

A Implementation Details for Streaming Models

We describe the hyperparameter details of StreamFlow for streaming models at Table 6. For EnCodec Token, we successfully train the model with small segment size. However, we found that extracting Mimi tokens with small segment size significantly decrease the performance because Mimi compressed waveform signal of 24,000 Hz into 12.5 Hz. In this regard, we pre-trained the StreamFlow-Mimi with larger segment size. Furthermore, we utilize small size of audio/token drop for Mimi model because we only utilized two-step generation. Due to the limited GPU resource, we only trained the StreamFlow-Mimi for 0.15M steps. However, our model shows much better performance than Mimi.

Table 6: Hyperparameters of StreamFlow for streaming EnCodec token reconstruction.

Module	Hyperparameter	SteamFlow-T	SteamFlow-S	SteamFlow-B	SteamFlow-Mimi
Time	Time Embedding	256	512	1024	1024
	Linear1	[256, 1024]	[512, 2048]	[1024, 4096]	[1024, 4096]
	Activation	SiLU	SiLU	SiLU	SiLU
	Linear2	[1024, 256]	[2048, 512]	[4096, 1024]	[4096, 1024]
Condition	Token	EnCodec	EnCodec	EnCodec	Mimi
	Token Hz	75 Hz	75 Hz	75 Hz	12.5Hz
	Frame per token	320	320	320	1920
	Token dim	128	128	128	512
	Linear1	[128, 256]	[128, 512]	[128, 1024]	[512, 1024]
	Activation1	GELU	GELU	GELU	GELU
	Linear2	[256, 256]	[512, 512]	[1024, 1024]	[1024, 1024]
	Activation2	GELU	GELU	GELU	GELU
	Upsampling	Repeating 2	Repeating 2	Repeating 2	Repeating 12
Input&Prompt Linear Reshape	h	160	160	160	160
	Linear1 (No Bias)	[160, 512]	[160,1024]	[160, 2048]	[160, 2048]
	Linear2	[512, 256]	[1024,512]	[2048, 1024]	[2048, 1024]
Output Linear Reshape	Linear1	[256, 512]	[512,1024,]	[1024,2048]	[1024,2048]
	Linear2 (No Bias)	[512,160]	[1024,160]	[2048,160]	[2048,160]
	h	160	160	160	160
Scale-DiT	Input Dim.	256	512	1024	1024
	Hidden Dim.	1024	2048	4096	4096
	Layer	8	8	8	8
	Head	4	8	16	16
	Stream Token	8	8	8	2
	Prompt Token	24	24	24	6
Pre-train	Training Step	1M	1M	1M	0.5M
	Learning Rate	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}
	Learning Scheduling	-	-	-	-
	Batch Size	512	512	512	128
	GPUs	4	4	4	4
	Noise Scale	0.25	0.25	0.25	0.25
	Segment Size	10240	10240	10240	48000
	Audio Drop	0.3	0.3	0.3	0.3
	Token Drop	0.2	0.2	0.2	0.2
Fine-tuning	Training Step	0.25M	0.25M	0.25M	0.15M
	Learning Rate	2×10^{-5}	2×10^{-5}	2×10^{-5}	2×10^{-5}
	Learning Scheduling	-	-	-	-
	Batch Size	64	64	64	64
	GPUs	4	4	4	4
	Noise Scale	0.25	0.25	0.25	0.25
	Segment Size	20480	20480	20480	30720
	Audio Drop	0.3	0.3	0.3	0.1
	Token Drop	0.2	0.2	0.2	0.1

B Implementation Details for Parallel Models

We describe the hyperparameter details of StreamFlow for parallel models at Table 7. We replace the linear-reshape transformation with STFT and iSTFT. We do not utilize any iSTFT head which was used in Vocos. We can not train the model with iSTFT head of Vocos during pre-training. We directly project components for iSTFT.

Table 7: Hyperparameters of StreamFlow for parallel EnCodec token reconstruction.

Module	Hyperparameter	SteamFlow + iSTFT	SteamFlow
Time	Time Embedding	1024	1024
	Linear1	[1024, 4096]	[1024, 4096]
	Activation	SiLU	SiLU
	Linear2	[4096, 1024]	[4096, 1024]
Condition	Token	EnCodec	EnCodec
	Token Hz	75 Hz	75 Hz
	Frame per token	320	320
	Token dim	128	128
	Linear1	[128, 1024]	[512, 1024]
	Activation1	GELU	GELU
	Linear2	[1024, 1024]	[1024, 1024]
	Activation2	GELU	GELU
	Upsampling	Repeating 2	Repeating 2
Input&Prompt Linear Reshape	h	-	160
	Linear1 (No Bias)	-	[160, 2048]
	Linear2	-	[2048, 1024]
Output Linear Reshape	Linear1	-	[1024, 2048]
	Linear2 (No Bias)	-	[2048, 160]
	h	-	160
STFT/iSTFT	Hop/Window/FFT	160/640/640	-
Scale-DiT	Input Dim.	1024	1024
	Hidden Dim.	4096	4096
	Layer	8	8
	Head	16	16
Pre-train	Training Step	1M	1M
	Learning Rate	2×10^{-4}	2×10^{-4}
	Learning Scheduling	-	-
	Batch Size	128	128
	GPUs	4	4
	Noise Scale	0.25	0.25
	Segment Size	48000	48000
	Audio Drop	0.3	0.3
	Token Drop	0.2	0.2
Fine-tuning	Training Step	0.25M	0.25M
	Learning Rate	2×10^{-5}	2×10^{-5}
	Learning Scheduling	-	-
	Batch Size	32	32
	GPUs	4	4
	Noise Scale	0.25	0.25
	Segment Size	48000	48000
	Audio Drop	0.3	0.3
	Token Drop	0.2	0.2

C Additional Experiments on Pre-trained StreamFlow

Table 8 provides additional objective evaluation results of the pre-trained StreamFlow on the LibriTTS-dev dataset without adversarial fine-tuning. We evaluate both non-streaming with different sampling steps and classifier-free guidance values. Notably, comparable performance to the 16-step setting is achieved even with only 4 sampling steps, demonstrating the efficiency in both offline and streaming scenarios.

Table 8: Objective Evaluation for the pre-trained StreamFlow without adversarial fine-tuning on the LibriTTS-dev subsets.

Model	Sampling Steps	CFG	M-STFT ↓	PESQ ↑	Period. ↓	V/UV ↑	Pitch ↓	UTMOS ↑
EnCodec	1	-	1.163	2.771	0.113	0.941	32.147	2.969
Non-streaming (offline)								
StreamFlow + iSTFT	16	1	1.301	3.094	0.087	0.953	28.635	3.450
StreamFlow + iSTFT	16	0.5	1.311	3.143	0.085	0.954	26.535	3.539
StreamFlow + iSTFT	16	0	1.399	2.827	0.094	0.950	27.020	3.482
StreamFlow + iSTFT	4	1	1.558	2.617	0.092	0.950	26.674	3.238
StreamFlow + iSTFT	4	0.5	1.561	2.668	0.089	0.952	27.905	3.339
StreamFlow + iSTFT	4	0	1.649	2.450	0.094	0.950	27.940	3.320
Streaming (online)								
StreamFlow	16	1	1.343	2.719	0.097	0.949	22.096	3.147
StreamFlow	16	0.5	1.341	2.790	0.093	0.952	20.328	3.224
StreamFlow	16	0	1.388	2.669	0.101	0.950	21.099	3.178
StreamFlow	4	1	1.509	2.479	0.101	0.946	20.956	3.012
StreamFlow	4	0.5	1.504	2.607	0.094	0.952	17.812	3.131
StreamFlow	4	0	1.570	2.557	0.095	0.952	18.697	3.149

D Implementation Details for Baselines

EnCodec We utilize an official implementation of EnCodec [8], which is the popular neural audio codec using RVQ and adversarial training.³ They utilize causal convolutional layer for streaming application, and train the model with 32 quantizer of RVQ. However, most application utilized eight quantizer as target tokens so we train the model with eight tokens to reconstruct the waveform signal.

Vocos We utilize an official implementation of Vocos [43], which is an iSTFT-based waveform generation model with adversarial training.⁴ They utilize an EnCodec as an input representation to reconstruct the waveform signal. We also generate the waveform signal by chunk-wise generation using Vocos to compare the streaming generation performance. For a fair comparison, we utilize the same previous and delayed tokens for chunk-wise generation for robust generation of Vocos.

MBD We utilize an official implementation of Multi-band Diffusion (MBD) [42] as a strong baseline for EnCodec token reconstruction.⁵ MBD consists of four models for each band, and has large-scale parameters of 411M. Then, they utilize 10 sampling steps for each band so it takes a lot of time to generate the waveform signal even with parallel generation.

RFWave We utilize an official implementation of RFWave [29], which a strong baseline using conditional flow matching for multi-band parallel generation.⁶ We utilize 20 sampling steps for better performance and CFG of 2 which is suggested by official implementation.

Mimi We utilize an official implementation of Mimi⁷ from Moshi [6]. They utilize causal convolutional layer for streaming generation, and train the model with 32 quantizer of RVQ and compressed high-resolution waveform signal of 24,000 Hz into 12.5 Hz for efficient speech language models. Moshi only utilizes eight quantizer of RVQ for target tokens by delayed prediction so we also train the model with eight tokens.

³<https://github.com/facebookresearch/encodec>

⁴<https://github.com/gemelo-ai/vocos>

⁵<https://github.com/facebookresearch/audiocraft>

⁶<https://github.com/bfs18/rfwave>

⁷<https://github.com/kyutai-labs/moshi>

E Societal Negative Impact

Although our method and model do not directly contribute to malicious use or ethical concerns, they can be misused when combined with text-to-speech or voice conversion models to deceive people. Therefore, it is crucial to explore fake audio detection and voice phishing detection models alongside our research. In the future, we aim to develop speech language models capable of identifying fake audio based on its content.

F Evaluation Details

M-STFT We employed the multi-resolution Short-Time Fourier Transform (M-STFT) distance implemented in the open-source Auraloss [44].⁸ Originally proposed in Parallel WaveGAN [50], the M-STFT quantifies the distances between ground-truth and generated audio samples across multiple STFT resolutions, thereby capturing both fine and coarse spectral details.

PESQ For evaluating reproduction quality, we utilized the wide-band (WB) Perceptual Evaluation of Speech Quality (PESQ) metric⁹. The audio signals were downsampled to a sampling rate of 16,000 Hz before calculating the PESQ scores to ensure consistency with standard evaluation protocols. Furthermore, we normalize the downsampled waveform signal to avoid overflow.

Periodicity, V/UV F1, and Pitch Following the observations of CarGAN [32] regarding the perceptual degradation caused by periodicity artifacts, we measured periodicity errors using the Periodicity Root Mean Square Error (RMSE)¹⁰. Additionally, we evaluated the Voice/Unvoice (V/UV) classification performance using the F1 score to assess the accuracy of voiced and unvoiced regions in the generated audio. Also, we calculated pitch errors using the pitch Root Mean Square Error (RMSQ). All metrics utilize pitch predicted by CREPE [16]. We used the pytorch implementation of CREPE.¹¹

UTMOS To assess the naturalness of the generated samples, we utilized the open-source Mean Opinion Score (MOS) prediction model, UTMOS [41].¹² UTMOS has demonstrated consistent MOS prediction performance on neutral English speech datasets, providing a reliable measure of the perceived naturalness of the synthesized audio without reference samples.

⁸<https://github.com/csteinmetz1/auraloss>

⁹<https://github.com/ludlows/PESQ>

¹⁰<https://github.com/descriptinc/cargan>

¹¹<https://github.com/maxrmorrison/torchcrepe>

¹²<https://github.com/tarepan/SpeechMOS>

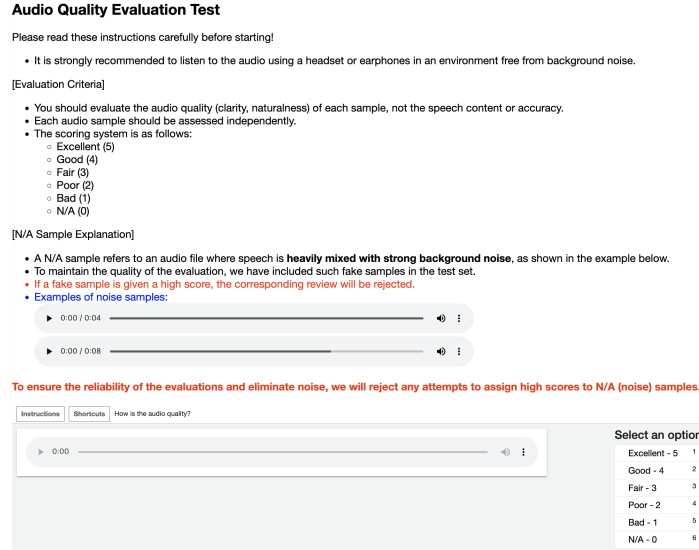


Figure 5: Details of the MOS evaluation interface provided to crowdsourcing participants

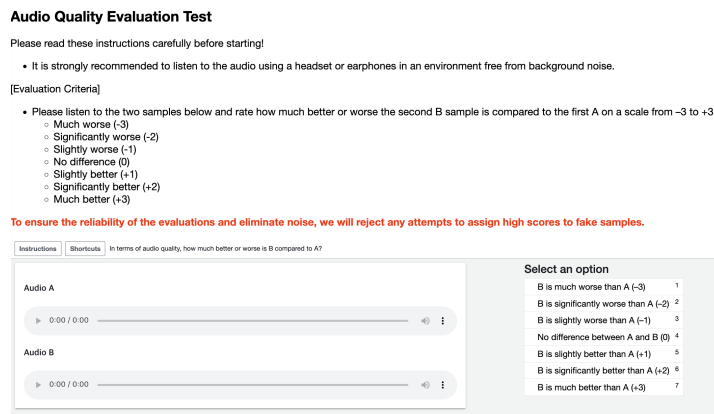


Figure 6: Details of the CMOS evaluation interface provided to crowdsourcing participants

G Crowdsourcing Details

We conducted Mean Opinion Score (MOS) evaluations using a 5-point scale to assess the quality of speech. The perceptual quality of each model was evaluated through a crowdsourced listening test using Amazon Mechanical Turk (MTurk)¹³. A total of 20 native English speakers from the United States participated, each rating 300 samples per model on a scale from 1 to 5. We paid \$180 for each MOS experiment. To ensure the reliability of responses, we established strict participant eligibility criteria: only individuals with a prior task approval rate of at least 50% and a minimum of 100 approved HITs were permitted to take part in this evaluation. Additionally, we included Gaussian noise fake samples as control samples to enhance evaluation robustness, and we give the instruction which noise samples should be assigned as N/A sample (0 point). Listener responses were excluded from the final analysis if they met either of the following conditions: (1) assigning a score over 1 to the noise augmented samples, or (2) spending less than half the duration of an audio sample on the evaluation. These measures were implemented to filter out inattentive participants and maintain the integrity of the collected ratings. The user interface used for the evaluation is illustrated in Figure 5 and Figure 6.

¹³<https://www.mturk.com/>

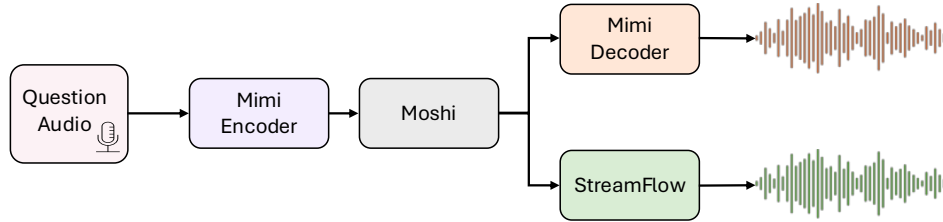


Figure 7: Inference pipeline comparing the Mimi decoder and the proposed StreamFlow, given identical Moshi output.

Table 9: Comparison of real-time decoding performance for full duplex spoken dialogue system, Moshi using the Mimi decoder and the proposed StreamFlow

Method	CER ↓	WER ↓	UTMOS ↑
Moshi w/ Mimi decoder	7.59	9.89	3.610
Moshi w/ StreamFlow (Ours)	7.19	9.82	3.847

H Replacing Mimi with StreamFlow in a Full-duplex Streaming Model

We replaced the original Mimi decoder in Moshi with StreamFlow, successfully integrating it into Moshi’s fully-duplex streaming speech language model. For this integration, we utilized the official Moshi code¹⁴, and conducted experiments using question audio from HeySQuAD [48]¹⁵ dataset as input. We upsampled the dataset, originally at 16 kHz sampling rate, to 24 kHz using Librosa and used it as input to Moshi. The outputs of the Moshi were kept identical across all conditions, allowing for a controlled comparison between the original Mimi decoder and our streaming StreamFlow. We evaluated both speech quality and speech intelligibility, using UTMOS, CER, and WER respectively as metrics. As shown in Table 9, our proposed StreamFlow consistently outperformed the Mimi decoder.

As shown in Figure 7, the pipeline of [Question Audio → Mimi Encoder → LLM → Selectable Decoder] demonstrates that improvements at the decoder stage can lead to substantial gains in speech generation quality. These experiments support the effectiveness of the proposed decoder in streaming-based speech generation scenarios and highlight its practical potential as a viable alternative to the original architecture.

¹⁴https://github.com/kyutai-labs/moshi/blob/main/moshi/moshi/run_inference.py

¹⁵https://huggingface.co/datasets/yijingwu/HeySQuAD_human