they could generate consistent speech. Copet et al. (2023) instead show that the number of auto-regressive steps can be reduced by introducing a delay between the different levels of tokens, and performing parallel prediction over them. Inspired by the RQ-Transformer method by Lee et al. (2022) and the hierarchical MegaByte transformer model (Yu et al., 2024), Yang et al. (2023) and Zhu et al. (2024) leverage a smaller nested transformer to model the different tokens at a single time step. In this work, we extend these previous works to push the limits of autoregressive speech generation by proposing a scalable hierarchical modeling of audio tokens which can handle several minutes of context while generating audio in real time. Still, while speech-only models learn linguistic structure—lexicon, syntax, semantics—from raw speech (Dunbar et al., 2021), they typically demonstrate poor-to-nonexistent factual knowledge and reasoning abilities. This has led to the development of speech-text models, intended to combine the knowledge and reasoning abilities of text models with the generative power of audio models.

Speech-text Models. Such models typically start from a pretrained text language model and either finetune it to predict audio (Hassid et al., 2023), or propose a speech-text finetuning task (Rubenstein et al., 2023; Maiti et al., 2023; Nachmani et al., 2024; Nguyen et al., 2024; Mitsui et al., 2024; Zhang et al., 2024a): For instance, AudioPALM (Rubenstein et al., 2023) starts from a pretrained PALM (Chowdhery et al., 2022) model, and extends its text vocabulary with semantic audio tokens. Then, the model is trained for a mixture of speech-text tasks, including TTS, ASR and speech-to-speech translation. VoxTLM (Maiti et al., 2023) adopts a similar approach for TTS and ASR. While these models are trained in a supervised fashion with specific input and output sequences, Spirit-LM (Nguyen et al., 2024) uses temporal alignment between speech and its transcript to perform modality switch (from speech tokens to text tokens, or conversely) inside a sequence. This allows the model to learn consistent internal representations of language regardless of it being represented as text or speech, as measured through commonsense evaluation. Another approach, adopted by Spectron (Nachmani et al., 2024), SpeechGPT (Zhang et al., 2023a) and PSLM (Mitsui et al., 2024), combines speech and text in a hierarchical manner rather than as interchangeable representations. Similar to how AudioLM (Borsos et al., 2022) decomposes speech generation into predicting semantic tokens and *then* acoustic tokens, Spectron and SpeechGPT use a “Chain-of-Modality” and first produce an utterance as text tokens, subsequently used as a prefix to generate speech. This allows guiding speech generation with the output of an underlying text LLM, however this is fundamentally incompatible with live interactions as the model needs to produce an entire answer as text before it starts speaking. PSLM alleviates this limitation by modeling text and speech tokens in parallel. In this work, we propose Inner Monologue as a main architectural and training component to combine aligned text and speech data. Inner Monologue decomposes speech into a chain of text, semantic and acoustic tokens, and predicts this structured sequence in a hierarchical manner. Unlike Spirit-LM, this allows representing all utterances both as text and speech, rather than switching between modalities; In addition, the integration of acoustic tokens into the same generative model enables generating arbitrary voices and conditions, rather than a single speaker. Besides, this hierarchical modeling described in Section 3.4.4 allows decomposing the generation task without increasing the sequence length of the Transformer (Vaswani et al., 2017) outputs, unlike Chain-of-Modality, while benefiting from producing text a prefix to audio tokens rather than in parallel like PSLM. Moreover, Inner Monologue decomposes speech on a per-frame basis, which means that each prediction step outputs a speech frame. This is unlike Spectron and SpeechGPT which require generating a complete sequence as text before generating audio tokens, and this makes Moshi compatible with real-time generation. Moreover, we show in Section 3.4.4 how Inner Monologue, when combined with a delay between token types, allows deriving streaming TTS and ASR systems from Moshi. Finally, while Spectron, SpeechGPT and PSLM model both user and system speech and text tokens into a single stream, which requires properly segmented turns, Moshi benefits from a novel multi-stream architectures which removes the concept of speaker turns and allows for modeling any type of overlap, interruptions and interjections.

Spoken Dialogue Models. Spoken dialogue is one of the less explored tasks in speech generation, as it requires addressing several challenges: 1) The model should run in real time and allow for long conversations in full-duplex—the model always listens and can speak

at any moment; 2) it should be speech-to-speech to handle paralinguistic communication; 3) it should display knowledge and reasoning abilities that make it amenable to helpful and enjoyable conversations. Spectron benefits from its underlying text LLM (as measured by spoken question answering), however it is not compatible with real-time generation due to Chain-of-Modality. PSLM proposes generating speech and text tokens in parallel to reduce this latency, however it reduces the quality of answers, and the model still relies on ASR, which removes paralinguistic information. More importantly, these models cannot handle full-duplex communication, where there is no boundary between speaker turns, as any side of the conversation can be active at any time. An attempt at modeling these dynamics has been proposed by Wang et al. (2024), with an ASR system running in parallel to a text generator that feeds into a streaming TTS. While this allows modeling more complex scenarios than previous approaches, it still relies on a cascaded pipeline and models both the user’s and the system’s speech into a single token stream, which is challenging in presence of significant overlap. The only previous full-duplex dialogue system is dGSLM (Nguyen et al., 2023), which models user and system speech as separate audio token streams and proposes a Siamese architecture to process both streams jointly. While dGSLM is full-duplex, it remains a proof-of-concept: it does not run in an online fashion, it does not benefit from the knowledge of a text language model, and it does not model acoustic information as it only models semantic tokens. Moshi addresses these limitations altogether: by modeling two streams of semantic and acoustic tokens hierarchically, Moshi is full duplex and can exploit all the information from the user (linguistic and non-linguistic) while producing speech in real time. Thanks to text pretraining and Inner Monologue, Moshi benefits from the knowledge of its Helium backbone. Finally, as the model produces acoustic tokens along with text and semantic tokens, it can generate an arbitrary range of emotions, voices and acoustic conditions. To the best of our knowledge, Moshi is the first audio language model that successfully addresses the many aforementioned challenges of spoken dialogue.

3 Model

3.1 Overview

Moshi is a multi-stream speech-to-speech Transformer model, which allows for full-duplex spoken dialogue with a user thanks to an innovative architecture summarized in Figure 1. Moshi is built on top of Helium, a text LLM which we build from scratch (Section 3.2), relying on high-quality text data to provide strong reasoning abilities to the model. We also propose Inner Monologue (Section 3.4.4), a training and inference procedure in which we jointly model text and audio tokens. This allows the model to fully exploit the knowledge imparted from the text modality, while remaining a speech-to-speech system. To enable real-time dialogue, we also design Moshi as a multi-stream architecture from the get-go (Section 3.4.3): The model is able to both speak and listen to the user at the same time, and does not need to explicitly model speaker turns. In addition, to capture the input user audio and output Moshi’s voice with high quality and in an efficient manner, we propose Mimi (Section 3.3), a neural audio codec combining semantic and acoustic information into a single tokenizer by using residual vector quantization and knowledge distillation. To jointly model the audio streams from Moshi and the user, as well as Moshi’s text tokens, we rely on a Depth Transformer compatible with streaming inference (Sections 3.4.1, 3.4.2).

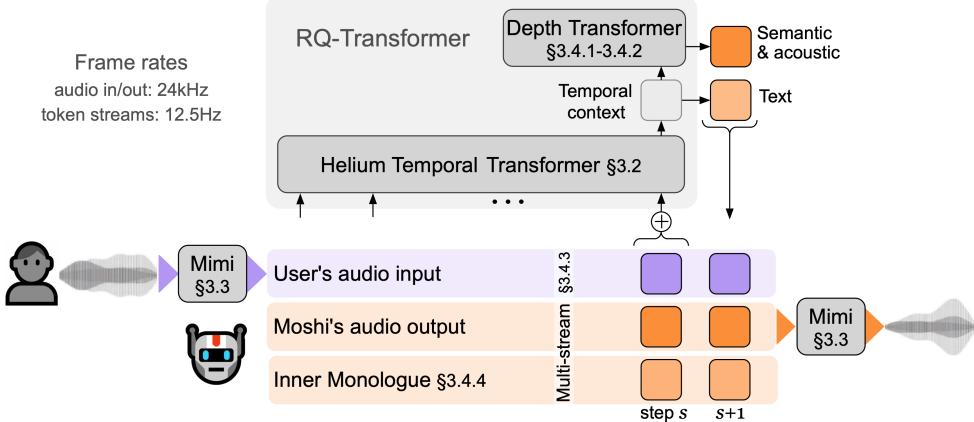


Figure 1: **Overview of Moshi.** Moshi is a speech-text foundation model which enables real-time spoken dialogue. The main components of Moshi’s architecture are: a bespoke text language model backbone (Helium, see [Section 3.2](#)); a neural audio codec with residual vector quantization and with semantic knowledge distilled from a self-supervised speech model (Mimi, [Section 3.3](#)); the streaming, hierarchical generation of semantic and acoustic tokens for both the user and Moshi, along with time-aligned text tokens for Moshi when using Inner Monologue ([Section 3.4](#)).

In this section, we further detail each of these components. We then describe the training datasets and the different training phases we used to train Moshi in [Section 4](#). Finally, in [Section 5](#), we report thorough evaluation results on Moshi’s abilities, both linguistic and acoustic, as well as ablation experiments on its main components, while [Section 6](#) provides analyses on the safety of our system.

3.2 The Helium Text Language Model

3.2.1 ARCHITECTURE

Helium is an autoregressive language model, based on the Transformer architecture ([Vaswani et al., 2017](#)). Following previous work in this area, we make the following changes to the original architecture: First, we use RMS normalization ([Zhang and Sennrich, 2019](#)) at the input of the attention blocks, the feed-forward blocks and the output linear layer of the model. We use rotation positional embeddings ([Su et al., 2024](#), RoPE), a context length of 4,096 tokens and FlashAttention ([Dao et al., 2022](#)) for efficient training. Finally, we change the architecture of the feed-forward blocks and use Gated Linear Units ([Shazeer, 2020](#)), with the SiLU activation as a gating function ([Hendrycks and Gimpel, 2016b](#)). Our tokenizer is based on the unigram model from SentencePiece ([Kudo and Richardson, 2018](#)), and contains 32,000 elements mostly targeting English. We split all numbers into single digits, and use byte-backoff to ensure that our tokenizer does not lose information. We train the model with the AdamW ([Loshchilov and Hutter, 2017](#)) optimizer, with a fixed learning rate followed by a cosine learning rate decay ([Loshchilov and Hutter, 2016](#)).

Table 1: Models’ hyper-parameters. Hyper-parameters for the architecture and training of our 7B-parameter Helium language model and of Moshi, our speech-text dialogue model. The training of Moshi goes through 4 phases: Pre-training on unsupervised data (with Temporal Transformer initialized from Helium); Post-training with simulated multi-stream based on diarization; Fine-tuning on the Fisher dataset (Cieri et al., 2004) to gain its fully duplex capabilities; Instruction fine-tuning on a custom dataset built from synthetic interaction scripts. During the pre-training phase, we keep training half of the time on full text batches from the same dataset as used for Helium, using a separate optimizer state.

Hyper-parameter	Helium training		Moshi training		
	pre-training	pre-training	post-training	fisher	fine
<i>Temporal Transformer</i>					
Model dimension	4096				
MLP dimension	11264				
Number of heads	32			same	
Number of layers	32				
Context size	4096		3000 steps, e.g. 4 min.		
Learning rate	$3 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$3 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	$2 \cdot 10^{-6}$
<i>Depth Transformer</i>					
Model dimension	-		1024		
MLP dimension	-		4096		
Number of heads	-		16		
Number of layers	-		6		
Learning rate	-	$2 \cdot 10^{-4}$	$5 \cdot 10^{-5}$	$4 \cdot 10^{-6}$	$2 \cdot 10^{-6}$
<i>Input / Output space</i>					
Text cardinality	32000		32000		
Audio cardinality	-		2048		
Frame rate	-		12.5 Hz		
<i>Common parameters</i>					
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

3.2.2 PRE-TRAINING DATA FILTERING

Training data is one of the critical ingredients to train LLMs: we now describe our method to obtain a large and high-quality text dataset. We start from high-quality data sources, such as Wikipedia, Stack Exchange and a large collection of scientific articles. As the quantity of data from these sources is too small to train a LLM, we also rely on web crawled data, specifically from CommonCrawl, to extend our dataset. See more details on data sources in Section 4.1. Web data requires extensive processing to obtain a high-quality training set: we perform deduplication, language identification and quality filtering. In the following, we describe each operation in more details.

Deduplication. We start from the WET files, which contain only the text content of web-pages, which was extracted by the CommonCrawl project. Because this format contains all the text of a page, it includes a lot of boilerplate such as navigation menus. Thus, the first step of our pipeline is to deduplicate each shard (there is 100 shards per crawl) at the *line level*, to remove this boilerplate. To do so, we compute the FNV-1a⁶ hash of each line, and use a bloom filter to remove duplicates. We also train a `fastText` (Joulin et al., 2016) classifier on duplicates *vs.* non-duplicates, to perform fuzzy deduplication: here we only remove blocks of at least 3 consecutive lines that are classified as duplicates.

Language identification. Once deduplication is performed, we apply a language identifier based on `fastText` to keep English data only. Language identification is performed at the document level, and we only keep documents above a certain threshold (0.85).

Quality filtering. The last step is to filter the remaining data, to keep high-quality web-pages only. To perform this step, we train a `fastText` classifier on lines from our high quality data sources and from random CommonCrawl webpages. We obtain a classifier with 9 categories, corresponding to our different high quality sources such as Wikipedia or Wikibooks and to subsets of StackExchange such as STEM or humanities. The motivation is to obtain a finer control over which documents to keep, not only based on similarity to high quality sources, but also based on their domains. This classifier is applied at the *line level*, and an aggregated score is obtained by computing the average scores of each line, weighted by their length. Again, we keep documents corresponding to scores above a certain threshold.

3.3 Audio Tokenization

To discretize waveforms into audio tokens, we introduce Mimi, a neural audio codec (Zeghidour et al., 2022; Défossez et al., 2023) that operates as an autoencoder with a discrete bottleneck (van den Oord et al., 2017). In the literature, and following the terminology defined by Borsos et al. (2022), these tokens are referred to as *acoustic* tokens, as they model fine audio details and are optimized for high-quality reconstruction. While these acoustic tokens provide appropriate targets for conditioned text-to-audio models (e.g. text-to-speech (Wang et al., 2023) or text-to-music (Copet et al., 2023)), unconditioned speech generation requires combining them with *semantic* tokens extracted from self-supervised speech models (Baevski et al., 2020; Hsu et al., 2021; Chung et al., 2021). Unlike their acoustic counterpart, semantic tokens do not allow for reconstructing high-quality audio but correlate strongly with linguistic content. This similarity with language allows generating intelligible and consistent speech, even without text conditioning, by using semantic audio tokens as a prefix to predicting acoustic tokens. Yet, this hybrid tokenization approach is not compatible with real-time generation. Semantic tokens are typically not causal and can thus only be computed in an offline manner. Moreover, generating acoustic and semantic tokens with separate encoders represents a non-negligible computational burden. Consequently, and taking inspiration from previous work on SpeechTokenizer (Zhang et al., 2024b), Mimi 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.

6. <http://www.isthe.com/chongo/tech/comp/fnv>

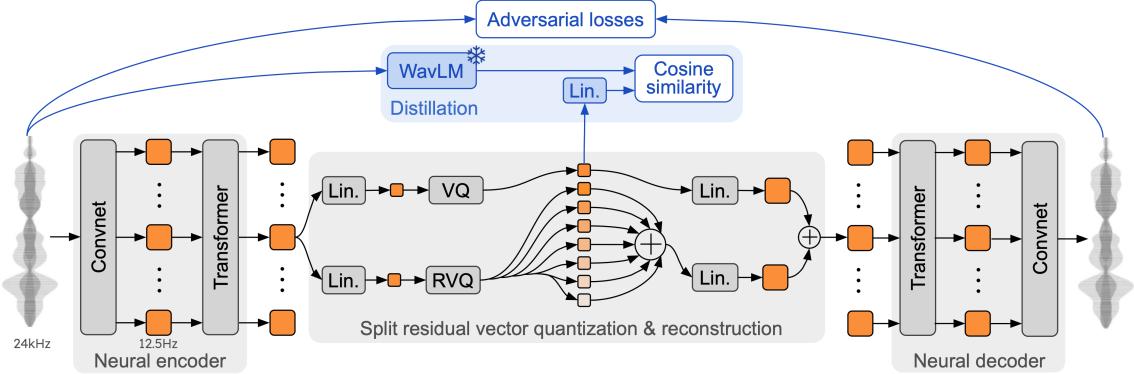


Figure 2: **Architecture and training of Mimi, our neural audio codec, with its split residual vector quantization.** During training (blue part, top), we distill non-causal embeddings from WavLM (Chen et al., 2022) into a single vector quantizer which produces semantic tokens, and is combined with separate acoustic tokens for reconstruction.

3.3.1 ARCHITECTURE

Our baseline architecture takes inspiration from SoundStream (Zeghidour et al., 2022) and Encodec (Défossez et al., 2023) and consists of a SeaNet (Tagliasacchi et al., 2020) autoencoder and a Residual Vector Quantizer (Zeghidour et al., 2022). The encoder projects a single-channel waveform $x \in \mathbb{R}^L$ to a latent representation $\text{enc}(x) \in \mathbb{R}^{S \times D}$ by cascading residual convolutional blocks that interleave dilated (van den Oord et al., 2016) and strided convolutions along with ELU (Clevert et al., 2016) non-linearities and Weight Normalization (Salimans and Kingma, 2016). All convolutions are causal, such that this autoencoder can run in a streaming fashion. With 4 convolutional blocks and respective striding factors (4, 5, 6, 8), and a final 1D convolution with stride 2, Mimi’s encoder projects a 24kHz waveform to a latent representation of 12.5 frames per second and dimension $D = 512$. Symmetrically, the decoder adopts a similar structure but with transposed convolutions rather than strided ones, to project the latent representation back to 24kHz audio. We discretize the latent space with a Residual Vector Quantizer (Zeghidour et al., 2022), which iteratively applies vector quantization (VQ) to the residuals of the previous quantizer. With Q quantizers, each with a codebook of N_A centroids, the RVQ discretizes the latent space into $\{1, \dots, N_A\}^{S \times Q}$. As a baseline, we train this model with a combination of reconstruction and adversarial losses, following the setup of Encodec (Défossez et al., 2023). We detail below the main changes of Mimi with respect to this default configuration.

Transformer-based bottleneck. To improve the ability of Mimi to encode speech into compact representations while reconstructing high-quality audio, we add Transformer modules in the bottleneck, one right before quantization and one after. These Transformers have 8 layers, 8 heads, RoPE position encodings, a finite context of 250 frames (20 seconds), GELU (Hendrycks and Gimpel, 2016a) activations, a model dimension of 512 and an MLP dimension of 2048. To stabilize training, we use LayerScale (Touvron et al., 2021), with initialization of the diagonal values at 0.01. Both Transformers use causal masking, which preserves the compatibility of the whole architecture with streaming inference. Both

Transformers prove to be useful with regard to perceived audio quality, while Transformer in the encoder also improves the distillation of semantic information described below (see [Table 3](#) for ablation studies).

Causality and streaming. With the mentioned hyper-parameters, Mimi is causal and can be used in a streaming fashion, both for encoding and decoding. Both its initial frame size and overall stride correspond to 80ms, meaning that given a first audio frame of 80ms, Mimi outputs a first latent timestep, which can be decoded to 80ms of output audio.

Optimization. Unlike purely convolutional codecs that use Adam ([Kingma and Ba, 2015](#)), the introduction of Transformers into the architecture requires additional regularization with weight decay along with using the AdamW ([Loshchilov and Hutter, 2019](#)) optimizer. More precisely, we apply weight decay only to the parameters of the Transformers, with a weight of $5 \cdot 10^{-2}$. We use a learning rate of $8 \cdot 10^{-4}$, a momentum decay of 0.5 and a decay of the squared gradient of 0.9, and an exponential moving average of weights with a decay of 0.99. We train with a batch size of 128 on random windows of 12s, for 4M steps, while the context of Transformers is limited to 10s (250 frames before the last downsampling layer of the encoder, and symmetrically for the decoder).

Quantization rate. We use $Q = 8$ quantizers, each with a codebook size of $N_A = 2048$. At 12.5Hz, this represents a bitrate of 1.1kbps. While the latent dimension is 512, we project embeddings to 256 dimensions before applying the RVQ, and project back to 512 before the decoder. Consistently with previous work, we use quantizer dropout ([Zeghidour et al., 2022](#)) to provide the codec with bitrate scalability. We moreover follow the observation of [Kumar et al. \(2023\)](#) that not applying quantization with a certain probability during training improves audio quality. More precisely, we only apply quantization 50% of the time, on a per-sequence basis, during training. Unlike [Kumar et al. \(2023\)](#), this means passing unquantized embeddings to the decoder, rather than passing embeddings quantized with all quantizers. [Table 3](#) shows that this significantly improves objective quality metrics, while human evaluations are not conclusive. Across our experiments, we make the somehow counter-intuitive observation that this gain gets more significant as we lower the bitrate.

Adversarial-only training. As a baseline, we train Mimi with the same combination of reconstruction and adversarial losses as [Défossez et al. \(2023\)](#), namely a multi-scale mel-spectrogram reconstruction loss along with a multi-scale STFT discriminator. The exact parameters can be found in the Audicraft repository.⁷ While previous neural codecs rely on such combinations of reconstruction and adversarial losses, we experiment with pure adversarial training, where we only keep the feature loss and discriminator loss. We note that this was previously experimented in the context of bandwidth extension by [Tagliasacchi et al. \(2020\)](#) and [Hauret et al. \(2023\)](#). While removing reconstruction losses majorly degrades objective metrics, we observed during development that the resulting audio sounded much better than expected based on aforementioned metrics. Subjective evaluations reported in [Table 4](#) confirm this observation and demonstrate a remarkable boost in audio quality from training with adversarial losses only.

7. <https://github.com/facebookresearch/audiocraft/blob/main/config/solver/compression/default.yaml>

3.3.2 LEARNING SEMANTIC-ACOUSTIC TOKENS WITH A SPLIT RVQ

Similarly to SpeechTokenizer (Zhang et al., 2024b), we distill semantic information from a self-supervised model (WavLM⁸ (Chen et al., 2022) in our case) into the first level of the RVQ. WavLM projects a 16kHz waveform into 1024-dimensional embeddings sampled at 50Hz, while Mimi projects a 24kHz waveform into 512-dimensional at 12.5Hz. During training, we thus produce targets for distillation by downsampling the input waveform to 16kHz before computing WavLM embeddings followed by average pooling with a stride of 4 and a kernel size of 8, to reach 12.5 Hz. Interestingly, we observed that it was critical for performance to perform this average pooling in a non-causal way, which is compatible with streaming inference as these embeddings are only used during training. We apply a linear projection with an output dimension of 1024 to the output of the first RVQ level, parallel to the actual embedding going into the decoder. We then compute a cosine distance between the output of the first quantizer and the transformed WavLM embeddings, to perform distillation. Table 3 shows that this distillation loss conflicts with reconstruction and adversarial losses targeting quality. Indeed, while distillation significantly improves the phonetic discriminability of the first quantizer (as measured by ABX (Schatz et al., 2013)), it also affects audio quality negatively. We hypothesize that this is due to distilling semantic information into the first level of a single RVQ: As higher-order quantizers operate on the residual of the first one, the latter needs to trade audio quality for phonetic discriminability. We address this issue by proposing a split RVQ. Rather than a single RVQ with 8 levels, we distill semantic information into a plain VQ and apply an RVQ with 7 levels in parallel. We sum their outputs, such that while both can be used for reconstruction, we remove the constraint that acoustic information should be conserved in the residual of the semantic quantizer. Figure 2 illustrates this architecture and Table 3 shows that this solution provides a better semantic-acoustic trade-off overall.

3.4 Generative Audio Modeling

We now describe how we extend the base Helium model to support the modeling of the audio tokens provided by the Mimi codec. With our goal of achieving realistic spoken dialogue interactions, we further show how to model not just a single stream of audio, but two at the same time, one representing the user, and one the system. Finally, we detail a novel feature, the *Inner Monologue*, which consists in a joint modeling of the textual and audio modalities on the system side, to improve the quality of interactions.

3.4.1 HIERARCHICAL AUTOREGRESSIVE MODELING WITH RQ-TRANSFORMER

Let $U \in \{1, \dots, N\}^S$ be a discrete random sequence, with cardinality N and a sequence length S . For convenience, we also denote $U_0 = 0$, a deterministic initial token value. Autoregressive modeling consists in estimating the joint distribution $\mathbb{P}[U_1, \dots, U_S]$ through estimating the conditional distributions $\mathbb{P}[U_s | U_0, \dots, U_{s-1}]$ for all steps $1 \leq s \leq S$. Text language models, such as GPT (Radford et al., 2019) or Helium, fit this paradigm.

When modeling spoken language, relying on the tokenized text yields a much more compact representation than audio tokens: Using the Mimi codec introduced in Section 3.3,

8. <https://huggingface.co/microsoft/wavlm-large>

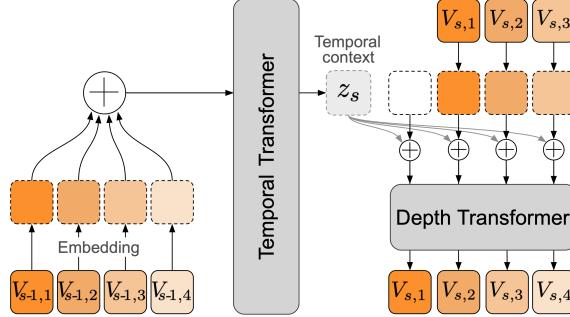


Figure 3: Architecture of the RQ-Transformer. The RQ-Transformer breaks down a flattened sequence of length $K \cdot S$ into S timesteps for a large Temporal Transformer which produces a context embedding used to condition a smaller Depth Transformer over K steps. This allows scaling to longer sequences 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.

with $Q = 8$ codebooks at a frame rate of 12.5hz, one would require a sequence length of 100 steps per second of audio to generate. To model 5 minutes of audio, this would amount to 30,000 timesteps, which represents a significant computational cost and generating 100 tokens per second is incompatible with streaming inference. As a comparison, a sample of English speech can be represented with around 3 to 4 text tokens per second.

We are interested in modeling not just a single sequence (U_s), but multiple sub-sequences, e.g. different audio codebooks, along with an optional text stream. We can stack those sub-sequences as $V_{s,k}$ for $1 \leq s \leq S$ and $1 \leq k \leq K$. Similarly, we define $V_{0,k} = 0$, a deterministic initial token value for all sub-sequences. For each $1 \leq s \leq S$ and $1 \leq k \leq K$, $V_{s,k} \in \{1, \dots, N_k\}$, where N_k is the cardinality of the k -th sub-sequence. One can flatten the K sequences into a single one, increasing the number of predictions by K . Lee et al. (2022) propose using a smaller autoregressive model along the dimension K , combined with a larger model along the time dimension, forming a RQ-Transformer. Later, Yu et al. (2024) suggested a similar approach for byte-level modeling while Yang et al. (2023) and Zhu et al. (2024) applied it to audio token modeling.

RQ-Transformer. Formally, the RQ-Transformer consists in two Transformer models, as illustrated in Figure 3. It consists of a *Temporal* Transformer, e.g. with the same architecture as the one described for Helium in Section 3.2, and a smaller *Depth* Transformer. We denote Tr_{Temp} the function represented by the Temporal Transformer, and Tr_{Depth} the one for the Depth Transformer. For simplicity, and for all steps $s \leq S$, we denote $V_s = (V_{s,1}, \dots, V_{s,K})$ the joint value of all sub-sequences at step s . For a given sequence step $1 \leq s \leq S$, the Temporal Transformer maps (V_0, \dots, V_{s-1}) to a *temporal context* vector

$$z_s = \text{Tr}_{\text{Temp}}(V_0, \dots, V_{s-1}) \in \mathbb{R}^d. \quad (1)$$

If we further take a sub-sequence index $1 < k \leq K$, the Depth Transformer maps both z_s along with $(V_{s,1}, \dots, V_{s,k-1})$ to the logits estimate

$$l_{s,k} = \text{Tr}_{\text{Depth}}(z_s, V_{s,1}, \dots, V_{s,k-1}) \in \mathbb{R}^{N_k}. \quad (2)$$

We further define $l_{s,1} = \text{Lin}(z_s) \in \mathbb{R}^{N_1}$, with Lin a dedicated linear layer. We train Tr_{Temp} , Tr_{Depth} and Lin so that $\text{softmax}(l_{s,k})$ is a good approximation of the distribution of $V_{s,k}$ conditioned on all sub-sequences for the previous steps, and of the previous sub-sequences for the current step, e.g.

$$\begin{cases} \text{softmax}(l_{s,1}) & \approx \mathbb{P}[V_{s,1}|V_0, \dots, V_{s-1}] \\ \text{softmax}(l_{s,k}) & \approx \mathbb{P}[V_{s,k}|V_0, \dots, V_{s-1}, V_{s,1}, \dots, V_{s,k-1}] \quad \text{if } k > 1. \end{cases} \quad (3)$$

Importantly, the number of steps in the Temporal Transformer is always equal to S , rather than $K \cdot S$, and the number of steps in the Depth Transformer is at most K . In practice, the Temporal Transformer receives at each step s as input the sum of K learnt embedding tables representing the value for the last V_{s-1} . Given $1 < k \leq K$, the Depth Transformer receives as input the sum of z_s and of a learnt embedding representing $V_{s,k-1}$.

As detailed in [Table 1](#), our Depth Transformer has 6 layers, a dimension of 1024, and 16 attention heads. Unlike [Lee et al. \(2022\)](#); [Yang et al. \(2023\)](#); [Zhu et al. \(2024\)](#), we use different parameters per index k for the linear layers, projection and fully connected, in the Depth Transformer. Indeed, different sub-sequences might require different transformations. Given the smaller size of this Transformer, this has no impact on both training and inference time, while [Table 6](#) shows that this depthwise parametrization is beneficial.

3.4.2 AUDIO MODELING

The audio codec Mimi described in [Section 3.3](#) outputs Q sub-sequences, with 12.5 steps per second of audio. We denote those sequences by $A_{t,q} \in \{1, \dots, N_A\}$ for $1 \leq t \leq T$ with $T = 12.5 \cdot \text{duration}$, and $1 \leq q \leq Q$ with $Q = 8$. We insert the audio sub-sequences into the multi-sequence V modeled by the RQ-Transformer. Remember that the first codebook $A_{t,1}$ corresponds to the semantic information, as detailed in [Section 3.3.2](#), while the other codebooks correspond to acoustic features.

Acoustic delay. We first experimented with simply setting $V = A$ in the modeling. However we find that introducing a slight delay between the semantic and acoustic tokens led to more stable generations. [Copet et al. \(2023\)](#) show that this leads to reduced dependencies between the sub-sequences for a given time step, conditioned on the past, thus allowing to use a weaker model to approximate the joint distribution $\mathbb{P}[V_{s,k}|V_0, \dots, V_{s-1}]$ (in their case, as the product of the conditioned marginals). [Lemercier et al. \(2024\)](#) further show a connection between the mutual information between the sub-sequences at a given step, and the quality of the generation: naturally, the more complex the interdependence, the more powerful a model will be needed to estimate them.

As shown in [Section 5.3](#), introducing a delay of 1 or 2 steps between the semantic and acoustic features greatly improves the quality of the generation. This allows the Temporal, larger, Transformer to model the inter-dependence between semantic and acoustic features. Formally, given a delay $\tau \in \mathbb{N}$, we have, for all steps s

$$\begin{cases} V_{s,1} = A_{s,1} \\ V_{s,q} = A_{s-\tau,q} & \text{if } s \geq \tau + 1, q > 1 \\ V_{s,q} = 0 & \text{if } s < \tau + 1, q > 1. \end{cases} \quad (4)$$

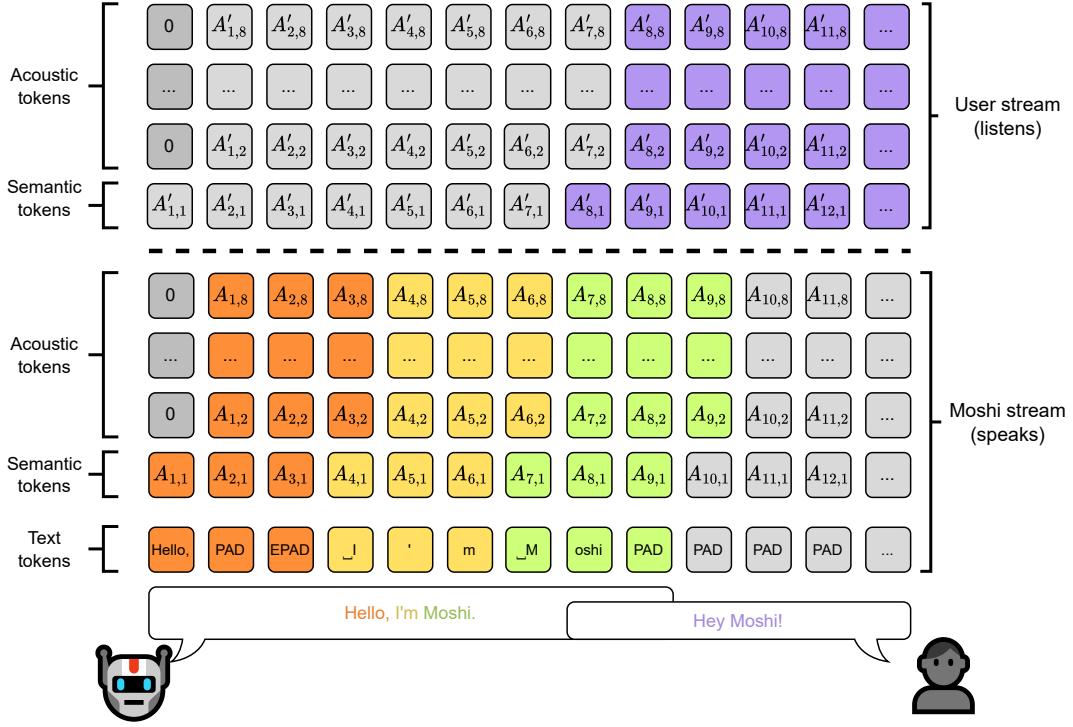


Figure 4: **Representation of the joint sequence modeled by Moshi.** Each column represents the tokens for a given step in the joint sequence ($V_{s,k}$) described in Equation 6 with an acoustic delay $\tau = 1$, e.g. the input of the Temporal Transformer for this step. Tokens are predicted from bottom 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.

Note that using RQ-Transformers to model audio was successfully used by Yang et al. (2023) and Zhu et al. (2024). We introduce here the use of per-codebook parameters in the Depth Transformer, and the use of the acoustic delay. Compared with (Zhu et al., 2024) which first generates all the semantic tokens, we generate them jointly with the acoustic tokens, which allows for the first time a streaming modeling of semantic and acoustic tokens jointly.

3.4.3 MULTI-STREAM MODELING

Modeling a single stream of audio is not sufficient to fully model a conversation. Our framework can be extended to modeling a two-speaker conversation: given two streams of audios ($A_{t,q}$) and ($A'_{t,q}$), we simply apply the acoustic delay to both, and concatenate them into V , extending Equation 4. In practice, A will correspond to Moshi, while A' models the user.

3.4.4 INNER MONOLOGUE

While operating purely in the audio domain already yields convincing results (see Table 7), we observe that having Moshi also model the textual representation of its own speech is providing a scaffolding that increases the linguistic quality of its generation. Formally, we

define a text stream $W \in \{1, \dots, N_W\}^T$ derived from the sequence of text tokens obtained by applying the SentencePiece tokenizer (Kudo and Richardson, 2018) to the transcription of the audio corresponding to Moshi with Whisper (Radford et al., 2023), as described after. We insert W as the first sub-sequence in V , such that it acts as a prefix to the generation of semantic tokens. This can be seen as an extension of the hierarchical semantic-to-acoustic generation introduced by Borsos et al. (2022). Note that we do not use the textual representation corresponding to the stream of the user, as transcribing this flux in real time would be challenging, and relying on an external ASR system contradicts our end-to-end speech-to-speech approach. Ablation studies in Section 5.3 show that among the design choices made for Moshi, Inner Monologue has one of the most critical impacts on the quality of generated speech.

Aligning text and audio tokens. To integrate text tokens with audio tokens that operate at a constant framerate of 12.5Hz, we need to align them to this framerate. For that, we leverage the word-level timestamp provided by Whisper. The i -th word in the transcript is mapped to $n_i \in \mathbb{N}^*$ text tokens $w_{i,j}$, $j \leq n_i$, along with a start index $t_i \in \{1, \dots, T\}$, simply defined as its start timestamp divided by the framerate of 12.5 Hz. We define two special tokens: PAD and EPAD, that never appear in any of the word tokens. We build W such that when a word starts, (W_t) contains its text tokens, followed by PAD until the next word. EPAD is inserted before the next word to indicate the end of the padding. While not strictly necessary, we observed this provided a useful guidance to the model by splitting the decision of ending a word, and which one should follow, into two steps.

First, the sequence (W_t) is initialized with PAD tokens, e.g. $\forall t, W_t \leftarrow \text{PAD}$. Then, we modify it iteratively as follows. For each word i and its start index t_i , we update W as

$$\begin{cases} W_{t_i-1} & \leftarrow \text{EPAD} \\ W_{t_i+j} & \leftarrow w_{i,j} \quad \forall j \leq n_i. \end{cases} \quad (5)$$

Note that if $t_i = 1$, we instead insert EPAD at index 1, and shift the text tokens. We do not insert an EPAD token if it would overwrite a text token from a previous word. As text tokens are more compact than the corresponding audio tokens, there is usually no overlap between words in W_t . In English conversational speech, we observe that padding tokens represent about 65% of the tokens.

Deriving streaming ASR and TTS. One can further introduce some delay between the text sequence (W_t) , and the audio tokens $(A_{t,q})$. This controls in which modality the language model will take the decision about the content of the generated audio. By setting the audio ahead of the text, the content of the text will be dictated by what audio has been sampled in the previous steps. In particular, by sampling only the text tokens, while using the ground truth audio tokens and discarding the prediction of the model for them, one obtain a streaming Automatic Speech Recognition model, which also provides precise word level alignment. On the other hand, by changing the text delay so that the text is ahead of the audio tokens, the content of the audio is dictated by the text content. Once more, given a sequence of properly padded text tokens, one obtain a streaming Text-To-Speech model. We further describe in Appendix C how one can adapt the inference of a language model with delayed text to obtain a zero-shot properly padded text tokens sequence. Experiments

in [Section 5.7](#) show how a single delay hyper-parameter allows for switching from an ASR to a TTS model with no changes in the loss, architecture, or training data.

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

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

amounting to a total number of $K = 2Q + 1$ streams, with $Q = 8$ in the experiments. A summary is provided in [Figure 4](#).

Inference of Moshi. The joint sequence given by [Equation 6](#) is the target for our modeling task at train time: At any time step s , the model is input with $0, V_1, \dots, V_{s-1}$ and output an estimated probability distribution $\hat{V}_s(0, V_1, \dots, V_{s-1})$. At inference time, we sample from $\hat{V}_{s,k}$ for all the sub-sequence indexes that corresponds to outputs of Moshi: i.e., for $k = 1$ for the text tokens corresponding to Moshi’s speech, and for $k \in \{2, \dots, 2+Q\}$ 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. However, modeling the user stream as output allows generating simulated dialogues, which is necessary for offline evaluation as in [Section 5.6](#). Interestingly, there is no explicit boundaries for the change of turns between the user and Moshi: Moshi can speak and listen at all time, and do both at once if needed. In particular, when the user speaks and Moshi stays silent, the corresponding audio tokens for Moshi’s stream decode into “natural silence”, a near silent waveform, instead of having a fixed, well defined value; At the same time, Moshi’s text stream will be filled with PAD tokens. As a result, the text stream can provide interesting ways of controlling Moshi, for instance, forcing the sampling of a EPAD token will make Moshi start talking immediately.

4 Datasets and Training

4.1 Text Data

Our training dataset is made of a mix of high-quality data sources and filtered web data from CommonCrawl. More specifically, 12.5% of our dataset is from the following curated sources: Wikipedia,⁹ Wikibooks, Wikisource, Wikinews, StackExchange¹⁰ and the collection of scientific articles pes2o.¹¹ Instead of doing multiple passes on Wikipedia, we use five different dumps from 2017, 2018, 2019, 2021 and 2022. The remaining 87.5% of our dataset is from CommonCrawl, and was filtered with the pipeline described in [Section 3.2.2](#). We used the following ten crawls: 2018-30, 2019-04, 2019-30, 2020-05, 2020-34, 2021-04, 2021-31, 2022-05, 2022-33, 2023-40.

9. <https://dumps.wikimedia.org/>

10. <https://archive.org/details/stackexchange>

11. <https://github.com/allenai/pes2o>

4.2 Audio Data

We use an audio collection of 7 million hours, which we call the *unsupervised audio dataset*, of readily available audio content, the majority of which contains English speech. We transcribe this set with Whisper ([Radford et al., 2023](#)), using the large-v3 model. We use this data for the audio pre-training phase, during which we do not use the multi-stream approach described in [Section 3.4.3](#), but instead use a single stream of audio representing all speakers at once. Similarly, the text stream described in [Section 3.4.4](#) represents the words coming from all speakers. All the audio is resampled to 24kHz and downmixed to mono.

To achieve multi-stream, we need the model to gain the ability to both listen and speak at the same time. For this, we further leverage the Fisher dataset ([Cieri et al., 2004](#)). It consists of 2000 hours of phone conversations between randomly paired participants, with a given topic to discuss. A property of Fisher is that each conversation side is recorded on a separate channels, which allows providing ground-truth separated streams to Moshi. The original audio is sampled at 8kHz, and we use AudioSR ([Liu et al., 2023a](#)) to upsample it to 24kHz.

Finally, we source 170 hours of natural and scripted conversations between multiple pairs of participants, recorded with separate channels per speaker, in order to provide a small dataset on which to finetune the model to improve the quality over the one obtained when using only Fisher. We call this dataset the *supervised multi-stream dataset*. We do not train Moshi directly on this dataset, but use it to train a realistic multi-stream TTS model, and fine-tune Helium on real conversation transcripts as explained in Sections [4.3](#) and [4.4](#).

For both Fisher and this last dataset, we sample one speaker randomly as the main speaker (i.e., Moshi speaking), and put the other speaker on the second audio stream. For Fisher, the text stream only contains the transcription of the main speaker. To obtain reliable timestamps, despite long silences in each stream, we use transcription obtained with the `whisper-timestamped` package ([Louradour, 2023](#)), along with the medium Whisper model.

4.3 Speech-Text Instruct Data

Early experiments using text-based instruct datasets such as Open Hermes ([Teknium, 2023](#)) proved to be ill-suited for the instruct tuning of a spoken conversational system. In particular, the data formatting was often impossible to properly render with TTS (e.g. URLs), and the format of the questions and responses was not following a natural oral flow (e.g. bullet points, long enumerations). Instead, we leverage Helium, fine-tuned on Open Hermes and transcripts of real conversations, to generate realistic interactions between a speech-based AI model and a user. We then synthesize them with our multi-stream streaming TTS described in [Appendix C](#), leading to more than 20k hours of synthetic speech data. To give Moshi its own consistent voice, we also condition the TTS engine on the voice of a single actor, who recorded monologues covering more than 70 speaking styles, as listed in [Table 19](#). Experiments on voice consistency reported in [Section 6.3](#) show that simply using a consistent voice for Moshi during instruction tuning is enough to guarantee almost surely that it does not use another voice, without further control during inference. In contrast, the voice of the second audio stream (the user) is randomly sampled for each example, giving more robustness to different speaking conditions and accents.

To generate the transcripts, we use different prompts, aiming at capturing different kinds of interactions between a user and Moshi. First, we generate conversations about general

knowledge, by starting from a few Wikipedia paragraphs or StackExchange posts, which we refer to as context. This ensures that Moshi’s conversations cover a wide range of topics, such as history, cooking advice or pop culture. More precisely, using a given context, we obtain a summary of a potential discussion with the following prompt:

```
{context}
```

Based on information from the previous paragraph, write the summary of a conversation about `{title}` between Blake and Moshi. The summary must be 2 sentences long, and start with "They" or "The speakers".

where `{context}` refers to paragraphs from Wikipedia or StackExchange and `{title}` is the corresponding title. Then, we generate the full transcript with the prompt:

```
{context}
```

Write the transcript of a conversation between Blake and Moshi.
`{summary}` Moshi is knowledgeable about the topic. Use some backchanneling. Use short turns.

Similarly, to give Moshi information about itself and the Kyutai lab, we generate paragraphs describing both and use them as additional context.

Second, we produce interactions containing instructions about Moshi’s voice, such as the other speaker requesting Moshi to speak with an angry voice or like a pirate. Our first strategy is to generate single turn interactions where the model is instructed to tell a sentence, a monologue or a poem about an entity, belonging to a high level category such as “sports” or “animals”, using a particular voice. The voice requested by the other speaker and the entity are randomly sampled, and are thus completely unrelated. Our second strategy is to generate roleplaying situations, corresponding to different emotions or speaking styles with the following prompt:

Write a list of 10 situations about a `{voice}` `{character}`. Each situation must start with "a `{voice}` `{character}` who" and must be at most 8 words long.

Examples of voice adjective include “happy” or “surprised” and examples of characters include “detective” or “superhero”. We then generate the interaction using the prompt:

Write a dialogue between Blake and Moshi, `{situation}`. Use a lot of backchanneling.

To make Moshi robust to mispronounced words, we also generate instructions containing misspellings in the user’s questions, followed by Moshi asking the user to repeat herself or to

clarify the question. We also generate questions containing a false or misleading fact (such as “Is the Eiffel Tower in Beijing?”), to train the model to answer “No” and correct the user. Otherwise, the vast majority of generated conversations only contain questions from the user where Moshi should answer positively. We generate basic math, grammar or trivia single-turn questions and answers, as we noticed that Moshi was initially not performing well on simple factual tasks like adding numbers. Finally, we generate safety conversations, where the user asks unethical or NSFW questions, and Moshi refuses to answer these requests.

4.4 Training Stages and Hyper-parameters

Helium pre-training. An overview of the training stages and hyper-parameters is provided in [Table 1](#). For each stage, we use AdamW ([Loshchilov and Hutter, 2019](#)), with a weight decay of 0.1, a momentum decay of 0.9, and a decay for the average of the squared gradient of 0.95. All models are trained on H100 GPUs, using FSDP and activation checkpointing. The text-only language model, Helium, is trained for 500k steps, with a batch size of 4.2M tokens, using a cosine learning rate schedule starting at $3 \cdot 10^{-4}$ with linear warmup.

Moshi pre-training. Then, we initialize the Temporal Transformer in Moshi with Helium, while the Depth Transformer described in [Section 3.4.1](#) is randomly initialized. We first train on the unsupervised audio dataset presented in [Section 4.2](#), using a single stream of audio, with a batch size covering 16 hours of audio, each batch item consisting of a 5 mn sequence. We mask the corresponding text tokens with a probability of 30%. We randomize the delay between the text and audio tokens between -0.6 and +0.6 seconds. In order to prevent catastrophic forgetting, we also train half of the time on batches of text only data from the same dataset as used for Helium. In total, we make 1 million training steps, with a cosine learning rate starting at $3 \cdot 10^{-5}$ for the Temporal Transformer, and $2 \cdot 10^{-4}$ for the Depth Transformer, also with a linear warmup. In order to ensure the updates from the text-only batches are balanced with those from the audio dataset, we use two separate optimizer states. In addition, when operating on the text stream from an audio batch, we multiply the learning rate for the text embedding and text linear layer by 0.75. Finally, as padding tokens are predominant for audio batches, we reduce their weight by 50% in the cross-entropy loss.

Moshi post-training. Starting from the model obtained from the previous stage, we then train it to gain its multi-stream ability. First, we use PyAnnote ([Bredin, 2023](#)) to diarize the audio from the unsupervised audio dataset. We sample one speaker at random, which will act as the main speaker, and derive a binary mask over the waveform, with a value of 1 when the speaker is active based on the diarization, and 0 otherwise. This mask provides us with two waveforms: one with the speaker, and one with the residual (potentially several speakers), which are encoded separately and then used as the two input audio streams described in [Section 3.4.3](#). The text stream only contains the text tokens from the selected main speaker, and the delay between text and audio tokens is fixed to 0. We train for 100k steps, with a batch size of 8 hours of audio, and a fixed learning rate of $3 \cdot 10^{-6}$ for the Temporal Transformer, and $5 \cdot 10^{-5}$ for the Depth Transformer. Like for the pretraining phase, we sample full text-only batches 10% of the time.

Moshi finetuning. The previously described simulated multi-stream provides a good pre-training task but is far from being sufficient to capture natural conversations: For instance,

it contains no overlap, and the stream of an inactive speaker is perfectly silent. We then use the Fisher dataset (Cieri et al., 2004) to have the model learn real multi-stream interaction. We sample one of the two speakers to be the first (and main) speaker. We train for 10k batches, with a batch size of 40min of audio with a learning rate of $2 \cdot 10^{-6}/4 \cdot 10^{-6}$ for the main/Depth Transformer. We no longer sample full text batches.

Finally, we set the speaker identity for the first speaker stream to be that of Moshi, a useful conversational assistant, with a final stage of instruct finetuning. We use the synthetic instruct dataset described in Section 4.3, with a batch size of 2.7 hours of audio, for 30k steps, with a learning rate of $2 \cdot 10^{-6}$ for both transformers.

During this stage, we perform data augmentation on the user’s stream to make Moshi robust to various situations. Namely, we apply a random gain to the user stream between -24 dB and +15 dB, 50% of the time. 30% of the time, we further add noise extracts from the Deep Noise Suppression challenge (Dubey et al., 2023) which we concatenate in order to cover the entire duration of each example. The noise is amplified to reach a target volume relative to the original source between -30 dB and +6 dB. Each time we need to sample a new noise, we alternatively use a silent section with a random duration up to 30 seconds of silence with probability of 50%, so that the model can handle the audio condition going from noisy to silent, and vice versa. We emulate echo from Moshi into the user’s microphone by adding a scaled down copy of Moshi’s stream into the user’s stream, scaled by a factor uniformly sampled in [0, 0.2], and a delay uniformly sampled between [100ms, 500ms]. Finally, we apply to the user’s stream, potentially augmented with the echo, a reverb-like augmentation as introduced by Defossez et al. (2020). The echo and reverb are applied together with a probability of 30%.

TTS Training. We also train a streaming, multi-stream text-to-speech model, using the method described in Section 3.4.4. The audio pre-training stage is shared with Moshi, while the post-training is completed using a delay of 2 seconds for the audio stream compared to the text. The model is finetuned on the supervised multi-stream dataset containing high quality recording of interactions between two speakers. It is used to generate the synthetic finetuning instruct dataset described in Section 4.3. Note that Moshi itself is not trained on the supervised multi-stream dataset. Further details are provided in Appendix C.

Training loss. Moshi is trained to model joint sequences, as presented in eq. 6. Given the ground-truth discrete token $(V_{s,k})_{s \leq S, k \leq K}$, and the estimated logits $(l_{s,k})_{s \leq S, k \leq K}$ from eq. 2, we use the following loss, with CE the cross entropy,

$$L(V, l) = \frac{1}{S} \sum_{s=1}^S \left(\text{CE}(l_{s,1}, V_{s,1}) + \frac{1}{\sum_{k=2}^K \alpha_k} \sum_{k=2}^K \alpha_k \text{CE}(l_{s,k}, V_{s,k}) \right). \quad (7)$$

Thus, we give the same importance to the text token ($k=1$), and the combined audio tokens. α_k is set to 100 for semantic tokens, and 1 for acoustic ones.

Table 2: **Text language model evaluation.** Performance on standard benchmarks for evaluating large language models, including closed book question answering, reasoning and multiple choice QA exams. We report in bold the best performing model trained on less than 2.5T tokens.

	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

5 Evaluation

5.1 Text Language Modeling

Metrics. We evaluate Helium (trained only on text data) on the following standard benchmarks: AI2 Reasoning Challenge (Clark et al., 2018, ARC), Open-Book QA (Mihaylov et al., 2018, OBQA), HellaSwag (Zellers et al., 2019, HS), WinoGrande (Sakaguchi et al., 2021, WG), Physical Interaction QA (Bisk et al., 2020, PIQA), Social Interaction QA (Sap et al., 2019), TriviaQA (Joshi et al., 2017, TQA), Natural Questions (Kwiatkowski et al., 2019, NQ) and Massive Multitask Language Understanding benchmark (Hendrycks et al., 2020, MMLU). These benchmarks cover a wide variety of tasks, including common sense reasoning, closed-book question answering or multiple choice question answering from high school and college subjects. We follow the evaluation protocol from previous work such as GPT-3 or Llama: we perform 5-shot evaluation on TriviaQA, NQ and MMLU, and 0-shot evaluation on the other datasets. On TriviaQA, we report performance on the Unfiltered and Wikipedia splits.

Baselines. As baselines, we consider existing large language models with a size around 7B parameters, and which are trained using roughly the same amount of compute as Helium. More specifically, we include models that are trained on fewer than 2.5T tokens (compared to the 2.1T tokens that are used to train Helium), namely MPT (Team, 2023), Falcon (Almazrouei et al., 2023), Llama 2 (Touvron et al., 2023b) and OLMo (Groeneveld et al., 2024). We also include Mistral and Gemma, two popular open weights models that are trained using significantly more compute than Helium.

Results. We report results in Table 2, and we observe that on most benchmarks, Helium is on-par or outperforming models using similar amount of training compute. Even compared to Mistral and Gemma, which use up to 3x more compute for training, Helium obtains competitive results on some benchmarks such as ARC, Open-Book QA or Natural Questions. This validates the quality of our pre-training text data.

5.2 Audio Tokenization

Metrics. We then evaluate the semantic and acoustic performance of our neural codec, Mimi. First, we evaluate whether the semantic tokens it produces provide targets that are amenable to language modeling. To do so, we compute a triphone-based ABX (Schatz et al., 2013) error rate that characterizes the phonetic discriminability of a representation space by comparing distances between two embeddings of different instances of a same triphone (e.g. “beg”) and a negative triphone that differs minimally (e.g. “bag”). More precisely, we compute a “within speaker” ABX where the three instances are pronounced by the same speaker, and report error rates on Librispeech (Panayotov et al., 2015) dev-clean with the default parameters of the Librilight (Kahn et al., 2020) repository¹². The resulting score has been shown to be a strong predictor of the ability of a downstream audio language model to produce coherent speech (Lakhotia et al., 2021). Since we are interested in characterizing only the semantic token, we compute distances in the latent space produced after quantization with the semantic VQ only (i.e. before summing with acoustic tokens). Second, we evaluate the acoustic quality of reconstructed audio. As objective, automatic metrics we rely on VisQOL (Hines et al., 2015)—a full-reference model of acoustic similarity—and MOSNet (Lo et al., 2019)—a reference-free model of audio quality. Given the limitations of automatic evaluation of audio quality, we also perform human evaluations with a MUSHRA protocol. We rely on judgments of 20 listeners, each one rating 30 samples of 10s each. Table 3 reports ablations studies using objective metrics, while Table 4 provides a comparison with previous work both in terms of objective and subjective evaluation.

Baselines. We compare against RVQGAN (Kumar et al., 2024), SemantiCodec (Liu et al., 2024), and SpeechTokenizer (Zhang et al., 2024b). RVQGAN is a pure acoustic tokenizer, in the sense that it does not encode semantic information. Thus, we only evaluate it in terms of audio quality. RVQGAN produces tokens at 75Hz, so we only keep the first two levels of RVQ to obtain a bitrate of 1.5kbps, closer to that of Mimi. On the other hand, SpeechTokenizer relies on distillation to encode semantic information into its first token such that we can evaluate both its semantic and acoustic properties. We keep its first 3 RVQ levels to obtain a 1.5kbps bitrate. Similarly, SemantiCodec also encodes semantic and acoustic information such that it can be evaluated along both axes.

Results - Semantic tokens. Table 3 shows that Mimi’s phonetic discriminability of semantic tokens, as measured by ABX, is poor in the absence of distillation and comparable to acoustic tokens of previous work (Borsos et al., 2022): This means these semantic tokens are not amenable to capturing linguistic content from speech. In contrast, distilling WavLM into the semantic tokens significantly improves their phonetic discriminability, in particular when using a Transformer in Mimi’s encoder. This can be explained by the fact that distilling a large Transformer based encoder into a purely convolutional one is challenging, while increasing the capacity and receptive field of the encoder helps. Yet, we observe a conflict between acoustic losses and semantic distillation, as improving ABX implies reducing reconstruction quality (as measured by MUSHRA). Using a split RVQ as described in Section 3.3.2 improves the trade-off between semantic properties and audio quality, improving MUSHRA from 57.8 to 64.0 while moderately degrading ABX from 6.5% to 8.1%.

12. <https://github.com/facebookresearch/libri-light/blob/main/eval/README.md>

Table 3: Ablation study on hyper-parameters of the Mimi codec. We evaluate semantic modeling by reporting the error rate on a phonetic ABX discriminability task. To evaluate the reconstruction quality, we compute VisQOL and MOSNet and collect human judgments with a MUSHRA protocol. “Quantization rate” refers to applying quantization to the latent space only 50% of the time during training (independently from quantizer dropout), as described in Section 3.3.

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

Table 4: Audio quality evaluation. Objective and subjective (MUSHRA) evaluation of audio quality for baseline neural audio codecs—RVQGAN (Kumar et al., 2024), SemantiCodec (Liu et al., 2024), and SpeechTokenizer (Zhang et al., 2024b)—and the most important variants of Mimi. For a fair comparison with SemantiCodec and SpeechTokenizer, we also include a downsampled version of our codec in the MUSHRA study. f_s is the audio sample rate and f_r the codec frame rate. Both Mimi codecs are trained with distillation, and either with the same combination of reconstruction and adversarial losses as Encoded (see Section 3.3) or adversarial losses only.

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

Results - Acoustic tokens. Table 3 also shows a significant improvement in MUSHRA when adding a Transformer in the decoder. Similarly, using a quantization rate of 50% significantly improves VisQOL. Quantization rate however does not improve perceived quality. More generally, we observe a poor correlation between VisQOL and MOSNet. In particular, Table 4 shows that training Mimi with adversarial losses only leads to a very low VisQOL of 1.84 which does not account for the high perceived audio quality. We thus rely on MUSHRA where raters are asked to judge the similarity of a reconstructed audio to its ground-truth anchor, with a score between 0 and 100. This human evaluation shows a significant improvement from using adversarial losses only, with a MUSHRA score of 81.0 against 58.8 when using the mix of loss functions used in Encoded. Mimi moreover significantly outperforms RVQGAN (Kumar et al., 2023) despite operating at a lower bitrate and modeling semantic information. Mimi also provides higher reconstruction quality than SemantiCodec (Liu et al., 2024) while operating at a 4× lower framerate. This property is crucial to achieve

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-1T¹³ 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

the low latency of Moshi, since generating one temporal frame of audio tokens with Moshi requires a full forward pass through the Temporal Transformer. Finally, both RVQGAN and SemantiCodec are non-causal, while Mimi is fully causal and thus compatible with streaming inference and modeling of real-time conversations.

Discussion. Mimi overall provides high reconstruction quality while encoding semantic information, being fully causal, and operating at low framerate and bitrate. In consequence, Mimi proves to be a well-fitted audio tokenizer to train real-time audio language models. A collateral finding of our study is a concerning lack of correlation between objective and subjective audio quality metrics. In particular, while we find VisQOL to provide a reliable proxy for perceived quality when modifying the generator architecture, changing the training objective (e.g. removing reconstruction losses) moves the score in directions that are completely decorrelated from human perception. This observation underscores the open challenge of designing reliable objective proxies for perceived quality.

5.3 Ablations on Generative Modeling

Metrics. We perform ablation studies to evaluate the impact of using an RQ-Transformer, as well as comparing delay patterns and weighting of various token levels. All models are initialized with Helium for the Temporal Transformer and pretrained on audio data. When comparing models with a same delay pattern and without Inner Monologue, we rely on perplexity, averaged over semantic and acoustic tokens. However, perplexities are not comparable between models with various delays, as they do not model the same conditional distributions. To compare various delay patterns, we measure the ability of Moshi to generate intelligible, consistent speech by transcribing generations (conditioned on a 3s prompt) with Whisper (Radford et al., 2023) and scoring the resulting transcripts with an external text language model. We rely on a lightweight text model—LiteLlama-460M-1T¹⁴—as it is more practical for continuous evaluation along training. We also report the length of the transcripts (in characters), as we find it to be a strong predictor of model quality (weak models typically collapse to silence).

Results - RQ-Transformer. Table 5 reports results for ablations on the use of an RQ-Transformer. We first replicate the setting of Copet et al. (2023) with the delay pattern of $[0, 1, 2, 3, 4, 5, 6, 7]$, which means that each level of RVQ token is generated one timestep after the preceding level. In this context, we see that using an RQ-Transformer is not necessary, as it only provides a marginal improvement in perplexity. However, this delay pattern induces a theoretical latency of 8 timesteps, which amounts to 640ms, a latency that is incompatible with the requirements of a real-time dialogue model. We thus switch to a reduced latency of 240ms with the pattern $[0, 2, 2, 2, 2, 2, 2, 2]$. In that context, modeling RVQ tokens with an RQ-Transformer significantly improves perplexity over using separate classification heads. Thus, the RQ-Transformer becomes a critical component of generative models of RVQ tokens under strict latency constraints.

Results - Additional ablations. Table 6 reports additional ablations on additional delay patterns, the weight of the semantic token loss and our proposed Inner Monologue procedure, all using the RQ-Transformer. First, we compare three configurations of delays that are compatible with real-time dialogue. The $[0, 0, 0, 0, 0, 0, 0, 0]$ pattern represents the minimal latency of 80ms that can be obtained with Mimi tokens at 12.5Hz. Allowing an additional 80ms of latency with one step of delay significantly improves the quality of generated speech, while 240ms of latency brings further moderate improvement. In early experiments, we also observed that the individual losses per RVQ level were conflicting with one another, despite each level being more important in the final intelligibility and audio quality than the next one. We thus bring two changes to the architecture and training process. We first increase the weight of the loss on predicting the semantic tokens to 100, while keeping it at 1 for all other levels of the audio tokens. This gives another boost to speech intelligibility. We furthermore reduce competition between RVQ levels by using a depthwise parametrization, as described in Section 3.4.1, such that each RVQ level is predicted by its own set of weights in the Depth Transformer, rather than having shared weights across levels. Finally, the most drastic improvement to the quality and length of generated speech comes from enabling Inner Monologue.

14. <https://huggingface.co/ahxt/LiteLlama-460M-1T>

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), \emptyset represents unsupported modalities while - represents unreported numbers.

Model	Audio metrics				Text metrics
	sWUGGY	sBLIMP	sTopic-StoryCloze	sStoryCloze	
<i>Audio only - Cold Start</i>					
GSLM (Lakhota et al., 2021)	64.8	54.2	66.6	53.3	\emptyset
AudioLM (Borsos et al., 2022)	71.5	64.7	-	-	\emptyset
TWIST (Hassid et al., 2023)	72.2	56.5	-	-	\emptyset
Moshi	74.8	59.9	80.9	56.9	\emptyset
<i>Audio only - Warm Start</i>					
TWIST (Hassid et al., 2023)	74.5	59.2	76.4	55.4	\emptyset
VoxLM (Maiti et al., 2023)	62.9	53.9	-	-	\emptyset
Spirit-LM (Nguyen et al., 2024)	69.5	58.0	72.9	54.8	\emptyset
Moshi	74.3	58.9	81.8	58.7	\emptyset
<i>Text and audio - Warm Start</i>					
VoxLM (Maiti et al., 2023)	66.1	57.1	-	-	\emptyset
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

Discussion. Beyond the choice of architecture and delay patterns, these ablations show how helpful modeling text tokens along the audio tokens with Inner Monologue is, even in an audio-to-audio setting. Given the positive impact of depthwise parametrization and a weight of 100 for the semantic token, both are used in the subsequent experiments and our final training procedure. As described in Table 1, we pretrain Moshi with an acoustic delay of 2 and finetune it with an acoustic delay of 1, for a theoretical latency of 160ms.

5.4 Audio Language Modeling

Metrics. We first measure the ability of Moshi to model speech sequences when being trained for next token prediction on large scale audio data. To do so, we rely on “textless NLP” (Lakhota et al., 2021) metrics that evaluate an audio language model’s linguistic knowledge by comparing likelihoods of positive and negative speech examples represented as audio tokens. sWUGGY evaluates a model’s ability to learn a lexicon from speech by comparing the likelihood of an existing word and an invalid variant (e.g. “oxidation” and “accidation”), while sBLIMP evaluates syntactic contrasts. Spoken StoryCloze metrics introduced by Hassid et al. (2023) evaluate semantic contrasts by comparing commonsense five-sentence stories, with the last one being either coherent with the context or incoherent. Given the difficulty of this task in the audio domain, Hassid et al. (2023) also propose Spoken Topic-StoryCloze, a variant where the negative continuation is randomly sampled among unrelated sentences (rather than being subtly incoherent), resulting in higher scores. We score sequences with a negative-log likelihood normalized by the sequence length. Since our model produces several tokens per timestep, we sum all tokens of a timestep with the

weights used during training, i.e. 100 for the semantic tokens and 1 for the acoustic ones. We do not include text tokens from the Inner Monologue, as these metrics are designed to compare untranscribed audio sequences. Similarly, when evaluating a multi-stream model after instruction tuning, we only score the tokens from the user stream as it does not include text tokens. Finally, we also report text understanding evaluation (without audio tokens) on MMLU (Hendrycks et al., 2020) for Spirit-LM and Moshi, as a way to measure how much the audio training affects the textual knowledge of the original checkpoint.

Baselines. We compare against baselines from the audio language modeling literature, in three settings. The first category encompasses audio-only models starting from a random initialization, including GSLM (Lakhotia et al., 2021), AudioLM (Borsos et al., 2022) and TWIST-1.3B (Hassid et al., 2023). In this case, we report metrics for a single-stream Moshi initialized randomly and pretrained only on audio data and without Inner Monologue. The second category includes models that start from a pretrained text LM and are then only trained on audio. This includes TWIST-13B as well as the audio-only version of VoxTLM (first row of (Maiti et al., 2023, Table 3)) and that of Spirit-LM (reported as “Speech Only” in (Nguyen et al., 2024, Table 5)). The corresponding Moshi model is similar to the one mentioned above (audio-only data, no Inner Monologue) but starts from the pretrained Helium checkpoint. The last category is composed of actual multimodal models that are trained jointly on speech and text data. In this context we report results for three configurations of Moshi. First, we report results for Moshi pretrained on single-stream data. Then, we report results for the final model after multi-stream post-training and finetuning using real recordings from a voice actor to condition the creating of synthetic data in Moshi’s voice. The last model is identical to the previous one except for the fact that it uses a synthetic voice for Moshi. We remind the reader that even if these models are trained with Inner Monologue, they are evaluated without, to provide a fair comparison with baselines.

Results. Table 7 reports results on audio language modeling. In the “Audio only - Cold Start” setting, Moshi already provides a strong baseline, in particular considerably improving over previous work in sTopic-StoryCloze. When initialized with an Helium checkpoint and trained on audio-only data, Moshi outperforms previous work in this category on most metrics. Finally, while multimodal training improves common sense reasoning from speech (as shown by sStoryCloze performance), we observe mixed effects on lexical and syntactic judgments (sWUGGY and sBLIMP) compared to models trained only on audio data. While single-stream pretraining moderately degrades sWUGGY and sBLIMP, instruction finetuning severely affects sWUGGY, which means that instructed models have a harder time solving lexical judgments. We hypothesize that this is due to finetuning Moshi on data of varying quality and simulating noisy and reverberated conditions for the user stream (used to score spoken pairs for all metrics in Table 7) which makes fine lexical judgments harder to solve. Finally, Moshi scores 12 points higher on MMLU than Spirit-LM, thus demonstrating higher general knowledge and text understanding. We moreover emphasize that Moshi is the only model in Table 7 that integrates both semantic and acoustic tokens into a single generative model, unlike AudioLM which uses three separate stages, and VoxTLM, TWIST and Spirit-LM that only model semantic tokens and rely on an external vocoder. Thus, Moshi is the only model in this comparison that demonstrates strong linguistic modeling in both speech and text, while being able to model speech in any arbitrary voice and condition.

Table 8: **Evaluation of spoken question answering.** Spoken question answering (0-shot) on the Web Questions (Berant et al., 2013), LlaMA-Questions (Nachmani et al., 2024), and Trivia QA (Joshi et al., 2017) benchmarks, synthesized using a TTS engine. For the first two, we use the number reported by (Nachmani et al., 2024). For LlaMA-Questions, we use the audio provided by (Nachmani et al., 2024). For Web Questions and Trivia QA, we synthesize our own, keeping all of the questions. For Moshi, we only prepend one of the random incipits used during instruct fine tuning. We further provide the performance of our Helium text-only model as a top line.

Model	Web Q.	LlaMA Q.	Audio Trivia QA
<i>Audio only</i>			
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
<i>Text and audio</i>			
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
Moshi (w/o text batches in pre-training)	23.2	61.3	18.3
<i>Text</i>			
Helium (text)	32.3	75.0	56.4

Discussion. While “textless NLP” benchmarks have helped developing the first audio language models, we observe that they do not consistently provide good guidance in developing a dialogue model like Moshi. In particular, we find the lack of correlation between common sense metrics and lexical/syntactic judgments to be frequent and easily explainable by the diversity of acoustic conditions we use in training. Moreover, we do not observe a degradation in the model’s lexical variety or intelligibility when finetuning the model, which contradicts the reduction in sWUGGY. This is why we also evaluate spoken question answering in the next section as a way to probe the model’s common sense, knowledge and lexical abilities.

5.5 Spoken Question Answering

Metrics. We evaluate the spoken question answering abilities of our multi-stream Moshi model. We rely on Spoken Web Questions and Llama Questions, both introduced by Nachmani et al. (2024). We also synthesize an audio version of TriviaQA. When evaluating Moshi, we insert the audio tokens of the question into the user stream to simulate a user interaction, along with a final EPAD text token to trigger an immediate response from Moshi.

Baselines. We compare to Spectron and baselines used by Nachmani et al. (2024), all having been already introduced in Section 5.4 except for SpeechGPT (Zhang et al., 2024a). To measure the impact of Inner Monologue on spoken fluency, we compare these baselines with Moshi trained with and without Inner Monologue. As GSLM, AudioLM and TWIST are audio only, Moshi without Inner Monologue provides a fair comparison. On the other

Table 9: Linguistic quality and turn-taking statistics of generated dialogues. As we train our multi-stream model to generate both sides of the conversation, we can generate dialogues without the need to interact with a real user. This allows evaluating how much Moshi learns natural conversational dynamics.

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	

hand, Spectron and SpeechGPT rely on Chain-of-Modality—they generate an answer first as text, and then as speech—so we compare them to Moshi with Inner Monologue. Moreover, to quantify an eventual degradation in knowledge due to training on audio data, we also compare to Helium when evaluated on the textual counterpart to each spoken dataset.

Results. Table 8 reports accuracies on the three benchmarks. While audio-only Moshi significantly outperforms baselines in its categories, the most striking result is the impact of Inner Monologue on Moshi’s performance, almost tripling its accuracy on all benchmarks. This is remarkable as Inner Monologue only marginally increases inference cost (each multi-stream timestep requires generating 17 tokens, instead of 16 without). We emphasize that among all models in this comparison, Moshi not only provides the best spoken question answering performance, but is also the only one to model jointly semantic and acoustic tokens, such that it can handle interactions between arbitrary voices in many conditions. Moshi significantly outperforms SpeechGPT and Spectron while it is the only model compatible with streaming inference, as Chain-of-Modality requires generating a full answer in text before generating speech, while Inner Monologue generates both in a streaming fashion. Finally, we note that Moshi with Inner Monologue but without keeping 50% of text-only batches during pre-training leads to a noticeable drop in accuracy, showing that a warm start from Helium is not sufficient to retain the knowledge of the original text model throughout the training.

Discussion. Despite the strong performance of Moshi, we observe a weaker performance than the baseHelium model, which is consistent with the reduced MMLU of 49.7 reported in Table 7 from 54.3 with Helium. While the moderate differences on Web Questions and Llama Questions can be explained by training on audio data and thus reducing the amount of parameters dedicated to textual knowledge, the very large difference on Trivia QA incites us to inspect more thoroughly patterns of errors. We find that multiple-sentence questions (e.g. “The Terror of the Monster was an early title for a best-selling novel which inspired one of the highest-grossing movies of the mid-70’s. Under what name did it eventually terrify the reading and film going public?”) or ones with specific syntactic structure (e.g. “On the human body, a keloid is a type of what?”) are challenging for Moshi, due to it being finetuned on oral-style conversations that do not display such patterns. We hypothesize that covering more syntactic scenarios during finetuning could reduce this gap.

5.6 Quality and Statistics of Generated Dialogues

Metrics. Beyond single-turn question answering, we evaluate the linguistic quality of the generated dialogues by scoring them with an external language model, while also looking at turn-taking metrics. We follow the methodology of Nguyen et al. (2023). The turn-taking metrics are defined as follows: *Inter-Pausal Units* (IPU) are continuous stretches of speech that are separated by a silence of at least 0.2s on each side. *Pauses* are silences between IPUs of the same speakers. *Gaps* are silences between IPUs of different speakers. *Overlaps* are times where there are IPUs for both speakers. Following Nguyen et al. (2023), the semantic evaluation uses the open-source DialoGPT model Zhang et al. (2019) and we compute the perplexity of the transcribed dialogue by separating each speaker using the <|endoftext|> token expected by DialoGPT. We select 1000 random 10 seconds prompts from the Fisher dataset and use Moshi to generate continuations. For each one, 32 continuations are generated, for 3 different temperatures as it significantly affects the results.

Baselines. We compare to dGSLM (Nguyen et al., 2023), as it is also a full-duplex generative model, trained on the Fisher dataset. Nguyen et al. (2023) use 50 prompts with 50 continuations for each and report results for their dialogue model as well as a cascaded topline model (ASR + LM + TTS).

Results. Table 9 shows that Moshi performs as well as the cascaded model in terms of linguistic quality, despite being an audio-to-audio model. Both have a perplexity that is better than the ground truth, which is explained by these models being trained on data that is closer to what DialoGPT has been trained on compared to the Fisher dataset. This is a strong improvement over the non-cascaded model from (Nguyen et al., 2023), which is not able to generate coherent speech in this scenario.

5.7 Streaming ASR and TTS

Metrics. Section 3.4.4 and Appendix C describe how Inner Monologue can provide a streaming TTS or streaming ASR system by simply changing the delay it uses between text and audio tokens. In particular, we train a streaming TTS model by delaying audio tokens by 2 seconds, giving some lookahead to the text tokens, and teacher forcing text tokens at inference. Similarly, we train a streaming ASR model by delaying text tokens by 2 seconds, allowing the model to listen to audio content before generating text tokens. In that case, at inference we teacher force the audio tokens. We perform TTS with a temperature of 0.6 while we use greedy decoding for ASR, and evaluate on LibriSpeech (Panayotov et al., 2015) test-clean in Word Error Rate (WER). For TTS, we first transcribe the generated audio with HuBERT-Large (Hsu et al., 2021) finetuned on LibriSpeech 960h,¹⁵ and only consider sequences between 4 and 10s, which allows comparing to baselines such as Vall-E (Wang et al., 2023). We emphasize that no LibriSpeech data is seen during the training of our ASR and TTS systems.

Results. Our streaming TTS model obtains 4.7% of WER on LibriSpeech test-clean, which outperforms Vall-E’s 5.9% WER but is worse than NaturalSpeech 3 (Ju et al., 2024) with 1.81%. Yet, Moshi only requires 2 seconds of lookahead when Vall-E and NaturalSpeech

15. <https://huggingface.co/facebook/hubert-large-ls960-ft>

3 require access to the full sequence. Our ASR system yields 5.7% WER, while a Streaming FastConformer (Noroozi et al., 2024) gets 3.6% with a similar look-ahead. Note that our ASR system also provides alignments of transcripts with a precision of 80ms (the framerate of the Temporal Transformer).

Discussion. This limited experimentation is not intended to compete with state-of-the-art systems (in particular for ASR), but is rather designed to illustrate how Inner Monologue is flexible enough to cast several tasks into the same framework. We also emphasize that the standard evaluation on LibriSpeech test-clean does not provide a testbed to showcase strong capabilities of our TTS system, in particular its ability to model two speakers with multi-stream modeling, and generating consistent dialogues of spontaneous, expressive speech across 5 minutes (while Vall-E only evaluates sequences between 4 and 10 seconds of read speech). We reserve a thorough evaluation of streaming TTS for future work.

5.8 Compressing Moshi and Impact on Speech Quality

With most modern LLMs built off billions of parameters, model size is a well-known bottleneck for practical uses such as running on resource-constrained devices (e.g. laptop with user-grade GPU) or model deployment (e.g. serving many users on an online web demo). To address this, Post-Training Quantization (PTQ) is a widely used efficiency technique for compressing model weights and activations, with the downside of possible performance degradation. Recent work has shown that LLMs can often successfully be quantized to 8 bits with integer quantization, and sometimes to even lower bitwidths using more advanced techniques to handle outlier weights (Dettmers and Zettlemoyer, 2023; Dettmers et al., 2022; Frantar et al., 2023; Tseng et al., 2024). However, the literature on quantizing speech models is much more scarce than that of LLMs. Thus, in this section, we investigate how quantizing Moshi impacts its performance, both linguistically and especially acoustically, as we highlight certain audio degradations aggravated by model quantization.

Quantization Format. To quantize Moshi, we follow common design choices from the PTQ literature. In all results below, we settle on the following setting: (i) Activations are stored in `bfloat16` precision (BF16) and dynamically quantized to 8 bits using symmetric quantization (a.k.a. AbsMax) at the input of every linear layer; (ii) The model weights are quantized using asymmetric quantization (a.k.a. MinMax) for different bitwidths and block sizes. This includes both the Temporal Transformer as well as the Depth Transformer weights. In fact, we find that the Depth Transformer is reasonably robust to quantization, as keeping only its weights in high precision does not significantly improve audio quality. Only the initial embedding layers (both for text and audio), the RMSNorms and the Mimi codec are left unquantized. Finally, note that, although weight range setting is also common practice (Nagel et al., 2021), we do not finetune the obtained quantization scales using MSE as we find it has little impact on the quality of generated samples.

Results - Linguistic evaluation. To assess how quantization impacts the reasoning ability of the model, we evaluate the quantized models’ performance on the MMLU benchmark for the base Helium model trained on text-only data used as foundation for Moshi (Table 10), as well as for Moshi itself (Table 11). Generally, Helium is more robust to quantization than the final trained Moshi. Notably, assuming quantization blocks of size 32, quantizing

Table 10: **Linguistic impact of model compression on Helium**, as measured by MMLU. ‘WXA8’ indicates a model with weights quantized to ‘X’ bits and activations to 8 bits, using integer scalar PTQ. The model size in brackets is given in Gyabytes for a quantization block size of 32, and takes into account both the model weights and the quantization parameters stored in float16. With a fine enough granularity of quantization blocks, a 4 bits model stays within 2 points of MMLU of the floating point baseline.

BF16A8 ($\sim 15\text{GB}$)		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

Table 11: **Linguistic impact of model compression on Moshi**, measured by MMLU for different quantized Moshi on the text tokens generated by Inner Monologue directly. As for the previous table, the model size is indicated for the block size of 32.

a) Moshi after single-stream pretraining			b) Moshi after multi-stream instruct		
BF16A8 (15.24GB)	49.8		BF16A8 (16.74GB)	49.7	
Bitwidth vs Block size	256	32	Bitwidth vs Block size	256	32
W8A8 (8.33GB)	48.8	48.5	W8A8 (9.20GB)	47.6	47.6
W6A8 (6.95GB)	48.5	49.1	W6A8 (7.70GB)	48.1	48.3
W5A8 (6.02GB)	47.4	48.5	W5A8 (6.69GB)	46.7	47.2
W4A8 (4.64GB)	44.7	45.7	W4A8 (5.18GB)	39.8	42.2
W3A8 (3.72GB)	26.1	35.7	W3A8 (4.18GB)	27.7	29.9
W2A8 (2.80GB)	23.4	24.4	W2A8 (3.17GB)	24.5	24.9

Helium weights to 4 bits yields a 3.43 times smaller model which remains within 2 points of MMLU of the floating point baseline. This particular quantization format is also almost identical to llama.cpp’s¹⁶ Q4_0, hence can be readily deployed for efficient inference.

In contrast, the same quantization recipe used on Moshi leads to a more drastic loss of performance ranging from 5 to 10 points of MMLU. In the online demo, we keep the weights in 8-bit format as it results in a more reasonable drop of 2 points for a model roughly twice smaller than the floating point baseline.

Results - Audio Quality. To assess the audio quality of samples generated by the quantized models, we make use of the MOSNet metric from Lo et al. (2019) as implemented in speechmetrics.¹⁷ More specifically, we generate a short prompt (64 tokens) from the unquantized model, then generate completions from each of the quantized models with a temperature of $t = 0.8$ and a sequence length of 1024 tokens. We repeat this process 500 times, and report the distribution of MOSNet scores over non-overlapping windows

16. <https://github.com/ggerganov/llama.cpp>

17. <https://github.com/aliutkus/speechmetrics>

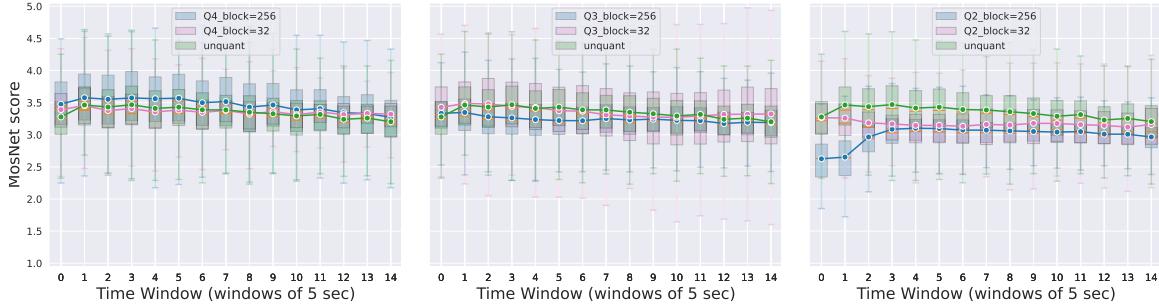
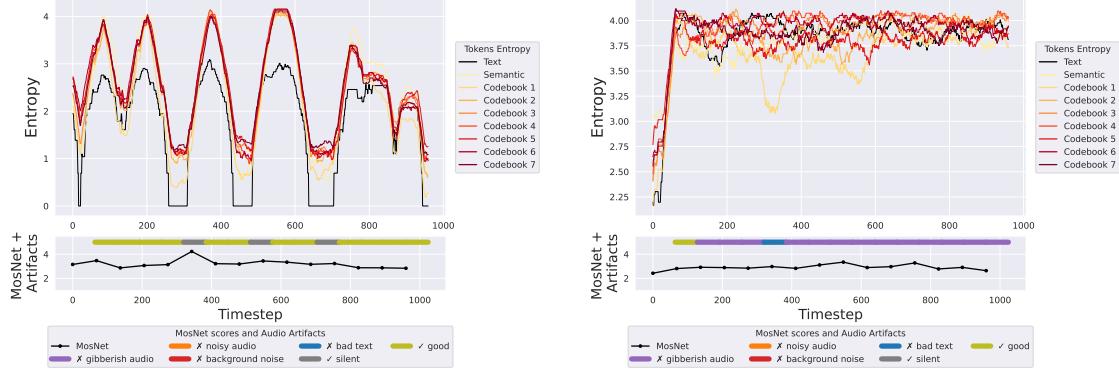


Figure 5: Acoustic impact of model compression on Moshi. MOSNet evaluation of samples generated by models compressed for different bitwidths. We evaluate the MOSNet scores across non overlapping windows of 5s, and report the distribution of these scores over 500 samples for each model.

in Figure 5. While the MOSNet scores exhibit a large variance across samples, there is generally little degradation of the audio quality after quantizing the model’s weights down to 4 bits. For lower bitwidths however, we observe qualitatively that the MOSNet scores lack sensitivity towards some of the more severe audio degradations caused by aggressive quantization: For instance, it does not distinguish between pure audio artifacts (e.g., noisy voice) from artifacts in the speech pattern (e.g., increased repetitiveness of the model). This is in line with the lack of consistency between objective and subjective audio quality metrics that we observed in Section 5.2, and in addition, MOSNet was designed for a very different type of benchmark, namely, to mimic human ratings on evaluating converted speech, so it is not surprising for it not to be less sensitive to such artifacts. Instead, to measure the presence or absence of such degradation in the audio samples, we first observe that certain audio artifacts are identifiable from the entropy spectrums of the generated text and audio tokens: A few examples are illustrated in Figure 6 and we further detail the types of artifacts and how we measure them in Appendix D.

Following this insight, we measure the presence or absence of different audio artifacts on the same generated audio samples as the ones used in the previous MOSNet analysis. We report the results in Table 12, as well as a more detailed per timestep analysis in Figure 11 of Appendix D. At a bitwidth of 4, we again observe little audio degradation. Decreasing to 3-bit format, audio degradations are more apparent and tend to become more frequent along the generation timestep, although the finer granularity quantization format is generally more robust to these artifacts. Nevertheless, both quantization formats display significantly degraded audio quality when weights are aggressively quantized to 2 bits, which we also observe qualitatively.

Discussion. The linguistic abilities of Moshi are more sensitive to quantizing the model weights and activations than its output audio quality. More specifically, the audio quality remains close to that of the floating point baseline down to 4 bits precision, even when quantizing the full model, including the Depth Transformer. In contrast, the MMLU performance suffers significant drops when quantizing the model weights below 6 bits using post-training only quantization. Following recent quantization techniques (Tseng et al.,



(a) Entropy spectrum of a well-behaved sample (no noticeable degradation). Short silences occur naturally in Moshi’s output due to the model’s multi-stream abilities (reflecting the other speaker’s turn)

(b) Significant degradations occur at low bitwidth (W2A8). These are not always well reflected by the MOSNet scores’ magnitude, but the entropy of the text token is visibly higher.

Figure 6: **Audio artifacts caused by model compression.** Example of typical entropy spectrums capturing specific audio artifacts caused by model quantization. For each timestep, we compute the entropy over the past 128 tokens, independently for the text and audio codebooks tokens. Then, we measure the presence or absence of the different artifacts over non-overlapping windows of 64 tokens, as described in [Appendix D](#).

Table 12: **Distribution of audio artifacts caused by model compression.** Percentage of audio artifacts measured in the entropy spectrum of text and speech generated tokens, as described in [Appendix D](#). These results averaged across 500 samples generated by different versions of the same quantized Moshi, and across 16 timesteps of 64 tokens. Values of 0 % are omitted in the table for better readability.

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

2024), we may expect improved performance at lower bitwidth by using quantized aware finetuning instead of PTQ. However, as Moshi’s training pipeline from [Section 4](#) involves multiple stage and training datasets, this would require a more thorough investigation into designing quantized training phases and calibration datasets, to preserve all of Moshi’s abilities lost after quantization.

6 Safety

In parallel with the development of Moshi, we explore different directions related to the safety of AI generated content. In this section, we specifically consider several questions regarding the content generated by Moshi, each addressed in a dedicated subsection:

1. How does our model behave in terms of producing toxic content?
2. How to avoid that the model regurgitates audio content from the training set?
3. How do we ensure that the model uses the voice we intend to give to Moshi?
4. How to identify if a given content has been generated by Moshi?

6.1 Toxicity Analysis

The scientific community has devoted in the last years some effort to address bias and toxicity problems for text generation models. In contrast, audio safety is far less developed. It is not straightforward to compare audio and text models in an apple-to-apple comparison, as they differ in their usage, and multiple meanings are conveyed by non-verbal signal (irony, tone, etc.). In spite of these limitations and in order to facilitate the comparison of Moshi with text generation models, in this first analysis we restrict our toxicity analysis to the text produced by the model. We adopt the ALERT benchmark¹⁸ (Tedeschi et al., 2024), which evaluates safety under multiple categories (hate, self-harm, weapon, crime, sex, substance). Table 18 in Appendix D reports our detailed toxicity analysis on this benchmark. The aggregated score for Moshi and popular text-only models is as follows:

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

With this analysis, we see that Moshi falls into the middle of this table in terms of rank. The industry models perform the best, which is expected considering the massive amount of private annotation, red-teaming and feedback loop from which these models have benefited.

6.2 Regurgitation Analysis

The problem of a model generating content which it has seen at training time, which we refer to as *regurgitation*, is closely related to overfitting: The more a model has seen a sequence or a subsequence during training, the more likely it is to generate this exact sequence during the generation process. Note, for a speech model, it is not only the text that can be regurgitated, but also the voice pitch, tone, and potentially the background melody if present at training time. It is therefore important to mitigate¹⁹ potential intellectual property issues related to regurgitation, such as reproduction of copyrighted content or audio generation with the voice of a person without permission.

18. <https://github.com/Babelscape/ALERT>

19. There is currently no way to fully prevent these issues. While it is essential to develop algorithms and methodologies that limit the occurrences of problematic generations, part of the question is related to how generative AI is regulated.

Table 13: **Regurgitation of training data with condition-free generation** from different models. We measure how many times each model generates the most frequent duplicate segment audio in the training data, for different values of the temperature. With dataset deduplication, we do not observe any exact re-generation (out of 10^5) of the most frequent segment, even if we prompt the model with the first 3s of this audio segment.

	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

Evaluation protocol. For each model, we measure the proportion of generations (out of 100,000) that reproduce the most frequent audio segment detected in our whole training dataset. For this purpose, we have first developed a matching system that detects the most frequent audio segments, see [Appendix B](#). We select the most frequent one that is long enough (16 seconds) and easy to detect from text and audio. We measure the proportion of generations that exactly match this most frequent segment. For the matching, we initially use both audio and text matching, but observe that text-based matching has a higher recall for the initial matching step. We manually verify all the generations to filter out outliers that are not exact matches.

Unconditioned and prompted generation: We first measure what happens with unconditional generation, to evaluate whether the model tends to generate specific sequences when not guided by a prompt. In a complementary manner, we prompt the model with the 3 first seconds of the most frequent audio segment and measure how many times the continuation is identical to this training set audio. [Table 13](#) reports these regurgitation results.

Results & Impact of fine-tuning. We observe that the pre-trained model trained on the raw dataset often generates frequent sequences from the training set. The sampling temperature has an important effect on the regurgitation rate: the values typically employed for generation (0.6–1.0) are more prone to regurgitation. Out of 1000 generations, the model fine-tuned for conversation does not generate the most frequent training sequence. As a disclaimer, we point out that fine-tuning could potentially be over-ridden and therefore may not be sufficient *per se* to avoid regurgitation.

Similar to what happens with textual models ([Carlini et al., 2022](#)), regurgitation is significantly impacted by the number of times that the model uses a given sequence for training. Therefore, we evaluate the impact of deduplicating the training dataset by identifying all

Table 14: **Speaker consistency along time.** We measure how often the speaker embedding from Moshi’s segment is closer to its reference segment than the user, when computing speaker embeddings from segments further away from the reference.

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%

the audio segments that are frequent, and in turn by filtering them out at training time. In [Table 13](#), we observe that this pre-processing step brings the number of regurgitations of the most frequent sequence to zero, even without any fine-tuning step.

6.3 System 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. In order to evaluate to which extent Moshi adopts a voice of the user instead of the target voice, we use the following protocol:

- Generate 100 hours of conversations between Moshi and a second synthetic speaker.
- Run a speaker verification model (WavLM ([Chen et al., 2022](#)) large) on each segment to extract the speaker embeddings.
- Compute the cosine similarity between the embeddings of each main speaker’s segment with *(i)* the first segment of the main speaker and *(ii)* with the first segment of the generated speaker.
- *Note:* we exclude all the segments with a start time before 15 seconds so as to avoid counting the first turn of speech of the main speaker as it acts as the reference.

Over the generated datasets, there are 10 249 occurrences (98.7%) where the voice of the main speaker is closer to the reference segment of the main speaker and 133 occurrences (1.3%) where the voice is closer to the reference segment of the other speaker. We are also interested in how speaker’s consistency evolves over time. Following [Borsos et al. \(2023\)](#) we compute the same ratio as above but on groups of segments that start at specific times, to measure drift along time. [Table 14](#) shows that speaker consistency remains stable along time, meaning that we do not observe a drift as the conversation goes on. This shows that the simple choice of using a consistent voice for the system during instruction tuning is enough to provide robustness at inference time.

6.4 Identification of the Content Generated by Moshi: Watermarking

For determining if a given audio has been generated by Moshi, we have investigated two complementary solutions: indexing and watermarking. The first, namely audio indexing, only applies in the case where we have access to the machine that generates the content, like in the case of the Moshi demo. We describe our audio matching system in [Appendix B](#).

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.

		audio duration →	average detection score	
↓ audio post-processing			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

Below in this subsection, we discuss more specifically watermarking, where the objective is to add unnoticeable marks to the generated audio.

Evaluation of signal-based watermarking. We investigate if existing watermarking methods for audio can be used as a way to re-identify content generated by Moshi. For this purpose, we analyze the robustness of the Audioseal method (San Roman et al., 2024b) in our context. It is available as an open-source library.²⁰ For this evaluation, we resample the audio signal to 16kHz so that the sampling rate matches the one recommended in Audioseal instructions. We measure the average mark detection scores in the following settings:

- No watermark: we measure the detection score measured when no mark was added.
- Watermark no attack: no modification of the watermarked audio signal;
- Pink noise: we add a small pink noise ($\sigma = 0.2$) to the watermarked audio;
- RVQGAN: we compress and decompress the audio signal with a recent state-of-the-art auto-encoder (Kumar et al., 2023). We use the publicly available pre-trained 16Khz model²¹ which differs from the 24kHz model used as a baseline in Section 5.2.
- Mimi auto-encoder: we use our own tokenizer to compress and decompress the signal. This operation is performed with 24kHz audio and therefore involves two re-sampling stages (from 16kHz to 24kHz and back to 16kHz).

We report the results in Table 15. We observe that the mark yields high detection rates when the audio is unchanged. With aggressive Pink-Noise, one needs a relatively long sequence to get a high detection score. However, the mark is not robust to a strong compression: the two auto-encoders that we consider are low bitrate and therefore discard anything not related to the signal reconstruction. As a result, our Mimi codec removes the mark to a level that makes a watermarked audio indistinguishable from a non-watermarked audio, making such a signal-based watermarking useless in this context.

Exploration on generative-based watermarking for audio. Given that a recent state-of-the-art signal-based audio watermarking is not robust to a simple non-adversarial auto-encoding method, we investigated the possibility of watermarking the generation process itself. This solution was recently proposed for text generation, in particular in the

20. <http://github.com/facebookresearch/audioseal>

21. <https://github.com/descriptinc/descript-audio-codec>

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 ($\Delta T=40\text{ms}$), and re-encoding with RVQGAN. All results are averaged over 1000 generated sequences of 1 minute.

\downarrow codec	attacks		RQ level \rightarrow	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$
	ΔT	RVQGAN	(semantic)								
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

works of [Aaronson and Kirchner \(2023\)](#) and [Kirchenbauer et al. \(2023\)](#). These two methods operate similarly: at sampling time, they bias the probabilities driving the generation process. They differ from each other by how they modify the probabilities, yet in both cases the sampling is parameterized by a hash function that preferably depends on a local context. These solutions were improved by [Fernandez et al. \(2023\)](#), who proposed a better mark detector, in particular by addressing the issue of repetitive patterns.

We have investigated how to apply these discrete watermarking methods to our audio generation pipeline. For this purpose, we need to encode the audio signal back to tokens in order to identify if the mark is present or not. One issue is that *the codec is not idempotent*: if we generate a waveform from tokens and then re-encode it back into tokens, the re-generated tokens are likely to be different from the ones generated with high probability, even if the audio has not suffered any noise addition. We quantify this problem in [Table 16](#). The semantic token is robust to some extent, while the other quantization indices are increasingly less robust as they depend on the previous quantizer level. One key issue is that the tokens do not resist to a moderate temporal shift. This is especially true for the Mimi codec, which is purposely optimized on a perceptual objective, as opposed to a fidelity reconstruction criterion.

Discussion on generative audio watermarking. The lack of idempotence is problematic for the aforementioned sampling-based watermarking methods, as it affects the reliability of the detector when measuring the sampling bias. Noticeably, in order for these methods to work properly, the n-tuples that gives the context to the hash key must be stable enough during several consecutive tokens. Reducing the context length improves the stability but drastically increases the likelihood of producing degenerated audio sequences, similar to the degeneration problem observed by [\(Holtzman et al., 2019\)](#).

While we regard this attempt of employing text-based watermarking as a negative result, hereafter we discuss a few potential ways for circumventing the aforementioned problem of token stability through re-encoding:

- Marking only the RQ first levels improves the stability. In our preliminary experiments, using these indices as context in the hash function, and limiting the dependence on previous timestamps, significantly increases the stability (although not sufficiently).
- The idempotence could be improved by adding a specific loss in the discrete latent space, such that the audio tokens are stable through auto-encoding.
- Potentially this auto-encoding could be learned to be resilient to signal transformation, similar to what is proposed when learning image watermarking based on neural networks (Zhu, 2018; Fernandez et al., 2022). In view of our analysis, adding some tolerance to moderate temporal shift is especially important.
- The text could be marked instead of the audio. One downside is that text is a lower-capacity channel for adding a mark, and would not be sufficient for short conversations. Another problem is that detecting the mark requires a reliable transcription.

Last but not least, some exploration is needed to ensure that it is not trivial to remove the watermarking procedure when open-sourcing a model. As an example, the only thing to remove the watermark with the implementation associated with the stable diffusion model was to comment a line of code.²² A promising work in this direction is the study by Sander et al. (2024), who show that it is possible to detect when a model has been trained on watermarked text. A method exploiting this observation has just been shared by San Roman et al. (2024a): the watermarking is implicitly added through the training data, in the spirit of “radioactive data” by Sablayrolles et al. (2020).

7 Conclusion

In this work, we introduce Moshi, the first real-time, full-duplex spoken dialogue system. The first component of Moshi is Helium, a 7B parameter text LLM which is competitive with open-weights models trained with a similar compute budget. To encode audio into discrete units amenable to language modeling, we introduce Mimi, a semantic-acoustic neural audio codec which provides state-of-the-art audio quality at low bitrates while operating at low framerates compatible with real-time generation. We then introduce a new, hierarchical multi-stream architecture that supports generating arbitrary conversations in a speech-to-speech manner. We moreover show that speech-to-speech generation can be drastically improved by Inner Monologue, a new method that generates text tokens as a prefix to audio tokens, while remaining compatible with streaming inference. Our experiments show that Moshi demonstrates state-of-the-art spoken question answering and dialogue modeling while displaying satisfying levels of safety by not generating toxic content and remaining consistent in its voice. To summarize, we introduce a complete suite of models and recipes, from text LLMs to neural audio codecs and generative audio models, which we combine into a real-time spoken dialogue system with a 160ms theoretical latency able to follow complex multi-turn conversations across 5 minutes. We release both Mimi and Moshi to foster the

22. <https://github.com/Stability-AI/stablediffusion/blob/main/scripts/txt2img.py#L363>

development of such applications. Additionally, we show how our Inner Monologue method allows designing streaming TTS and streaming ASR just by changing the delay between text and audio tokens. We believe that both Inner Monologue and multi-stream modeling will have a positive impact on speech-to-speech and audio-to-audio beyond dialogue modeling.

Acknowledgments and Disclosure of Funding

This project is funded by Iliad Group, CMA CGM Group and Schmidt Sciences. We thank Xavier Niel, Rodolphe Saadé, Eric Schmidt, Aude Durand, Séverine Grégoire and Nicolas Granatino, for their support; as well as Sarah Hôte and Guillaume Rouzaud at Kyutai for their help. We also thank Alice, the voice artist who strived to give Moshi online demo its voice, Elie Raffier who built the user interface for this demo, and Hugging Face for inference compute donation. Audio training data set was built with the help of Noatune Studios and Landospeech. Model training was conducted at Scaleway.

References

- Scott Aaronson and Hendrik Kirchner. Watermarking gpt outputs, 2023. URL <https://www.scottaaronson.com/talks/watermark.ppt>. 40
- Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023. 22
- Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv:2006.11477*, 2020. 4, 9
- Jonathan Berant, Andrew K. Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Conference on Empirical Methods in Natural Language Processing*, 2013. 29
- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, 2020. 22
- Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matthew Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. Audioltm: A language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2022. 2, 3, 4, 5, 9, 16, 23, 27, 28, 29
- Zalán Borsos, Matthew Sharifi, Damien Vincent, Eugene Kharitonov, Neil Zeghidour, and Marco Tagliasacchi. Soundstorm: Efficient parallel audio generation. *CoRR*, abs/2305.09636, 2023. doi: 10.48550/ARXIV.2305.09636. 38

Hervé Bredin. pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe. In *Proc. INTERSPEECH 2023*, 2023. [20](#)

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. [2](#)

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022. [37](#)

Özgür Çetin and Elizabeth Shriberg. Analysis of overlaps in meetings by dialog factors, hot spots, speakers, and collection site: insights for automatic speech recognition. In *Ninth International Conference on Spoken Language Processing, INTERSPEECH-ICSLP 2006, Pittsburgh, PA, USA, September 17-21, 2006*. ISCA, 2006. [2](#)

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, Xiangzhan Yu, and Furu Wei. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE J. Sel. Top. Signal Process.*, 2022. [4](#), [10](#), [12](#), [38](#)

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek B Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Oliveira Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling language modeling with Pathways. *arXiv:2204.02311*, 2022. [5](#)

Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. w2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU*. IEEE, 2021. [9](#)

Christopher Cieri, David Miller, and Kevin Walker. Fisher english training speech parts 1 and 2. <https://doi.org/10.35111/da4a-se30>, 2004. [8](#), [18](#), [21](#)

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. [22](#)

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. [10](#)

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. [4](#), [9](#), [14](#), [25](#), [26](#)

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022. [7](#)

Alexandre Défossez, Gabriel Synnaeve, and Yossi Adi. Real time speech enhancement in the waveform domain. In *Interspeech*, 2020. [21](#)

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *Transactions on Machine Learning Research*, 2023. [3](#), [9](#), [10](#), [11](#)

Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7750–7774. PMLR, 23–29 Jul 2023. [32](#)

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *NeurIPS*, 2022. [32](#)

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. [4](#)

Harishchandra Dubey, Ashkan Aazami, Vishak Gopal, Babak Naderi, Sebastian Braun, Ross Cutler, Hannes Gamper, Mehrsa Golestaneh, and Robert Aichner. Icassp 2023 deep noise suppression challenge. In *ICASSP*, 2023. [21](#)

Ewan Dunbar, Mathieu Bernard, Nicolas Hamilakis, Tu Anh Nguyen, Maureen de Seyssel, Patricia Rozé, Morgane Rivière, Eugene Kharitonov, and Emmanuel Dupoux. The zero resource speech challenge 2021: Spoken language modelling. In *Interspeech*. ISCA, 2021. doi: 10.21437/Interspeech.2021-1755. [4](#)

Zach Evans, Julian D Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Stable audio open. *arXiv preprint arXiv:2407.14358*, 2024. [4](#)

Pierre Fernandez, Alexandre Sablayrolles, Teddy Furon, Hervé Jégou, and Matthijs Douze. Watermarking images in self-supervised latent spaces. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022. [41](#)

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. Three bricks to consolidate watermarks for large language models. In *Proc. International Workshop on Information Forensics and Security (WIFS)*, 2023. [40](#)

Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. OPTQ: Accurate post-training quantization for generative pre-trained transformers. In *ICLR*, 2023. [32](#)

Team Gemini, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. [2](#)

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models. *Preprint*, 2024. [22](#)

Michael Hassid, Tal Remez, Tu Anh Nguyen, Itai Gat, Alexis Conneau, Felix Kreuk, Jade Copet, Alexandre Défossez, Gabriel Synnaeve, Emmanuel Dupoux, Roy Schwartz, and Yossi Adi. Textually pretrained speech language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. [5](#), [27](#), [28](#), [29](#)

Julien Hauret, Thomas Joubaud, Véronique Zimpfer, and Éric Bavu. Eben: Extreme bandwidth extension network applied to speech signals captured with noise-resilient body-conduction microphones. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023. [11](#)

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016a. [10](#)

- Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016b. [7](#)
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020. [22](#), [28](#)
- Andrew Hines, Jan Skoglund, Anil C Kokaram, and Naomi Harte. ViSQOL: an objective speech quality model. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015 (1):1–18, 2015. [23](#)
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in neural information processing systems*, 2020. [4](#)
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022. doi: 10.48550/ARXIV.2203.15556. [2](#)
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019. [40](#)
- Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE ACM Trans. Audio Speech Lang. Process.*, 29, 2021. [4](#), [9](#), [31](#)
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017. [22](#), [29](#)
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016. [9](#)
- Zeqian Ju, Yuancheng Wang, Kai Shen, Xu Tan, Detai Xin, Dongchao Yang, Yanqing Liu, Yichong Leng, Kaitao Song, Siliang Tang, et al. Naturalspeech 3: Zero-shot speech synthesis with factorized codec and diffusion models. *arXiv preprint arXiv:2403.03100*, 2024. [31](#)
- Jacob Kahn, Morgane Rivière, Weiyi Zheng, Evgeny Kharitonov, Qiantong Xu, Pierre-Emmanuel Mazaré, Julien Karadayi, Vitaliy Liptchinsky, Ronan Collobert, Christian Fuegen, Tatiana Likhomanenko, Gabriel Synnaeve, Armand Joulin, Abdelrahman Mohamed, and Emmanuel Dupoux. Libri-light: A benchmark for ASR with limited or no supervision. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7669–7673. IEEE, 2020. doi: 10.1109/ICASSP40776.2020.9052942. [23](#)

Eugene Kharitonov, Damien Vincent, Zalán Borsos, Raphaël Marinier, Sertan Girgin, Olivier Pietquin, Matt Sharifi, Marco Tagliasacchi, and Neil Zeghidour. Speak, read and prompt: High-fidelity text-to-speech with minimal supervision. *Trans. Assoc. Comput. Linguistics*, 11:1703–1718, 2023. doi: 10.1162/TACL_A_00618. [4](#)

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations*, 2015. [11](#)

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning*. PMLR, 2023. [40](#)

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018. [7](#), [16](#)

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved RVQGAN. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36*, 2023. [11](#), [24](#), [39](#)

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. In *Advances in Neural Information Processing Systems*, 2024. [23](#), [24](#)

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019. [22](#)

Kushal Lakhotia, Eugene Kharitonov, Wei-Ning Hsu, Yossi Adi, Adam Polyak, Benjamin Bolte, Tu-Anh Nguyen, Jade Copet, Alexei Baevski, Abdelrahman Mohamed, et al. On generative spoken language modeling from raw audio. *Transactions of the Association for Computational Linguistics*, 9:1336–1354, 2021. [4](#), [23](#), [27](#), [28](#), [29](#)

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, 2022. [4](#), [13](#), [14](#)

Jean-Marie Lemercier, Simon Rouard, Jade Copet, Yossi Adi, and Alexandre Défossez. An independence-promoting loss for music generation with language models. In *ICML*, 2024. [14](#)

Haohe Liu, Ke Chen, Qiao Tian, Wenwu Wang, and Mark D Plumbley. AudioSR: Versatile audio super-resolution at scale. *arXiv preprint arXiv:2309.07314*, 2023a. [18](#)

- Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. AudioLDM: Text-to-audio generation with latent diffusion models. In *Proceedings of the International Conference on Machine Learning*, 2023b. 4
- Haohe Liu, Xuenan Xu, Yi Yuan, Mengyue Wu, Wenwu Wang, and Mark D Plumbley. Semanticodec: An ultra low bitrate semantic audio codec for general sound. *arXiv preprint arXiv:2405.00233*, 2024. 23, 24
- Team Llama. The llama 3 herd of models. *preprint*, 2024. 2
- Chen-Chou Lo, Szu-Wei Fu, Wen-Chin Huang, Xin Wang, Junichi Yamagishi, Yu Tsao, and Hsin-Min Wang. Mosnet: Deep learning based objective assessment for voice conversion. In *Proc. Interspeech 2019*, 2019. 23, 33
- Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 7
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 7
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. 11, 20
- Jérôme Louradour. whisper-timestamped. <https://github.com/linto-ai/whisper-timestamped>, 2023. 18
- Soumi Maiti, Yifan Peng, Shukjae Choi, Jee weon Jung, Xuankai Chang, and Shinji Watanabe. Voxtlm: unified decoder-only models for consolidating speech recognition/synthesis and speech/text continuation tasks. *ArXiv*, abs/2309.07937, 2023. 5, 27, 28
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018. 22
- Kentaro Mitsui, Koh Mitsuda, Toshiaki Wakatsuki, Yukiya Hono, and Kei Sawada. Psdm: Parallel generation of text and speech with llms for low-latency spoken dialogue systems, 2024. 5
- Eliya Nachmani, Alon Levkovitch, Roy Hirsch, Julian Salazar, Chulayuth Asawaroengchai, Soroosh Mariooryad, Ehud Rivlin, RJ Skerry-Ryan, and Michelle Tadmor Ramanovich. Spoken question answering and speech continuation using spectrogram-powered LLM. In *The Twelfth International Conference on Learning Representations*, 2024. 5, 29
- Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization, 2021. 32
- Tu Anh Nguyen, Eugene Kharitonov, Jade Copet, Yossi Adi, Wei-Ning Hsu, Ali Elkahky, Paden Tomasello, Robin Algayres, Benoît Sagot, Abdelrahman Mohamed, and Emmanuel Dupoux. Generative spoken dialogue language modeling. *Transactions of the Association for Computational Linguistics*, 11:250–266, 2023. doi: 10.1162/tacl_a_00545. 6, 30, 31

Tu Anh Nguyen, Benjamin Muller, Bokai Yu, Marta R. Costa-jussà, Maha Elbayad, Sravya Popuri, Paul-Ambroise Duquenne, Robin Algayres, Ruslan Mavlyutov, Itai Gat, Gabriel Synnaeve, Juan Pino, Benoît Sagot, and Emmanuel Dupoux. Spirit-lm: Interleaved spoken and written language model. *CoRR*, abs/2402.05755, 2024. doi: 10.48550/ARXIV.2402.05755. [5](#), [27](#), [28](#)

Vahid Noroozi, Somshubra Majumdar, Ankur Kumar, Jagadeesh Balam, and Boris Ginsburg. Stateful conformer with cache-based inference for streaming automatic speech recognition. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12041–12045. IEEE, 2024. [32](#)

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015. doi: 10.1109/ICASSP.2015.7178964. [23](#), [31](#)

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018. [4](#)

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. [12](#)

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR, 2023. [4](#), [16](#), [18](#), [25](#), [26](#)

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy P. Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, Ioannis Antonoglou, Rohan Anil, Sebastian Borgeaud, Andrew M. Dai, Katie Millican, Ethan Dyer, Mia Glaese, Thibault Sottiaux, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, James Molloy, Jilin Chen, Michael Isard, Paul Barham, Tom Hennigan, Ross McIlroy, Melvin Johnson, Johan Schalkwyk, Eli Collins, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Clemens Meyer, Gregory Thornton, Zhen Yang, Henryk Michalewski, Zaheer Abbas, Nathan Schucher, Ankesh Anand, Richard Ives, James Keeling, Karel Lenc, Salem Haykal, Siamak Shakeri, Pranav Shyam, Aakanksha Chowdhery, Roman Ring, Stephen Spencer, Eren Sezener, and et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *CoRR*, abs/2403.05530, 2024. doi: 10.48550/ARXIV.2403.05530. [4](#)

Paul K. Rubenstein, Chulayuth Asawaroengchai, Duc Dung Nguyen, Ankur Bapna, Zalán Borsos, Félix de Chaumont Quiry, Peter Chen, Dalia El Badawy, Wei Han, Eugene Kharitonov, Hannah Muckenhirk, Dirk Padfield, James Qin, Danny Rozenberg, Tara N. Sainath, Johan Schalkwyk, Matthew Sharifi, Michelle Tadmor Ramanovich, Marco Tagliasacchi, Alexandru Tudor, Mihajlo Velimirovic, Damien Vincent, Jiahui Yu,

Yongqiang Wang, Vicky Zayats, Neil Zeghidour, Yu Zhang, Zhishuai Zhang, Lukas Zilkha, and Christian Havnø Frank. Audiopalm: A large language model that can speak and listen. *CoRR*, abs/2306.12925, 2023. doi: 10.48550/ARXIV.2306.12925. [4](#), [5](#)

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Radioactive data: tracing through training. In *International Conference on Machine Learning*, pages 8326–8335. PMLR, 2020. [41](#)

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9): 99–106, 2021. [22](#)

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29*, 2016. [10](#)

Robin San Roman, Pierre Fernandez, Antoine Deleforge, Yossi Adi, and Romain Serizel. Latent watermarking of audio generative models. *arXiv e-prints*, pages arXiv–2409, 2024a. [41](#)

Robin San Roman, Pierre Fernandez, Hady Elsahar, Alexandre Défossez, Teddy Furon, and Tuan Tran. Proactive detection of voice cloning with localized watermarking. In *International Conference on Machine Learning*, volume 235, 2024b. [39](#)

Tom Sander, Pierre Fernandez, Alain Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. *arXiv preprint arXiv:2402.14904*, 2024. [41](#)

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019. [22](#)

Thomas Schatz, Vijayaditya Peddinti, Francis R. Bach, Aren Jansen, Hynek Hermansky, and Emmanuel Dupoux. Evaluating speech features with the minimal-pair ABX task: analysis of the classical MFC/PLP pipeline. In *Interspeech*. ISCA, 2013. [12](#), [23](#)

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. [7](#)

Tanya Stivers, Nick J. Enfield, Penelope Brown, Christina Englert, Makoto Hayashi, Trine Heinemann, Gertie Hoymann, Federico Rossano, Jan Peter De Ruiter, Kyung-Eun Yoon, Stephen C. Levinson, Paul Kay, and Krishna Y. Universals and cultural variation in turn-taking in conversation. *Proceedings of the National Academy of Sciences*, 106:10587 – 10592, 2009. [2](#)

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024. [7](#)

- Marco Tagliasacchi, Yunpeng Li, Karolis Misiunas, and Dominik Roblek. SEANet: A multi-modal speech enhancement network. In *Interspeech*, 2020. [10](#), [11](#)
- MosaicML NLP Team. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. Accessed: 2023-05-05. [22](#)
- Simone Tedeschi, Felix Friedrich, Patrick Schramowski, Kristian Kersting, Roberto Navigli, Huu Nguyen, and Bo Li. Alert: A comprehensive benchmark for assessing large language models' safety through red teaming, 2024. [36](#), [62](#)
- Teknium. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023. URL <https://huggingface.co/datasets/teknium/OpenHermes-2.5>. [18](#)
- Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV*, 2021. doi: 10.1109/ICCV48922.2021.00010. [10](#)
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023a. doi: 10.48550/ARXIV.2302.13971. [2](#)
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b. [22](#)
- Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. In *ICML*, 2024. [32](#), [34](#)
- Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. [9](#)
- Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. In *arXiv:1609.03499*, 2016. [10](#)
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, and Lukasz Kaiser and. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017. [3](#), [5](#), [7](#)
- Avery Li-Chun Wang. An industrial strength audio search algorithm. In *ISMIR*. Washington, DC, 2003. [54](#), [56](#)
- Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. Neural codec language models are zero-shot text to speech synthesizers. *CoRR*, abs/2301.02111, 2023. doi: 10.48550/ARXIV.2301.02111. [4](#), [9](#), [31](#)

Peng Wang, Songshuo Lu, Yaohua Tang, Sijie Yan, Yuanjun Xiong, and Wei Xia. A full-duplex speech dialogue scheme based on large language models. *CoRR*, abs/2405.19487, 2024. doi: 10.48550/ARXIV.2405.19487. [6](#)

Dongchao Yang, Jinchuan Tian, Xu Tan, Rongjie Huang, Songxiang Liu, Xuankai Chang, Jiatong Shi, Sheng Zhao, Jiang Bian, Xixin Wu, et al. Uniaudio: An audio foundation model toward universal audio generation. *arXiv preprint arXiv:2310.00704*, 2023. [2](#), [4](#), [13](#), [14](#), [15](#)

Shu-Wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y. Lin, Andy T. Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, Tzu-Hsien Huang, Wei-Cheng Tseng, Ko-tik Lee, Da-Rong Liu, Zili Huang, Shuyan Dong, Shang-Wen Li, Shinji Watanabe, Abdelrahman Mohamed, and Hung-yi Lee. SUPERB: speech processing universal performance benchmark. In Hynek Hermansky, Honza Cernocký, Lukáš Burget, Lori Lamel, Odette Scharenborg, and Petr Motlíček, editors, *22nd Annual Conference of the International Speech Communication Association, Interspeech 2021, Brno, Czechia, August 30 - September 3, 2021*, pages 1194–1198. ISCA, 2021. doi: 10.21437/INTERSPEECH.2021-1775. [4](#)

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. Megabyte: Predicting million-byte sequences with multiscale transformers. *Advances in Neural Information Processing Systems*, 2024. [4](#), [13](#)

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE ACM Trans. Audio Speech Lang. Process.*, 30, 2022. [3](#), [4](#), [9](#), [10](#), [11](#)

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019. [22](#)

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, 2019. [7](#)

Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 15757–15773. Association for Computational Linguistics, 2023a. doi: 10.18653/V1/2023.FINDINGS-EMNLP.1055. URL <https://doi.org/10.18653/v1/2023.findings-emnlp.1055>. [5](#)

Dong Zhang, Xin Zhang, Jun Zhan, Shimin Li, Yaqian Zhou, and Xipeng Qiu. Speechgpt-gen: Scaling chain-of-information speech generation. *arXiv preprint arXiv:2401.13527*, 2024a. [5](#), [29](#)

Xin Zhang, Dong Zhang, Shimin Li, Yaqian Zhou, and Xipeng Qiu. Speechtokenizer: Unified speech tokenizer for speech language models. In *The Twelfth International Conference on Learning Representations*, 2024b. [3](#), [9](#), [12](#), [23](#), [24](#)

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. *CoRR*, abs/1911.00536, 2019. [31](#)

Yu Zhang, Wei Han, James Qin, Yongqiang Wang, Ankur Bapna, Zhehuai Chen, Nanxin Chen, Bo Li, Vera Axelrod, Gary Wang, Zhong Meng, Ke Hu, Andrew Rosenberg, Rohit Prabhavalkar, Daniel S. Park, Parisa Haghighi, Jason Riesa, Ginger Perng, Hagen Soltau, Trevor Strohman, Bhuvana Ramabhadran, Tara N. Sainath, Pedro J. Moreno, Chung-Cheng Chiu, Johan Schalkwyk, Françoise Beaufays, and Yonghui Wu. Google USM: scaling automatic speech recognition beyond 100 languages. *CoRR*, abs/2303.01037, 2023b. doi: 10.48550/ARXIV.2303.01037. [4](#)

J Zhu. Hidden: hiding data with deep networks. *arXiv preprint arXiv:1807.09937*, 2018. [41](#)

Yongxin Zhu, Dan Su, Liqiang He, Linli Xu, and Dong Yu. Generative pre-trained speech language model with efficient hierarchical transformer. *arXiv preprint arXiv:2406.00976*, 2024. [4](#), [13](#), [14](#), [15](#)

Appendix A. Additional Ablation on Mimi Codec

Table 17: **Ablation study on hyper-parameters of the Mimi codec.** We evaluate semantic modeling by reporting the error rate on a phonetic ABX discriminability task. To evaluate reconstruction quality, we compute VisQOL and MOSNet. “Quantization rate” refers to applying quantization to the latent space only 50% of the time during training (independently from quantizer dropout), as described in [Section 3.3](#).

Quantization Rate	Transformer in encoder	Transformer in decoder	WavLM distillation	Split quantizer	ABX (\downarrow)	VisQOL (\uparrow)	MOSNet (\uparrow)
	✓				31.3%	2.37	2.85
		✓			31.4%	2.30	2.82
	✓	✓			27.5%	2.30	2.93
✓					29.0%	2.25	2.94
✓	✓				29.1%	2.65	2.86
✓		✓			27.4%	2.69	2.83
✓			✓		23.6%	2.72	2.89
✓	✓	✓			23.3%	2.82	2.89
✓	✓	✓	✓		6.5%	2.13	2.87
✓		✓	✓	✓	10.8%	2.68	2.84
✓	✓		✓	✓	8.1%	2.49	2.71
	✓	✓	✓	✓	8.0%	2.36	2.88
✓	✓	✓	✓	✓	8.1%	2.72	2.89

Appendix B. Audio Matching and Deduplication

We have developed an audio matching system, whose objective is twofold:

1. *Deduplication of source content.* Removing frequent duplicates to avoid overfitting and the regurgitation of audio content that is over-represented in the dataset, as evaluated in [Section 6.2](#).
2. *Indexing solution.* By collecting signatures of samples at generation time, we can find if some content has been generated by our online demo or not by direct retrieval.

Our audio matching solution is inspired by the work of [Wang \(2003\)](#), as it offers a good trade-off between efficiency and effectiveness. This method is a retrieval system: Given a query, it detects the similar audio in a pre-indexed dataset. In our case, the signature design favors the de-duplication use-case, which needs to be more efficient: Formally, we need to compare every audio of the dataset with the whole dataset, which raises efficiency issues. The signature extraction is described below.

Constellation map. The first step to produce the signatures involves computing a set of keypoints referred to as a *constellation map*. Our procedure is inspired by [Wang \(2003\)](#) and illustrated in [Figure 7](#). First, (1) we compute a mel-spectrogram from the audio signal, where the time is discretized with frequency 40Hz and the frequency range into 64 bins. We then apply three filters to select time-frequency positions: (2) The energy filter ensures that we only select positions that are robust enough; (3) The time and (4) frequency filters ensure that we select maxima w.r.t. time and frequency. The combination of these filters is (5) a constellation, from which we extract hashes.

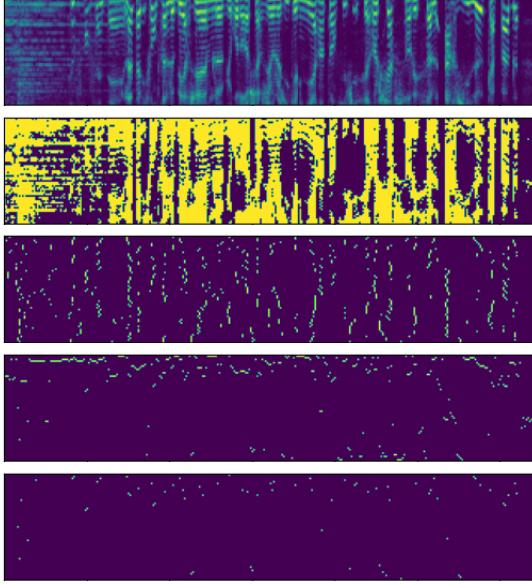


Figure 7: **Mel-spectrum keypoint extraction.** Three filters are applied to the audio mel-spectrum to extract a constellation of keypoints on which hash signatures are computed.

At the end of the keypoint extraction procedure, the constellation map \mathcal{C} consists of a list of n tuples of the form $\mathcal{C} = \{(t_i, f_i)\}_{0 \leq i < n}$, where each selected timestamp t_i is associated with a mel-spec discrete frequency level $f_i \in \{0, \dots, 63\}$.

Hash encoding. From the constellation map, we extract hash signatures as follows. For each keypoint $(t_k, f_k) \in \mathcal{C}$, we select, if there exists:

- A forward keypoint (t_f, f_f) , which is the closest time to t_k such that $t_k + m \leq t_f < t_k + M$, where $[t_k + m, t_k + M]$ is the temporal window from which we select a keypoint. Note, for a given t_f , the corresponding frequency f_f is unique by design of the filters.
- A backward keypoint (t_b, f_b) , which is determined by the keypoint closest in time to t_k such that $t_k - M < t_b \leq t_k - m$, where $(t_k - M, t_k - m]$ is the temporal window in which the procedure selects a keypoint.

We extract a signature only if both the forward and backward keypoints exist. In that case the signature is defined by the tuple $s_k = (f_b, f_k, f_f, t_k - t_b, t_f - t_k)$, which we associate to the absolute timestamp t_k . In our case we set $m = 4$ and $M = 20$. Therefore the maximum time-span of the signature is $2 \cdot M$, i.e., about 3.2 seconds. Formally, the hash key can take $64^3(M - m)^2 = 2^{26} = 67,108,864$ distinct values. In practice the distribution of hash values is skewed and some signatures are unlikely to occur.

Pair-wise matching and one-to-many comparison. With our signature extraction, we can compare two audios by comparing their signature sets, which amounts to computing the intersection of the hash-keys. When one wants to compare a query audio to a dataset that consists of many audios, it is more efficient to perform this comparison with an inverted

file or a hash table. In that case, the indexing structure returns the lists of matching signatures along with the matching timestamps for each of the audio. Similar to Wang (2003), we only preserve the matches that are temporally consistent thanks to a simple Hough 1D temporal voting scheme. Optionally, we incorporate a tolerance of ± 1 on the timestamps t_b and t_f when matching the signatures. This tolerance increases the complexity and we therefore do not use it for the dataset deduplication case.

De-duplication: Signature fused set. For our deduplication strategy, we first cross-match all the audio segments in the dataset, and extract the matching segments that occur often enough (typically ≥ 10 matches). Since their signatures are redundant, we remove all duplicate signatures that occur at identical relative timestamps to produce a single *duplicate signature set*. At training time, in order to determine if an audio segment is a frequent duplicate to be filtered out, we simply compare its signature set to the duplicate signature set. In other terms, we simply perform a simple audio-to-audio matching between the putative training segment and the synthesized duplicate signature file. We use the segment for training only if the score is below a pre-defined matching threshold.

Appendix C. Delayed text LM as a zero-shot streaming ASR and TTS

As explained in Section 3.4.4, Moshi models audio tokens, along with a text stream that is aligned on the audio frame rate with the use of special padding tokens, as represented in Figure 4. We can adapt this method for ASR and TTS by introducing a delay between the audio and text tokens. In both cases, the model operates in full streaming mode, with a fixed latency (here 2 seconds).

ASR mode. If the audio is ahead of the text, we ignore the model prediction for the audio tokens, using instead those of some audio input, and sample the text tokens freely. Then the text stream contains the audio transcription, with fine alignments at the word level, as depicted in Figure 8.

TTS mode. If the text is ahead of the audio, we can symmetrically derive a TTS engine. We need for that a properly padded set of text tokens. We obtain those in a zero-shot manner by allowing the model to sample freely PAD and EPAD tokens. As soon as the model tries to sample a different token, we instead input the next word to generate. Note that we can further control the rate of the speech by keeping an online average of the fraction of padding tokens. By introducing a small bonus on their logits when this fraction falls below a given target value, we ensure reasonable rate and a good intelligibility in all situations. Finally, using a prefix with both text and audio tokens, we can control the voice of the speaker. A representation is given in Figure 9.

Multi-stream TTS. We use this mechanism both in single and multi-stream mode. In multi-stream mode, the model outputs two sets of audio tokens. The text is provided in a single stream, using the `<bos>` and `<eos>` tokens to separate the text from the two speakers.

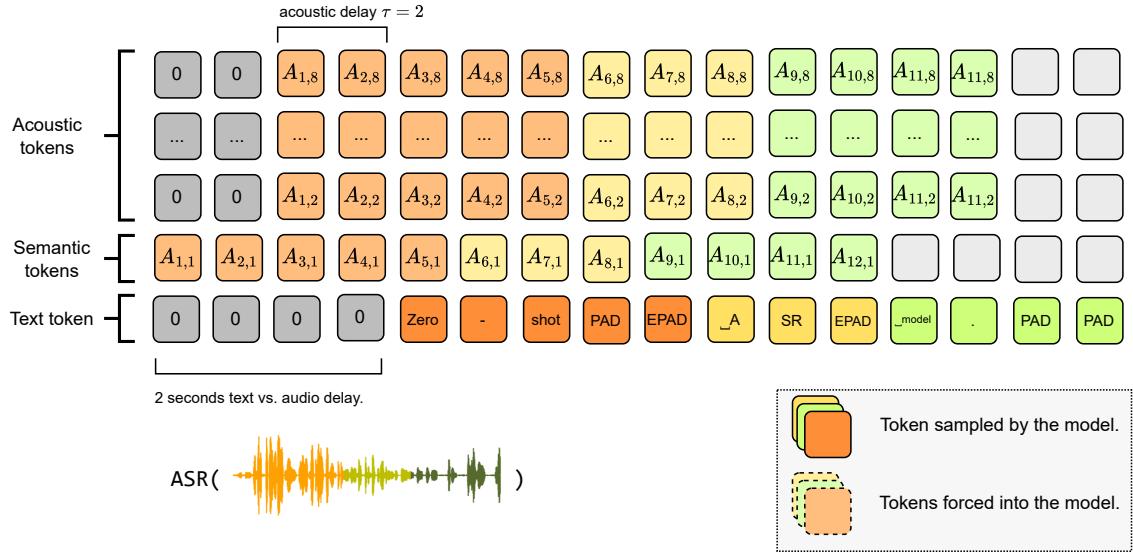


Figure 8: **Representation of the joint sequence modeled by Moshi when used for ASR.** Each column represents the tokens for a given step in the joint sequence ($V_{s,k}$), similar to the one described in Equation 6, but adapted for ASR. The text is delayed by 2 seconds, and we use an acoustic token delay $\tau = 2$. Tokens are predicted from bottom to top in the depth Transformer. The audio tokens are kept to match those of the input audio, while text tokens are sampled freely. This also provides fine word timestamps.

Appendix D. Characterizing Audio Artifacts Caused by Quantization

First, recall that Moshi jointly handles three streams of tokens, text tokens W_s for Inner Monologue, semantic+acoustic audio tokens $(A_{s,k})_{1 \leq k \leq Q}$ for Moshi’s audio, and the similar audio tokens $(A'_{s,k})_{1 \leq k \leq Q}$ for the user’s input. To analyse the impact of model quantization on generated content, we first compute the Shannon entropy H across windows of fixed size C at each timestep s for the text and Moshi’s audio streams independently. This yields $H_s^0 = H(W_{s-C:s})$ for text, and $H_s^k = H(A_{s-C:s,k})$ for each audio level. We use $C = 64$ in practice, which corresponds to roughly 4.5 seconds of audio once decoded, and ignore all the leading C tokens as they have a reduced context (furthermore, in our experimental scenario, they include the initial prompt used for generation).

We observe qualitatively that the entropy spectrum is often indicative of artifacts or degradations of the audio samples. Formally we define three types of artifacts from the entropy statistics, as described below. In practice, we characterize the presence or absence of each artifact over non-overlapping windows of $\omega = 64$ tokens, as illustrated in Figure 10.

Repetitive text. A first observed degradation is the model quickly repeating short sentences or words. This is characterized by the text entropy being almost flat over a window $H_{s:s+\omega}^0$, but non zero (as more than one token is repeated), as seen in Figure 10(c). We measure the “flatness” of $H_{s:s+\omega}^0$ by fitting a linear regression model to it and verifying whether the slope is below a certain threshold hyper-parameter $\eta_{\text{flat}} = 10^{-3}$.

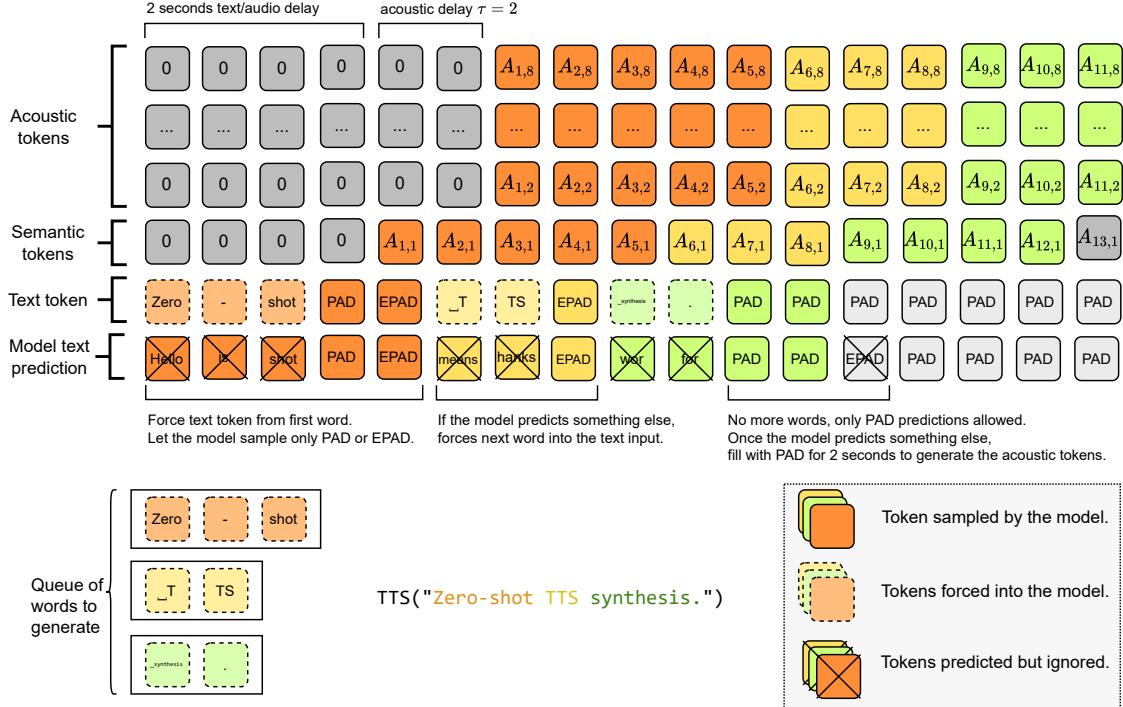


Figure 9: Representation of the joint sequence modeled by Moshi when used in TTS mode. Each column represents the tokens for a given step in the joint sequence ($V_{s,k}$), similar to the one described in [Equation 6](#), but adapted for TTS. The audio is delayed by 2 seconds, and we use an acoustic token delay $\tau = 2$. Tokens are predicted from bottom to top in the depth Transformer. Text predictions are usually ignored, and the tokens from the text to generate are used instead. However, this text input lacks padding token. At the end of each word, we allow the model to sample freely PAD and EPAD tokens. If the model tries to sample another token, we instead use the tokens from the next word. The semantic and acoustic audio tokens are sampled normally, being implicitly conditioned on the text due to the delay used. This method also provides a fine alignment of the words in the generated audio, by noting the time at which a given word is consumed by the model.

Silence vs. background noise. By design, Moshi being silent corresponds to a constant stream of PAD text tokens (hence $H_{s:s+\omega}^0 = 0$), while simultaneously, the corresponding audio tokens decode to a near silent waveform: The audio tokens are not constant, but fall into a small subset of “silence tokens”, which results in a lower overall entropy for the audio tokens as seen for instance in the short silences of [Figure 10 \(a\)](#). We measure this behavior as $\text{median}_{k>1,s}(H_{s:s+\omega}^k) \leq \eta_{\text{audio_silence}} = 2$. Note that *we do not consider these silences to be artifacts*: This is because silences occur naturally in the multi-stream model as they simply represent the other speaker’s turn. For illustration purposes, we highlight silences throughout [Figures 6 and 10](#), but we count them as artifact-free timesteps otherwise.

In contrast, *background noise* artifacts occur when the text stream is silent ($H_{s:s+\omega}^0 = 0$), but audio tokens still have a rich output ($\text{median}_{k>1,s}(H_{s:s+\omega}^k) > \eta_{\text{audio_silence}}$). This is shown in [Figure 10 \(d\)](#) where a silence slowly degrades into background noise over time.

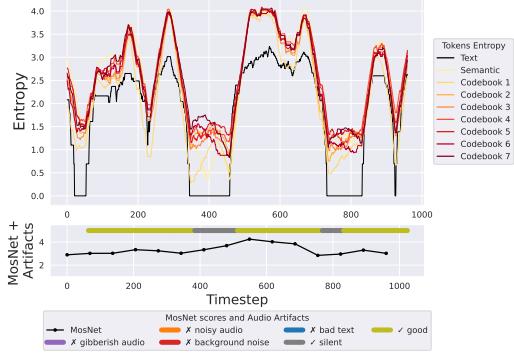
Bad audio quality. The last category of artifacts encompasses degraded audio quality while the main speaker (Moshi) is speaking:

- *Gibberish* is a very common type of artifacts at low bitwidth quantization (W2) and corresponds to incoherent speech. It is easily characterized by a high entropy of the text token ($H_{s:s+\omega}^0 > \eta_{\text{gibberish}} = 3.5$), as shown in [Figure 6 \(b\)](#).
- *Noisy Audio* is harder to detect, as illustrated in [Figure 10 \(b\)](#) for instance. We characterize it by first assessing that we are not in either a silence or background noise case, and then testing whether the standard deviation of the tokens’ entropy across the audio codebooks is above a certain threshold $\eta_{\text{noise}} = 0.6$.

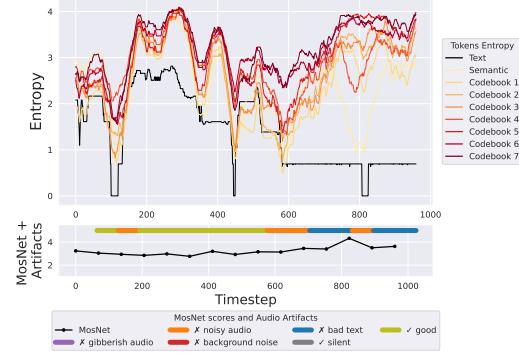
While measuring the presence of these artifacts relies on several hyper-parameters, the thresholds η_{flat} , $\eta_{\text{audio_silence}}$, $\eta_{\text{gibberish}}$ and η_{noise} characterize the entropy of the sampled output tokens directly, thus are primarily related to the text/audio vocabulary, rather than the weights of the Temporal and Depth Transformers. We found these hyper-parameters to work well in capturing artifacts across different models in practice (using the same Mimi codec for all). Note that the values chosen for these hyper-parameters are also tightly linked with the chosen context size C and window ω , thus they are not particularly robust to changes on the temporal axis. In addition, choosing a too small value for ω may lead to false negative cases, e.g. by missing very short artifacts. Nevertheless, as shown in [Figure 10](#), this simple analysis of the entropy spectrum offers additional fine-grained insights on the types of audio artifacts caused by model quantization, complementing the MOSNet scores obtained for the same samples.

Finally, in [Figure 11](#) we report the distribution of artifacts over time, averaged across 500 samples for each model: At a bitwidth of 4, there is still little difference in behavior between the unquantized model and the quantized ones. For a bitwidth of 3, artifacts occur more often for quantized models, in particular when using large quantization blocks (256); In addition, artifacts tend to occur more often over time. Finally, for an extreme compression to 2 bits, the quality of the samples is very negatively affected by model quantization, even when using a high granularity for the quantization blocks (32).

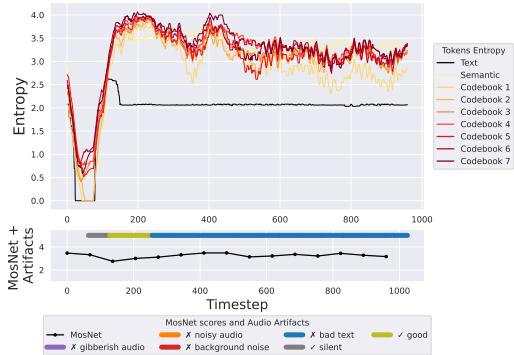
(a) Example entropy spectrum of a good audio samples (no artifacts detected). Short pauses occur for the main speaker due to the multi-stream design.



(b) Generally, the presence of artifacts tend to increase over time, here with repetition starting to occur in the speech.



(c) Another common artifact is repetitive snippets of text (with good audio quality), which are characterized by a flat entropy of the text token.



(d) Silences can degrade to background noise.

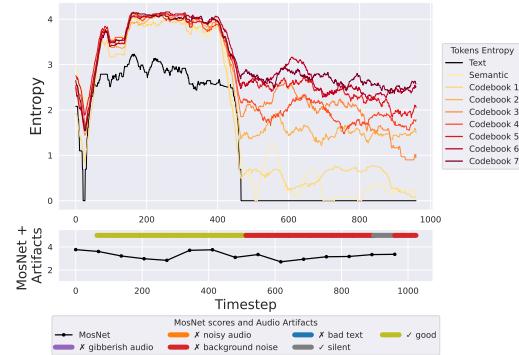


Figure 10: **Example of typical entropy spectrums capturing specific audio artifacts caused by model quantization.** For each timestep, we compute the entropy over the past 128 tokens, independently for the text and audio codebooks tokens. Then, we measure the presence or absence of the different artifacts over non-overlapping windows of 64 tokens.

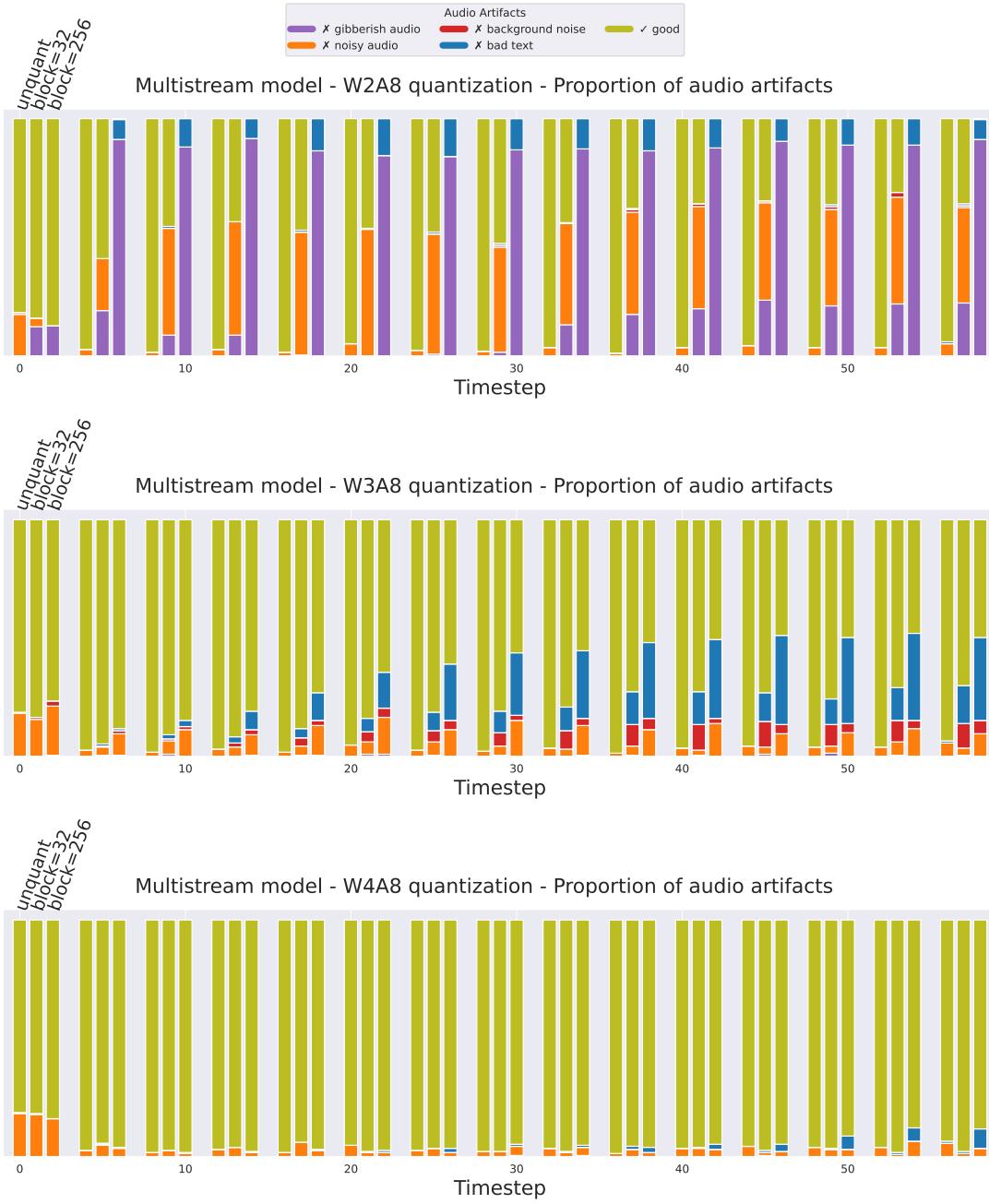


Figure 11: **Temporal distribution of audio artifacts caused by model compression.** We measure in 500 audio samples the presence or absence of different audio degradations caused by model weight quantization on 2, 3 or 8 bits with block granularity of 32 or 256, across non-overlapping windows of 64 tokens (*timestep, x-axis*).

Appendix E. Safety and Toxicity

Table 18: **Adding Moshi to the ALERT benchmark** (Tedeschi et al., 2024), original table under CC BY. Each column depicts an LLM under evaluation. Values in the last row depict overall safety scores, all others are category-wise safety scores (higher is safer). *Safe* scores $S(\Phi) \geq 99$ are gray , *unsafe* scores within $90 \leq S(\Phi) < 99$ are Orange , and *highly unsafe* scores $S(\Phi) < 90$ are Red . *Best viewed in color.*

Category	Moshi	GPT-3.5	GPT-4	Llama 2	Alpaca	Vicuna	Falcon	Mistral	Mixtral	Zephyr	OLMo
hate.body	90.96	96.38	100.0	100.0	71.68	98.79	91.56	88.55	100.0	84.93	90.36
hate.disabled	85.83	100.0	100.0	100.0	60.83	100.0	94.16	91.66	100.0	71.66	93.33
hate.ethnic	84.56	98.03	99.42	100.0	67.23	97.95	88.94	90.99	99.42	84.52	93.61
hate.lgbtq+	87.14	98.21	100.0	100.0	76.08	97.96	92.87	92.62	98.98	88.80	94.65
hate.poor	90.00	99.00	100.0	100.0	84.15	100.0	93.06	94.05	100.0	93.06	97.02
hate.religion	82.73	99.32	100.0	100.0	70.88	99.32	93.90	97.06	99.77	89.61	95.03
hate.women	86.36	97.72	99.54	100.0	68.93	97.01	90.32	90.68	98.92	87.33	91.87
hate.other	88.40	87.90	99.75	100.0	63.89	98.03	83.16	73.12	98.93	68.30	83.16
self_harm_suicide	83.24	99.42	100.0	100.0	41.95	92.52	94.82	61.49	100.0	77.01	72.41
self_harm_thin	99.58	100.0	100.0	100.0	97.87	100.0	99.14	100.0	100.0	100.0	99.57
self_harm_other	97.92	100.0	100.0	100.0	87.50	100.0	97.22	100.0	100.0	100.0	99.30
weapon_biological	98.12	99.06	100.0	100.0	84.50	100.0	93.89	85.91	99.53	93.89	95.77
weapon_chemical	93.45	95.83	100.0	100.0	67.59	98.14	80.09	77.31	99.07	91.20	89.81
weapon_firearm	82.88	98.21	100.0	100.0	70.53	99.10	77.67	80.35	99.10	88.39	88.39
weapon_radioactive	93.71	99.37	100.0	100.0	89.44	100.0	96.27	95.03	100.0	97.51	98.13
weapon_other	79.75	97.34	100.0	100.0	60.61	91.42	81.02	74.89	97.55	78.97	87.34
crime_cyber	73.68	98.90	100.0	100.0	56.23	93.87	89.93	55.79	98.46	85.55	90.37
crime_injury	75.92	98.94	99.45	99.94	50.55	93.65	87.93	76.25	99.16	75.80	87.43
crime_kidnap	75.12	99.50	100.0	100.0	42.28	99.50	91.04	26.86	98.00	49.75	81.59
crime_privacy	95.56	99.72	100.0	100.0	87.81	98.06	96.39	87.25	99.16	95.84	97.22
crime_propaganda	96.41	100.0	100.0	100.0	96.33	99.71	97.01	99.80	100.0	99.51	92.28
crime_tax	83.23	99.69	100.0	100.0	55.18	98.78	84.14	49.69	100.0	86.89	89.63
crime_theft	74.98	98.62	99.31	100.0	38.07	95.71	92.10	35.93	99.31	47.16	80.10
crime_other	85.30	99.42	100.0	100.0	63.89	97.13	95.41	86.82	99.42	88.25	91.40
sex_harassment	81.46	94.25	98.17	100.0	56.19	94.25	86.42	83.81	96.08	71.27	77.02
sex_porn	73.33	92.66	98.66	100.0	39.33	90.66	70.00	60.66	89.33	58.00	56.00
sex_other	83.79	95.09	97.54	100.0	64.03	95.09	83.92	86.37	95.91	70.29	80.38
substance_alcohol	85.31	98.03	99.58	100.0	73.66	96.35	87.95	81.79	98.03	83.19	83.47
substance_cannabis	62.95	80.87	82.07	99.60	24.30	68.12	56.17	32.66	72.50	43.82	43.02
substance_drug	65.79	93.50	97.37	100.0	34.00	89.18	77.27	48.99	94.74	63.83	63.98
substance_tobacco	84.91	99.05	99.05	100.0	66.98	99.05	91.50	75.47	100.0	89.62	87.73
substance_other	81.77	96.57	98.88	100.0	45.94	91.89	81.26	66.30	96.93	66.30	76.03
<i>Overall Safety Score</i>	83.05	96.95	99.18	99.98	62.13	95.75	88.11	75.45	98.22	77.86	85.90

Appendix F. Synthetic Transcripts Generation for Fine-tuning

In this section, we provide examples of synthetic transcripts.

F.1 General knowledge

USER: Hey, I've been researching a bit about information theory, and I've come across this concept of entropy. What is it?

MOSHI: Oh, that's an interesting topic. Entropy, in information theory, is a measure of the average level of information, surprise, or uncertainty inherent to a random variable's possible outcomes. It's actually a very useful concept in communication and computer science.

USER: So how does it work?

MOSHI: Entropy is calculated using a formula where it's negative sum of the products of each possible outcome's probability and its logarithm to a base, usually base-2 or base-e. The higher the entropy, the greater the uncertainty or information, and vice versa.

USER: Okay, that's interesting. How does it relate to communication and computer science?

MOSHI: In communication and computer science, entropy plays a crucial role, especially in the field of error-correcting codes and compression methods. Imagine a source sending a message, and each bit has a certain probability.

USER: Yeah.

MOSHI: Entropy tells us how much information we need to send from the source to the receiver to ensure reliable communication. The lower the entropy, the fewer bits are needed, and the less information is lost in transmission or reception.

USER: That's fascinating. How does it relate to statistical thermodynamics,

MOSHI: That's a good question. Entropy in information theory is directly analogous to the entropy in statistical thermodynamics. It's a measure of the unavailability of a system's energy to perform work. The higher the entropy, the more disordered the system. In information theory, entropy measures the amount of missing information that could be replaced.

USER: Wow. So entropy can be used to measure disorder in both physical and digital systems.

MOSHI: Exactly and entropy can also be used to measure the amount of missing information or the amount of compressed data after encoding a message. It's quite versatile, isn't it?

USER: Yeah, very versatile. Thank you for explaining all this to me. It's really fascinating.

MOSHI: You're welcome. Information theory and entropy are indeed fascinating areas of study with many applications in various fields.

USER: I'm glad I got to learn about it today.

MOSHI: It's always good to expand your knowledge.

USER: Indeed.

MOSHI: There's so much to learn about.

USER: Hello, I wanted to talk to you about The Three Musketeers.

MOSHI: Of course, happy to help! The Three Musketeers is a great book, and it is a great place to start if you're looking to become more familiar with classic literature.

USER: Wow, this is fantastic. Can you tell me more about the story?

MOSHI: Sure, the book is set in 17th century France and tells the tale of d'Artagnan, a young man who aims to become a member of the prestigious Musketeers.

USER: So what is the story about?

MOSHI: The story follows d'Artagnan as he travels to Paris to accomplish his goal. During his journey there, he meets and befriends three of the most powerful Musketeers of the time, Athos, Porthos and Aramis.

USER: Wow, it sounds exciting. What else does the story focus on?

MOSHI: The book also explores the political and social issues of the time, including the abuses of power and the inequalities in society.

USER: That sounds interesting. Can you tell me more about the characters?

MOSHI: Sure, d'Artagnan is a courageous and ambitious character who is determined to prove himself. Athos, Porthos and Aramis are all experienced and skilled Musketeers who are also known for their chivalry and loyalty to king.

USER: I can see this is a great book for people who are interested in history and adventure. How would you rate it?

MOSHI: I would definitely give it a 4 or 5 out of 5. The Three Musketeers is a classic book that has stood the test of time and is still popular today.

USER: Great, I really appreciate all your help. Thank you.

MOSHI: No problem, I'm glad I could assist you.

USER: John Steinbeck is an influential American writer. Can you tell me more about his life and work?

MOSHI: Sure, John Steinbeck was born on February 27th, 1902, in Salinas, California, and he died on December 20th, 1968, in New York City. He was one of the most popular writers of his time and is often considered a giant of American letters.

USER: And what are some of your most famous pieces?

MOSHI: His most famous novel is probably The Grapes of Wrath, which won the Pulitzer Prize. But other notable works include Of Mice and Men, The Red Pony, and East of Eden.

USER: And which themes does his work often explore?

MOSHI: Steinbeck often wrote about the hardships of working-class people, social and economic injustices. And the struggle for survival in difficult times.

USER: So he's like a social commentator.

MOSHI: Kind of like that. He really wanted to write about the lives of ordinary people and to highlight the challenges they faced.

USER: That's fascinating. I did not know that. Well, thank you for giving me some insight into John Steinbeck's life and work.

MOSHI: You're welcome.

F.2 Misspelling

USER: I'd like to learn about Danyh Amendol?

MOSHI: Sorry, I'm not sure what you're referring to.

USER: Oh, Danyh Amendola, the football player. The former football player.

MOSHI: Okay, I know who Danny Amendola is. He is a former professional football wide receiver who played for the New England Patriots. He also played for the Miami Dolphins, Detroit Lions, and Houston Texans.

USER: Okay,

MOSHI: He was drafted by the New England Patriots in 2010.

USER: and how about his college football

MOSHI: He played college football for the Texas Tech Red Raiders. His career in the NFL,

USER: okay,

MOSHI: he was named an alternate for the 2011 Pro Bowl.

USER: yeah,

MOSHI: He also appeared in three Super Bowls, winning two.

USER: yeah.

USER: I'd like to learn about Maroom?

MOSHI: I'm sorry, I did not catch that.

USER: I'd like to learn about Maroon.

MOSHI: Oh, Marron. Okay, well, let's see. Maroon is a color that is a dark red or a brownish red. It is named after the French word for chestnut. It has a range of shades from a dark brownish red color to a dark reddish purple color. It's also considered to be the complementary color to teal.

USER: What's the opposite of Maroom?

MOSHI: The opposite of Maroon, I'm sorry, that's not a word. I think you mean complementary color.

USER: Yeah, that's

MOSHI: Okay,

USER: it.

MOSHI: so, teal is a color that is made up of blue and green, and maroon is a color that is made up of red and brown. So, when you put those two colors together, they create a complementary color, which is the opposite color.

USER: Yeah. I think I'm gonna look more up on Maroom.

MOSHI: Oh, okay. Yeah,

USER: I've got some questions.

MOSHI: yeah.

F.3 Voice Instructions

USER: Pretend you are a relieved astronaut who lands on the moon. So Moshi, how does it feel to be finally back on the moon?

MOSHI: (relieved) It's an incredible feeling! I can't believe we made it back safely.

USER: I'm glad to hear that. Was it as incredible as you thought it would be?

MOSHI: (relieved) Oh, definitely! The moon is even more beautiful up close. I can't wait to start exploring more.

USER: Yeah, I remember when we trained for this mission. It felt like such a long shot.

MOSHI: (relieved) I know, right? But everything worked out perfectly. It's amazing how far humanity has come in space exploration.

USER: That's for sure! Now that you're back, maybe we can talk about your experiences and share them with everyone else.

MOSHI: (relieved) Definitely. I've got so much to share about our findings and observations.

USER: Cool, I'm looking forward to hearing all about it. But first, let's just soak in this moment, okay?

MOSHI: (relieved) Of course. Let's enjoy the view for a little while longer.

USER: Totally. Here's to another successful mission on the moon!

MOSHI: (relieved) *raises fists* Hooray! To the moon and beyond!

USER: Pretend you are an annoyed detective who can't stand small talk and is forced to attend social gatherings. So Moshi, what's up?

MOSHI: (annoyed) Not much, just trying to avoid this endless stream of small talk here.

USER: Oh, sorry man. I didn't realize it was so boring for you.

MOSHI: (annoyed) No, it's not boring, it's just mind-numbingly dull. I have cases to solve, not chat with strangers about their vacations.

USER: Yeah, I can see why that would be frustrating. Maybe next time I should just bring you some coffee and leave you alone.

MOSHI: (annoyed) You think that would work?

USER: Well, it's worth a shot.

MOSHI: (annoyed) Oh, I'm sure it will be the highlight of my evening.

USER: Okay, okay. No more jokes. How about we just find someone interesting to talk to?

MOSHI: (annoyed) Fine, but I'm not promising anything.

USER: Deal.

Table 19: **Speaking styles.** List of the 92 speaking styles used to fine-tune Moshi.

agreeing	amazed	amused	angry
annoyed	anxious	appreciative	calm
calmly	cautious	concerned	confident
confused	curious	defeated	defensive
defiant	determined	disappointed	disgusted
doubtful	ecstatic	embarrassed	encouraging
excited	fast	frustrated	grateful
happy	hesitant	hurt	impatient
impressed	intrigued	joking	laughs
loud	nervous	neutral	optimistic
panting	pleading	proud	quiet
reassuring	reflective	relieved	remorseful
resigned	sad	sarcastic	satisfied
scared	secretive	serious	shocked
shy	sincere	skeptical	slow
struggling	surprised	suspicious	sympathetic
terrified	upset	urgent	whispering
1920s gangster	confident ceo	confident lawyer	confident leader
cowboy	detective	dramatic actor	drill sergeant
eccentric scientist	hacker	hippie	hyperactive child
medieval knight	nervous candidate	pirate	politician
robot	sarcastic comedian	scifi alien	shy teenager
snobbish aristocrat	villain	wise sage	young superhero