RSTnet: Real-time Speech-Text Foundation Model Toolkit

RSTnet teams
dcyang@se.cuhk.edu.hk

Abstract

Building a real-time speech-text foundation model capable of understanding and generating speech has garnered significant attention. Notable examples of such models include ChatGPT-4o and Moshi. However, challenges in training these models continue to persist in the research community. This technical report introduces RSTnet, a new open-source platform designed for developing real-time speech-text foundation models. RSTnet offers a comprehensive framework for data processing, pre-training, and fine-tuning, aimed at helping researchers build their real-time speech-text models efficiently. It builds upon previous works, such as the real-time spoken dialogue model (Moshi) and the universal audio generation model (UniAudio). RSTnet consists of two key components: (1) streaming audio codec models, and (2) speech-text foundation models. In this report, we provide detailed information on the training and evaluation procedures for both components. Additionally, we offer a brief review of real-time speech-text foundation models, summarizing prior works, analyzing their strengths and weaknesses, and suggesting potential research directions. The project can be found at: https://github.com/yangdongchao/RSTnet

1 Introduction

Large language models (LLMs) (*e.g.* GPT4 [1], LLAMA [31]) become more versatile and effective at handling diverse and complex Natural Language Processing (NLP) tasks after scaling their model and training data. The success of LLMs has inspired us to develop multi-modal LLMs that can understand and generate speech in real-time. Notably, recent advancements such as GPT-4o and Moshi [8] have demonstrated the feasibility of training real-time speech-text foundation models. Although GPT-4o has released its real-time speech chat API, it is a closed model with limited technical details available. Moshi is the first real-time full-duplex spoken large language model that has presented its technical reports and released its models. However, training such models poses significant challenges in the research community. One notable challenge is the lack of comprehensive training and fine-tuning frameworks. To address this gap, we present a toolkit called RSTnet, which serves as a real-time speech-text foundation model. RSTnet is built upon the foundation of previous codec-based speech language models like Moshi and UniAudio. It incorporates a streaming audio codec that effectively encodes speech data into speech tokens. These speech tokens are then further modeled using an auto-regressive (AR) based language model.

In this report, we delve into how RSTnet is designed, trained, and evaluated. Detailed recipes are provided for both the streaming audio codec models and the speech-text foundation models, covering data preparation, model architectures, and performance evaluation metrics. By outlining the training process and providing clear guidelines, we aim to assist researchers utilizing RSTnet in building their own real-time speech-text models.

Furthermore, we also provide a comprehensive review of prior work in the field of real-time speech-text foundation models. This includes a summary of existing approaches, an analysis of their

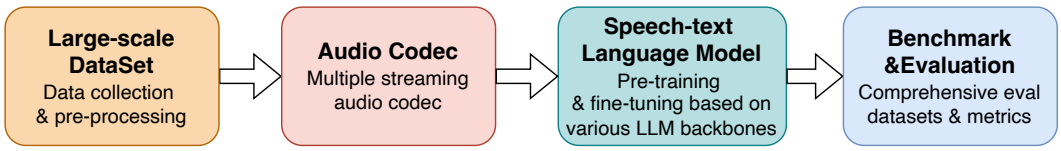

Figure 1: The overview of our RSTnet, which includes four main parts: (1) Dataset collection and pre-processing; (2) Streaming audio codec; (3) Real-time speech-text language model; (4) Benchmark and evaluation tools.

strengths and limitations, and a discussion of potential research directions. As real-time AI continues to progress, we believe that RSTnet serves as a valuable tool for pushing the boundaries of what speech-text models can achieve. It facilitates further research and exploration in this domain, opening up new possibilities for real-time speech understanding and generation.

2 RSTnet

2.1 Overview

Figure 1 shows the overview of our RSTnet. In the following, we give the details of the training framework for streaming audio codec and speech-text foundation models.

2.2 Data Collection Pipeline

The data collection module of RSTnet is designed to collect the training data for the speech-text language model pre-training and fine-tuning. Specifically, we mainly focus on unsupervised single-stream audio datasets, unsupervised multi-stream audio datasets, and speech-text instruction data for instruct tuning.

- **Unsupervised single-stream audio dataset**
- **Unsupervised multi-stream audio dataset**
- **Speech-text instruct data for instruct tuning**

2.3 Streaming Audio Codec Training Recipes

Although there are many existing audio codec models, most of them are not streaming. One of the reasons is that they do not use the causal convolutional layers or causal transformer layers, e.g. DAC [18] and FACodec [13]. Encodec [7] and MimiCodec [8] are typical streaming audio codecs. Encodec consists of the causal convolution layers and LSTM layers, and MimiCodec adopts the causal convolution layers and causal transformer layers. Considering the importance of streaming codecs for real-time speech dialogue, we focus on building a streaming audio codec framework. We follow the open-source model structure from MimiCodec and Encodec and build a training and inference framework based on our codebase. Our software architecture includes:

- **Data Preparation:** The streaming codec training only needs the audio-only data. Thus, we only need to prepare a *scp* file that saves the audio path. We prepare the corresponding script to obtain the scp files.
- **Dataloader:** We adopt a simple torch dataloader for the audio codec training. The data loader needs to sample the audio signal with 24khz and 16khz, where the 16khz audio signal is used to extract semantic features from self-supervised learning models, such as WavLM, Hubert, W2Vec-bert, and so on. Following Encodec and MimiCodec, we try to compress the 24khz audio.
- **Streaming encoder-decoder backbone:** We follow the structure of MimiCodec, compressing the 1-second 24khz audio data into 12.5 frames with the causal convolution and transformer layers. We also expect users can use different down-sampling steps to train their audio codec.

- **Quantizer:** In this stage, we follow MimiCodec to use the RVQ-based quantizer. We default to use 8 codebooks (one for semantic and 7 for acoustic information). In the future, we also plan to support scalar quantization.
- **Discriminators** we provide three different discriminators: MultiFrequencyDiscriminator, MultiPeriodDiscriminator, and MultiScaleDiscriminator. In practice, we suggest using any one of them or combining all of them.
- **Semantic teacher models:** Following MimiCodec, we use the pre-trained audio model to extract semantic features, these features can be used as additional supervised labels to force the first VQ layer of audio codec to encode semantic information. We provide four different pre-trained models to extract semantic features, including WavLM, Hubert, Whisper, and Wav2Vec-Bert. Users can set different types of models when training their audio codec models.
- **Training Loss** For the training loss, we follow the details of MimiCodec, including (1) Removing reconstruction loss; (2) not applying quantization with a certain probability during training; (3) Semantic guidance for the first VQ layer.
- **Evaluation:** We also provide the 8 evaluation metrics, including SSIM, PESQ, STOI, MS-STFT-Loss, SI-SNR, MCD, VISQOL, and DNSMOS.

We split the whole streaming audio codec software into 4 stages: (1) data preparation; (2) Model training; (3) Model inference; and (4) Model evaluation. We provide complete recipes to perform these four stages, which are written in the bash scripts by following the Kaldi manner.

2.4 Real-time Speech-Text Foundation Model

For the real-time speech-text language model, we follow the setting of Moshi, which uses a multi-scale transformer [40, 42] to model multiple audio codec codebook layers. The training of Moshi includes four stages:

- **(1) Text language model pre-training:** It follows the common recipes of LLM training, such as GPT3 [1] and LLaMA 2 [31]. In our view, this stage can be omitted, we can directly use open-sourced LLMs backbone. We give the details in the following.
- **(2) Single-stream speech-text multi-scale language model pre-training:** In this stage, we need to collect large-scale speech data and text data. During training, we can randomly choose audio-only data or text-only data as a batch. If we choose text-only data, we only calculate the loss from the global transformer. Instead, if we choose audio-only data, we can calculate the loss from the global transformer and local transformer. Note that, for the unlabeled audio data, we need to use whisper tools to obtain the duration information for each word. The details can be found in the data preparation part.
- **(3) Multi-stream speech-text multi-scale language model post-training:** In this stage, we add two streaming training: user streaming and client streaming. To construct such training data, we follow Moshi to use PyAnnote [27] to diarize the audio from the unsupervised audio dataset: if a speech data includes two speakers, we can randomly choose one as a user, the other as the client speaker.
- **(4) Fine-tuning with high-quality conversation data:** Building a good conversation dataset to fine-tune the pre-trained model.

Considering the fact that pre-training a speech-text foundation model from scratch will cost a lot of data and GPU resources. We implement our software from two aspects: (1) Training from scratch. (2) Fine-tuning pre-trained Moshi.

We first give the details of fine-tuning pre-trained Moshi. The pre-training and post-training can be easily implemented using a similar way. As Figure 2 shows, the experimental flow of fine-tuning recipe includes three stages: Data Preparation, Audio tokenization, and speech-text language model training.

2.4.1 Data Preparation for multi-stream conversation data

The data preparation module is designed to convert four types of source data into three paradigms. Specifically, four types of source data refer to unsupervised single-stream audio and text datasets

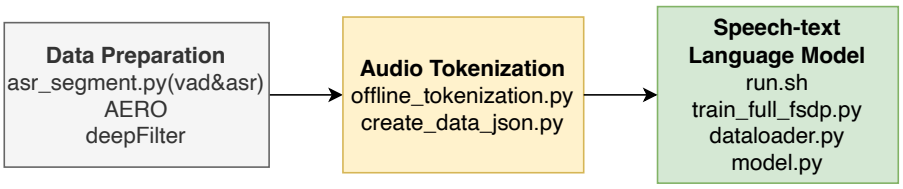

Figure 2: The experimental flow of speech-text foundation model training recipe.

for pre-training and post-training, unsupervised multi-stream audio datasets for fine-tuning, and speech-text instruct data for instruct tuning. Three paradigms refer to single-stream text tokens, single-stream codes, and multi-stream codecs. We now describe the pre-processing pipeline for each type of source data respectively.

- **(1) Unsupervised single-stream audio dataset** need to be converted to single-stream codecs for pre-training and to multi-stream codecs for post-training. We first use PyAnnote [27] to diarize the audio from the unsupervised audio dataset. Then we conduct Voice Activation Detection (VAD) with PyAnnote [27] and segment original audios into segments shorter than 105 seconds. For single-stream codecs, we transcribe each segment with Whisper [28], whereas for multi-stream codecs, we first split the speech into two channels according to diarization result and then transcribe each channel respectively. To obtain word-level timestamps for alignment, we adopt WhisperX [3], an enhanced version of Whisper [28].
- **(2) Unsupervised single-stream text dataset** need to be converted to single-stream text tokens for pre-training. This can be done with SentencePiece [17] tokenizer.
- **(3) Unsupervised multi-stream audio dataset** need to be converted to multi-stream codecs for fine-tuning. We first conduct VAD and ASR as in (1). It should be noticed that in VAD, both channels are regarded as a whole, whereas ASR is conducted in each channel respectively. In order to improve speech quality, we further apply audio super-resolution with AERO [23], and reduce the noise with Deepfilternet [29].
- **(4) Speech-text instruct data for instruct tuning** need to be converted to multi-stream codecs for instruct tuning. The utterances in each turn is first concatenated into segments of less than 100 seconds, where each channel contains utterances of a particular speaker. Then, we utilize WhisperX [3] to get transcriptions with timestamps.

2.4.2 Audio Tokenization

In this section, we describe how text and audio are tokenized and aligned with the help of word-level timestamps obtained in Section 2.4.1. We adopt SentencePiece [17] to tokenize text whereas MimiCodec (See Section 2.3) to tokenize audio. The text stream is initialized with <pad> token, and subword tokens of each word in inserted into a slot, beginning at the start timestamp of the word. Following [8], a <epad> token is inserted before the start token of each word. In RSTnet, the <epad> token is implemented as <unk>. After alignment, the text tokens, the semantic tokens, and the acoustic tokens of the same audio segment are stacked as a single tensor for the dataloader to unpack.

Additionally, the dataloader applies a delay pattern when loading the codec codes. Specifically, the acoustic tokens are shifted back for a certain step size following [8]. If the codec codes contain multiple streams, the acoustic tokens of each stream are shifted simultaneously.

2.4.3 Speech-text Language model fine-tuning

After finishing the data preparation and audio tokenization stages, you can begin the model fine-tuning. Fine-tuning pre-trained Moshi is very easy, you only need to care about three parts:

- **Dataloader:** We design a dynamic batch dataloader, and we can set the maximum batch tokens for each GPU. Furthermore, we sort the data based on their duration, thus we can build a batch with similar-length audio.
- **Trainer:** The trainer function includes four steps: (1) load the pre-trained Moshi; (2) wrap it with FSDP; (3) set optimizer and schedule; (4) calculate loss and save checkpoints.

- **Model:** We implement the forward function based on the training principle of the multi-streaming speech-text language model.

2.4.4 Speech-text Language model pre-training and post-training

3 Related works

Audio Codec Models Recently, advancements in neural network-based audio codecs have led to several promising developments [43, 7, 39, 18, 13, 8]. These systems typically follow an encoder-quantizer-decoder framework: the encoder first extracts deep features from audio data, then the features are quantized by the quantizer, and a decoder is used to recover the audio from quantized features. Audio codec models have been widely used to build audio language models [40, 4, 15].

Nowadays, the research of audio codec models has been significantly influenced by the application of audio language models. In the early phase, a lot of works tried to build a high-fidelity neural audio compression model, such as SoundStream [43], Encodec [7], and DAC-Codec [18]. Some early studies [4] find that the audio codec model can be used as the tokenizer for audio language models. However, previous audio codec models introduce multiple codebooks to maintain a good reconstruction performance. The multiple codebooks characteristic brings new challenges for language models, e.g. long sequence problems and slow generation speed. To overcome this issue, the researcher begins to explore low-bit audio codecs, such as HiFi-Codec [39], Single-Codec [19], WavTokenizer [12], BigCodec [37], and so on. Although these works claim that they can compress the audio data with low-bit, their reconstruction performance is still poor than previous multiple codebooks-based codec models. Furthermore, these works cannot be viewed as streaming audio codecs, due to their introduced non-casual structure.

Another research line is that considering the feature factorization during the audio codec models training, such as SpeechTokenizer [44], SemanticCodec [22], FACodec [13], LLM-Codec [38], and so on. One of the reasons is that previous works [4] have shown that language models can effectively model semantic information. Similarly, these works also can not be viewed as streaming audio codecs. Instead, MiMiCodec [8] is a typically streaming codec, making it suitable for applications that require low-latency speech processing.

Multimodal Large Language Models Recently, there has been tremendous progress in the area of multimodal LLMs. These works can be divided into several categories: (1) Empowering the LLMs to understand audio, e.g. some works use pre-trained LLMs as the base and incorporate additional input modalities, such as vision [45, 21, 41, 46, 20, 32] and audio [5, 16, 10, 30, 11, 33]. In general, these works consist of a pre-trained LLM, a pre-trained vision/audio encoder, and a modality adaptor. (2) Using LLM training paradigm to perform audio generation tasks, such as TTS [34, 15], music generation [2, 6], universal audio generation [40], and so on. (3) Empowering LLMs to understand and generate audio in an end-to-end style, typical examples being GPT4-o and Moshi. In this report, we focus on building streaming end-to-end speech-text language models.

Some works claim that they can understand and generate speech in streaming style, such as Mini-Omni [36], LLAMA-Omni [9] and PSLM [24]. In essence, Mini-Omni and LLAMA-Omni both try to combine a streaming TTS module with existing LLMs. Mini-Omini solves the text and audio token alignment by introducing an additional transformer module (following MusicGen [6]). LLAMA-Omni solves the text and audio token alignment by introducing CTC loss. PSLM [24] proposes generating speech and text tokens in parallel to receive low latency for spoken dialogue systems. However, these works cannot support full-duplex streaming between the user and the system. Joint training between LLMs and generative model has been widely explored in both audio [11] and image [35] generation. Such a strategy has been validated as an effective way to build multi-modal LLMs. One of the drawbacks is that such a strategy tends to align the text information from LLM and speech tokens, without considering the inner connection between speech and text modalities. For example, non-text information in the speech is easy to ignore during the first alignment training stage. Furthermore, the newly introduced TTS module may have unstable performance, just like previous works [34].

4 Discussion

4.1 What makes a good audio codec for speech-text foundation models?

Moshi introduces a fresh perspective on audio codecs, emphasizing a high compression ratio in the time dimension. For example, MimiCodec compresses 1-second 24kHz audio into 12.5 frames. This design suits speech-text foundation models by reducing the information gap between text and speech tokens. To address information loss, MimiCodec leverages large-scale transformers and multiple codebooks, along with knowledge distillation to ensure the first layer encodes more semantic information. However, the question remains: is MimiCodec the ideal codec? We may consider the following:

- **Single VQ layer or RVQ?**
- **Is forcing the first VQ layer to encode semantic information effectively?**
- **Does a conflict exist between the goals of audio codec (reconstruction) and LLMs (generation)?**

4.2 Are there alternative speech-text foundation model approaches?

Moshi offers a comprehensive solution by integrating text, semantic, and acoustic tokens. Yet, we can refer to another approach, such as modeling only text and semantic tokens [26, 25, 14]. dGSLM [25] and Spirit-LM [26] have demonstrated the viability of semantic token modeling using a speech-text interleaving framework. After predicting semantic tokens, a vocoder such as BigVGAN or a diffusion-based model can be used for audio reconstruction. The challenge lies in making the vocoder streaming-capable. In the future, we plan to implement this alternative by modeling text and semantics via LLMs and using a vocoder for audio recovery.

5 Conclusion

In this report, we present a new open-source platform for real-time speech-text foundation models named RSTnet. Specifically, our RSTnet includes the completed training recipes for streaming audio codec and real-time speech-text foundation models. Furthermore, we also give a simple review about this research direction, and give some discussion for the potential improvement direction and research problems.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

[3] Max Bain, Jaesung Huh, Tengda Han, and Andrew Zisserman. Whisperx: Time-accurate speech transcription of long-form audio. *INTERSPEECH 2023*, 2023.

[4] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, et al. Audiolm: a language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[5] Zhehuai Chen, He Huang, Andrei Andrusenko, Oleksii Hrinchuk, Krishna C Puvvada, Jason Li, Subhankar Ghosh, Jagadeesh Balam, and Boris Ginsburg. Salm: Speech-augmented language model with in-context learning for speech recognition and translation. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 13521–13525. IEEE, 2024.

[6] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[7] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

[8] Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue.

[9] Qingkai Fang, Shoutao Guo, Yan Zhou, Zhengrui Ma, Shaolei Zhang, and Yang Feng. Llama-omni: Seamless speech interaction with large language models. *arXiv preprint arXiv:2409.06666*, 2024.

[10] Shujie Hu, Long Zhou, Shujie Liu, Sanyuan Chen, Hongkun Hao, Jing Pan, Xunying Liu, Jinyu Li, Sunit Sivasankaran, Linquan Liu, et al. Wavllm: Towards robust and adaptive speech large language model. *arXiv preprint arXiv:2404.00656*, 2024.

[11] Atin Sakkeer Hussain, Shansong Liu, Chenshuo Sun, and Ying Shan. Mugen: Multi-modal music understanding and generation with the power of large language models. *arXiv preprint arXiv:2311.11255*, 2023.

[12] Shengpeng Ji, Ziyue Jiang, Xize Cheng, Yifu Chen, Minghui Fang, Jialong Zuo, Qian Yang, Ruiqi Li, Ziang Zhang, Xiaoda Yang, et al. Wavtokenizer: an efficient acoustic discrete codec tokenizer for audio language modeling. *arXiv preprint arXiv:2408.16532*, 2024.

[13] 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.

[14] Eugene Kharitonov, Ann Lee, Adam Polyak, Yossi Adi, Jade Copet, Kushal Lakhotia, Tu-Anh Nguyen, Morgane Rivière, Abdelrahman Mohamed, Emmanuel Dupoux, et al. Text-free prosody-aware generative spoken language modeling. *arXiv preprint arXiv:2109.03264*, 2021.

[15] 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. *Transactions of the Association for Computational Linguistics*, 11:1703–1718, 2023.

[16] Zhifeng Kong, Arushi Goel, Rohan Badlani, Wei Ping, Rafael Valle, and Bryan Catanzaro. Audio flamingo: A novel audio language model with few-shot learning and dialogue abilities. *arXiv preprint arXiv:2402.01831*, 2024.

[17] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Jan 2018.

[18] Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. *Advances in Neural Information Processing Systems*, 36, 2024.

[19] Hanzhao Li, Liumeng Xue, Haohan Guo, Xinfa Zhu, Yuanjun Lv, Lei Xie, Yunlin Chen, Hao Yin, and Zhifei Li. Single-codec: Single-codebook speech codec towards high-performance speech generation. *arXiv preprint arXiv:2406.07422*, 2024.

[20] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[21] Hao Liu, Wilson Yan, and Pieter Abbeel. Language quantized autoencoders: Towards unsupervised text-image alignment. *Advances in Neural Information Processing Systems*, 36, 2024.

[22] 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] Moshe Mandel, Or Tal, and Yossi Adi. Aero: Audio super resolution in the spectral domain. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[24] Kentaro Mitsui, Koh Mitsuda, Toshiaki Wakatsuki, Yukiya Hono, and Kei Sawada. Pslm: Parallel generation of text and speech with llms for low-latency spoken dialogue systems, 2024.

[25] Tu Anh Nguyen, Eugene Kharitonov, Jade Copet, Yossi Adi, Wei-Ning Hsu, Ali Elkahky, Paden Tomasello, Robin Algayres, Benoit Sagot, Abdelrahman Mohamed, et al. Generative spoken dialogue language modeling. *Transactions of the Association for Computational Linguistics*, 11:250–266, 2023.

[26] Tu Anh Nguyen, Benjamin Muller, Bokai Yu, Marta R Costa-Jussa, Maha Elbayad, Sravya Popuri, Paul-Ambroise Duquenne, Robin Algayres, Ruslan Mavlyutov, Itai Gat, et al. Spirit-lm: Interleaved spoken and written language model. *arXiv preprint arXiv:2402.05755*, 2024.

[27] Alexis Plaquet and Hervé Bredin. Powerset multi-class cross entropy loss for neural speaker diarization. In *Proc. INTERSPEECH 2023*, 2023.

[28] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR, 2023.

[29] Hendrik Schröter, A Maier, Alberto N Escalante-B, and Tobias Rosenkranz. Deepfilternet2: Towards real-time speech enhancement on embedded devices for full-band audio. In *2022 International Workshop on Acoustic Signal Enhancement (IWAENC)*, pages 1–5. IEEE, 2022.

[30] Changli Tang, Wenyi Yu, Guangzhi Sun, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun Ma, and Chao Zhang. Salmonn: Towards generic hearing abilities for large language models. *arXiv preprint arXiv:2310.13289*, 2023.

[31] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[32] Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.

[33] Chen Wang, Minpeng Liao, Zhongqiang Huang, Jinliang Lu, Junhong Wu, Yuchen Liu, Chengqing Zong, and Jiajun Zhang. Blsp: Bootstrapping language-speech pre-training via behavior alignment of continuation writing. *arXiv preprint arXiv:2309.00916*, 2023.

[34] Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*, 2023.

[35] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. Next-gpt: Any-to-any multimodal llm. *arXiv preprint arXiv:2309.05519*, 2023.

[36] Zhifei Xie and Changqiao Wu. Mini-omni: Language models can hear, talk while thinking in streaming. *arXiv preprint arXiv:2408.16725*, 2024.

[37] Detai Xin, Xu Tan, Shinnosuke Takamichi, and Hiroshi Saruwatari. Bigcodec: Pushing the limits of low-bitrate neural speech codec. *arXiv preprint arXiv:2409.05377*, 2024.

[38] Dongchao Yang, Haohan Guo, Yuanyuan Wang, Rongjie Huang, Xiang Li, Xu Tan, Xixin Wu, and Helen Meng. Uniaudio 1.5: Large language model-driven audio codec is a few-shot audio task learner. *arXiv preprint arXiv:2406.10056*, 2024.

[39] Dongchao Yang, Songxiang Liu, Rongjie Huang, Jinchuan Tian, Chao Weng, and Yuexian Zou. Hifi-codec: Group-residual vector quantization for high fidelity audio codec. *arXiv preprint arXiv:2305.02765*, 2023.

[40] 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.

[41] Lijun Yu, Yong Cheng, Zhiruo Wang, Vivek Kumar, Wolfgang Macherey, Yanping Huang, David Ross, Irfan Essa, Yonatan Bisk, Ming-Hsuan Yang, et al. Spae: Semantic pyramid autoencoder for multimodal generation with frozen llms. *Advances in Neural Information Processing Systems*, 36, 2024.

[42] 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*, 36:78808–78823, 2023.

[43] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.

[44] Xin Zhang, Dong Zhang, Shimin Li, Yaqian Zhou, and Xipeng Qiu. Speechtokenizer: Unified speech tokenizer for speech large language models. *arXiv preprint arXiv:2308.16692*, 2023.

[45] Kaizhi Zheng, Xuehai He, and Xin Eric Wang. Minigpt-5: Interleaved vision-and-language generation via generative vokens. *arXiv preprint arXiv:2310.02239*, 2023.

[46] Lei Zhu, Fangyun Wei, and Yanye Lu. Beyond text: Frozen large language models in visual signal comprehension. *arXiv preprint arXiv:2403.07874*, 2024.