StreamFlow: Streaming Audio Generation from Discrete Tokens via Streaming Flow Matching

Ha-Yeong Choi[1] **Sang-Hoon Lee**[2,3*]

[1]Gen AI Lab, KT Corp., Seoul, Korea
[2]Department of Software and Computer Engineering, Ajou University, Suwon, Korea
[3]Department of Artificial Intelligence, Ajou University, Suwon, Korea

Abstract

Diffusion models have demonstrated remarkable generative capabilities, and Conditional Flow Matching (CFM) has improved their inference efficiency by following optimal transport paths. However, CFM-based models still require multiple iterative sampling steps, which makes them unsuitable for real-time or streaming generation scenarios. In this paper, we introduce StreamFlow, a novel streaming generative model designed for real-time audio generation from discrete tokens. StreamFlow leverages a causal noising training framework along the time axis and predicts multi-time vector fields at once on each stream, enabling streaming inference with minimal latency. To further improve generalization, we propose Scale-DiT, a Diffusion Transformer architecture that enhances robustness by modeling, normalizing, and scaling feature differences prior to skip connections. This significantly improves the robustness and performance of DiT without increasing the parameter size. We validate the effectiveness of StreamFlow through audio reconstruction tasks using discrete tokens from EnCodec and Mimi, demonstrating both high-fidelity synthesis and streaming capability. Furthermore, we successfully incorporated our model into fully-duplex streaming speech language models of Moshi by replacing the Mimi decoder. Audio samples are available at `https://streamflow25.github.io/demo/`.

1 Introduction

Recently, conditional flow matching (CFM) models [31] have demonstrated powerful generative capabilities across modalities such as text, image, and audio. Initially, CFM was introduced for image-to-image translation and text-to-image generation by conditioning on target labels or text representation such as those from CLIP [42]. Since then, CFM has been extended to a wide range of audio applications, including text-to-speech [24, 10, 4, 8], text-to-audio [41, 26], voice conversion [56, 5, 62], and waveform generation [30, 32, 50]. Additionally, flow matching has been successfully adapted to discrete domains, such as text generation, through discrete flow matching [11]. Most existing CFM approaches focus on parallel generation tasks, such as fixed-size image generation and fixed-length audio generation in a non-autoregressive manner. Although masking and infilling training frameworks with CFM objectives and Transformers [24, 10, 4] can endow the model with in-context learning over long sequences, these models still require numerous sampling steps over a fixed target space, limiting their applicability to real-time streaming generation tasks. To date, streaming

[*]Corresponding author

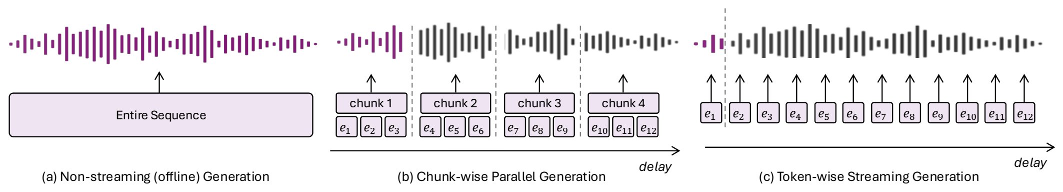

Entire Sequence
(a) Non-streaming (offline) Generation
chunk 1
chunk 2
chunk 3
chunk 4
e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8 e_9 e_{10} e_{11} e_{12}
delay
(b) Chunk-wise Parallel Generation
e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8 e_9 e_{10} e_{11} e_{12}
delay
(c) Token-wise Streaming Generation

Figure 1: Comparison of generation strategies: (a) Non-Streaming Generation, (b) Chunck-wise Parallel Generation, (c) Token-wise Streaming Generation

generation in diffusion models has largely been explored in video domain. From an engineering perspective, StreamDiffusion [19] introduces the batch denoising method that computes the diffusion process with a batch. Rolling Diffusion [43] introduces a novel DDPM [14] that progressively denoises data over time.

However, to the best of our knowledge, streaming audio generation from discrete tokens has not yet been explored within the frameworks of diffusion or flow matching. While recent parallel generation approaches based on CFM [30] have achieved promising results, they typically operate on a chunk-wise generation, as shown in Figure 1-(b), relying on pre-trained non-autoregressive models. Moreover, generating raw waveform at a high sampling rate (e.g., 24 kHz) introduces additional challenges, as it requires significantly finer resolution compared to lower-resolution acoustic features (e.g., 12.5 and 75 Hz), thus increasing both modeling complexity and computational cost.

Additionally, recent speech language models are getting more attention for end-to-end human-like communication such as GPT-4o [15]. They utilize neural audio codecs with causal layers and GAN-based objective for high-quality waveform generation [7]. However, these models often suffer from perceptual degradation when generating high-resolution waveforms directly from highly compressed, low-bitrate discrete tokens. This is primarily due to the absence of intermediate temporal modeling and progressive refinement, which are essential for maintaining local coherence and preserving fine-grained audio details over time, especially in streaming scenarios where real-time and frame-level consistency are crucial.

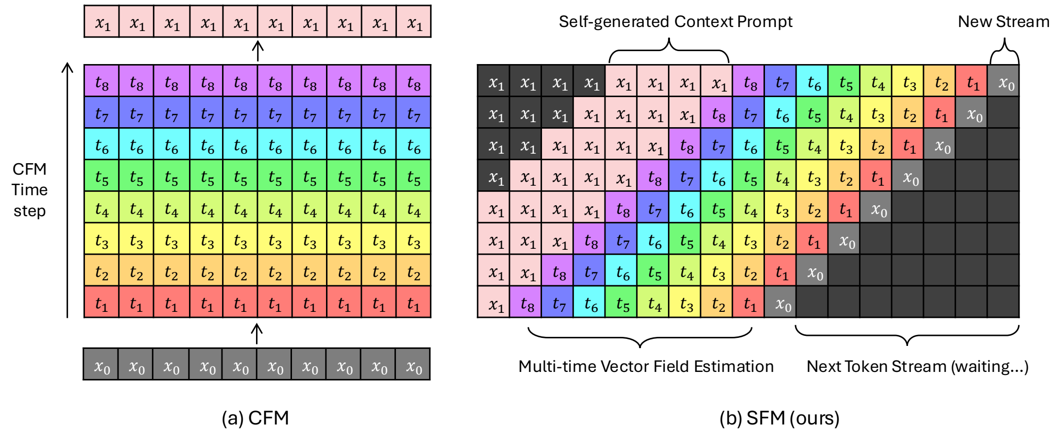
In this paper, we introduce StreamFlow, a novel real-time streaming flow matching model for streaming audio generation. StreamFlow simultaneously estimates multi-time vector fields at each step, thereby offering both streaming generation and refinement capabilities. To facilitate long-range temporal modeling, we adopt diffusion transformers with in-context learning capabilities. In addition, we introduce Scale-DiT, a new diffusion transformer architecture designed to improve feature regularization. Scale-DiT computes the difference between residuals and features, normalizes this difference, and applies a scaling operation before the skip connection. This design improves robustness without increasing model size. We evaluate the effectiveness of StreamFlow on real-time audio reconstruction tasks using discrete tokens from EnCodec [9] and Mimi [7]. Furthermore, we verify that StreamFlow can be seamlessly integrated into existing speech generation pipelines that rely on discrete neural audio codecs. The main contributions of this work are as follows:

- We propose StreamFlow, a novel streaming generative model that leverages self-conditioned context to estimate multi-time vector fields, enabling token-wise streaming generation.
- We introduce a new DiT structure, Scale-DiT that regularizes the feature space by modeling the difference between residual and features, normalizing the difference, and scaling it for elucidating the features.
- We demonstrate the effectiveness of the proposed Streaming Flow Matching framework for streaming audio generation by reconstructing high-quality waveforms in real time from discrete tokens produced by neural audio codecs such as EnCodec and Mimi.

2 Related Work

2.1 Waveform Generation

Conventionally, neural waveform generation tasks have been investigated for text-to-speech systems requiring conversion of acoustic features such as Mel-spectrogram to raw waveform signal, called


CFM Time step
Self-generated Context Prompt
New Stream
Multi-time Vector Field Estimation
Next Token Stream (waiting...)
(a) CFM
(b) SFM (ours)
Figure 2: Comparison of (a) conditional flow matching (CFM), which performs full-sequence vector field estimation across all timesteps, and (b) streaming flow matching (SFM), our method that leverages self-generated context for multi-time vector field estimation and enables low-latency token-wise streaming.

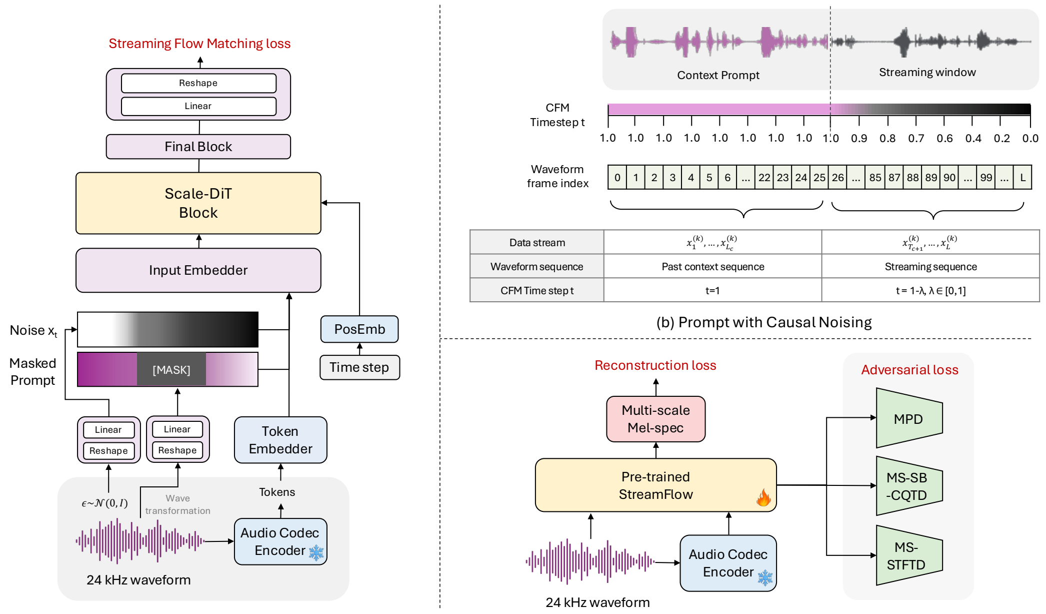
by *Neural Vocoder*. WaveNet [38] introduced causal dilated convolutional networks to increase the receptive field for high-resolution waveform signal modeling. Due to the slow sampling speed of auto-regressive generation, many works [37, 40, 18] have shifted their focus to parallel waveform generation models for efficient and fast waveform generation. GAN-based parallel models have shown promising parallel generation performance by generating realistic waveform signal. PWG [54] and MelGAN [22] first adopted adversarial training, and HiFi-GAN [20] introduced a novel discriminator, multi-period discriminator (MPD) reflecting implicit periodic features. BigVGAN [25] scaled up the neural vocoder for out-of-distribution robustness by introducing snake activation and an anti-aliasing filter. Then, Vocos [46] boosted the sampling speed by incorporating iSTFT on low-resolution feature space.

2.2 CFM in Waveform Generation

Recently, CFM has been adopted for high-fidelity waveform generation. Although previous diffusion-based models [21, 2] showed slow inference speed and lower performance than GAN-based models, PeriodWave [30], RFWave [32], and FlowDec [50] demonstrated much higher performance on the waveform generation tasks compared to GAN models. Although CFM still requires more sampling steps compared to GAN, they have shown promising results on the waveform generation tasks. Furthermore, PeriodWave-Turbo [28] accelerated CFM-based models by incorporating adversarial feedback, achieving improved performance with smaller sampling steps. However, as shown in Figure 2-(a), streaming generation within the CFM has not been explored.

2.3 Neural Audio Codec

Meanwhile, neural audio codec has garnered more attention than neural vocoder in that it facilitates various practical applications combined with large language models. SoundStream [59] and EnCodec [9] introduced a practical neural audio codec by incorporating residual vector quantization (RVQ) [12, 49] and adversarial training. DAC [23] improved RVQ with carefully designed bottleneck and various losses. HiFi-Codec [55] proposed group-residual vector quantization to increase the capacity of RVQ. SpeechTokenizer [61] and X-Codec [57] distilled the self-supervised speech representation on the quantized representation to increase semantic information without any labeled data. Mimi [7] also introduced a high-compressed low-bitrate audio codec with SSL distillation for an efficient speech language model. Furthermore, single VQ models including WavTokenizer [16], BigCodec [53], and TS3-Codec [51] have been investigated for efficient modeling. Recently, finite scalar quantization (FSQ)-based models [1, 39] have been adopted to improve the performance of low-bitrate neural audio codec. CosyVoice 2 [8] introduces supervised semantic tokens and flow matching with causal layer and lookahead tokens.



(a) Architecture of StreamFlow (b) Prompt with Causal Noising (c) Fine-tuning with Adversarial Training

Figure 3: The Overall framework of StreamFlow.

2.4 Full-Duplex Real-time Speech Language Models

Recent advances in speech foundation models [7, 3] have introduced full-duplex systems capable of simultaneous speech input and output in real-time. Achieving this requires both low-latency streaming architectures and highly efficient low-bitrate neural codecs. A common design pattern involves causal convolution-based decoders for real-time waveform generation. However, such causal models generally underperform compared to their parallel counterparts in terms of perceptual quality. Moreover, the reliance on low-bitrate codecs, while necessary for efficient streaming, often introduces noticeable degradation in audio quality and fidelity. To address these limitations, we propose SFM, a method that enables high-fidelity, low-latency waveform generation—bridging the gap between efficiency and perceptual quality in streaming audio generation, as illustrated in Figure 2-(b).

3 StreamFlow

This section introduces the StreamFlow framework for high-quality real-time audio streaming generation. We begin with the fundamentals of Flow Matching (Section 3.1) and extend it to an efficient Streaming Flow Matching method (Section 3.2). We then describe the data stream, causal noising, training objective, and streaming generation process. Next, we present Scale-DiT for improved contextual learning and generalization (Section 3.3), and explore waveform transformation for efficient streaming generation (Section 3.4). Finally, we discuss adversarial training for enhanced robustness and efficiency (Section 3.5).

3.1 Preliminary: Flow Matching

Flow Matching (FM) [31, 33, 48] is a generative modeling method designed to transform a simple prior distribution $p_0(x)$ (e.g., $x_0 \sim \mathcal{N}(0, I)$) into a target data distribution $p_1(x) \approx q(x)$ by learning a time-dependent velocity field $v_t(x)$. This transformation is governed by the following ordinary differential equation (ODE):

$$\frac{d\phi_t(x)}{dt} = v_t(\phi_t(x); \theta), \quad \phi_0(x) = x,\ x \sim p_0, \tag{1}$$

where $\phi_t(x)$ represents the state of the system at time t as it evolves under the velocity field v_t. The FM objective aims to align the learned velocity field $v_t(x; \theta)$ with an ideal vector field $u_t(x)$ that generates the desired probability path $p_t(x)$. This alignment is achieved by minimizing the following training objective:

$$L_{\text{FM}}(\theta) = \mathbb{E}_{t\sim[0,1],x\sim p_t(x)} \left\| v_t(x;\theta) - u_t(x) \right\|^2. \tag{2}$$

To define the probability path $p_t(x)$, a common approach is to employ linear interpolation between the prior sample x_0 and the target data point x_1:

$$x_t = (1-t)x_0 + tx_1. \tag{3}$$

The corresponding ideal vector field $u_t(x_t \mid x_1)$ that drives the transformation along this path is given by:

$$u_t(x_t \mid x_1) = x_1 - x_0. \tag{4}$$

This method effectively learns a smooth and computationally efficient mapping from p_0 to p_1.

3.2 Streaming Flow Matching

We introduce Streaming Flow Matching (SFM), a novel approach designed for real-time streaming inference by simultaneously estimating multi-time vector fields. Unlike traditional FM, which processes entire sequences simultaneously, SFM utilizes dynamically structured streaming sequences and jointly optimizes local and global vector fields. Notably, by estimating multiple vector fields across multiple time axes, SFM provides not only the global consistency inherited from classical FM but also satisfies real-time constraints in streaming scenarios.

Data Streams SFM partitions the input sequence $\mathbf{x}$ into multiple data streams, each characterized by a streaming window size L_s and a context prompt ratio γ. The total length is thus $L = L_\text{c} + L_\text{s}$, where $L_\text{c} = \gamma \cdot L_\text{s}$. Specifically, each data stream $\{\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_L^{(k)}\}$ simultaneously provides contextual information (i.e., the context prompt) and performs streaming inference (i.e., the streaming window). To achieve this, SFM defines two separate timestep sequences. The first sequences corresponds to the past context segment, which remains fixed at $t = 1$. The second sequences handles the streaming segment, where t decreases linearly as $t = 1 - \lambda, \lambda \in [0, 1]$ along to the L_s.

Causal Noising To ensure causality, the SFM introduces the causal noising technique, selectively masking the streaming window while retaining context prompt for in-context learning. This approach imposes temporal constraints, preventing information leakage from future data. For each stream k, let the initial state $\mathbf{x}_0^{(k)}$ follow a Gaussian distribution $\mathcal{N}(0, I)$, and let $\mathbf{x}_1^{(k)}$ be the target data point. Then, the following interpolation path is defined for $t \in [0, 1]$:

$$\mathbf{x}_t^{(k)} \;=\; (1-t)\,\mathbf{x}_0^{(k)} \;+\; t\,\mathbf{x}_1^{(k)}. \tag{5}$$

Within this path, the streaming window is monotonically reduced from $t = 1$ to 0, effectively reducing the range of actively learned stream. The context prompt, however, remains fixed throughout training, ensuring that the model retains past context information while progressively refining the newly generated output. In a streaming scenario, the context prompt is dynamically updated to integrate newly generated frames while preserving historical context. Meanwhile, the model iteratively refines its predictions through causal noising, wherein structured noise is introduced at each step and progressively reduced over time. Furthermore, newly generated frames are incorporated via a causal shift operation, ensuring a smooth transition between past and present content. As a result, this process not only preserves temporal coherence and stabilizes streaming generation but also prevents information leakage and enhances robustness.

Training Objective The training objective of SFM is to predict the ideal velocity field $\mathbf{u}_t^{(k)}$. This velocity field is derived via the temporal derivative of the interpolated path $\mathbf{x}_t^{(k)}$, which simplifies to

$$\mathbf{u}_t^{(k)} \;=\; \frac{d\,\mathbf{x}_t^{(k)}}{dt} \;=\; \mathbf{x}_1^{(k)} - \mathbf{x}_0^{(k)}. \tag{6}$$

We define the following loss function to minimize the difference between the predicted $\mathbf{v}_t^{(k)}(\theta)$ and the ideal $\mathbf{u}_t^{(k)}$:

$$L_{\text{SFM}}(\theta) = \sum_{k=1}^{K} \mathbb{E}_{t,\mathbf{x}_0^{(k)},\mathbf{x}_1^{(k)}} \left[\left\| \mathbf{v}_t^{(k)}\left(\tilde{\mathbf{x}}_t^{(k)}; \mathbf{c}^{(k-1)}; \theta\right) - \left(\mathbf{x}_1^{(k)} - \mathbf{x}_0^{(k)}\right) \right\|^2 \right], \tag{7}$$

where $\tilde{\mathbf{x}}_t^{(k)}$ is the causally noised interpolation state, $\mathbf{c}^{(k-1)}$ represents context prompt providing temporal consistency and in-context streaming generation ability.

Streaming Generation Thanks to in-context learning ability of Transformer, the model can learn the long-term dependency during streaming generation. To emerge this ability, we train the model by applying causal noising to the data stream, and estimating the multi-time vector fields at once according to the level of noising. Following [24], we also utilize condition drop for prompt and tokens during training. During inference, we prepend the generated samples at the front of the sequence as a prompt to in-context learn from the generated data. Furthermore, $\mathbf{x}_t$ can be refined by reflecting the next token stream during generation.

3.3 Scale-DiT

Inspired by [34], we introduce Scale-DiT, a novel architecture designed to enhance training stability and improve model performance. We aim to preserve the scalability of existing DiT while addressing limitations in optimization and feature representation. This approach allows the model to regulate feature more effectively during training, leading to more stable optimization within Multi-Head Self-Attention (MHSA) and Feed-Forward Network (FFN) layers. Specifically, Adaptive Layer Normalization (AdaLN) is integrated with a dynamic scaling mechanism to regularize the feature space effectively. Given an input sequence $x \in \mathbb{R}$, we first apply AdaLN to obtain a normalized representation $\hat{x}$ along with trainable scaling and shifting parameters $\gamma_1, \beta_1, \gamma_2, \beta_2$, and gating factors α_1, α_2. The MHSA output is computed as:

$$\hat{x} = \sigma(x) \cdot (1 + \gamma_1) + \beta_1, \tag{8}$$

where $\sigma(\cdot)$ denotes the layer normalization (LN) function.

$$\mathbf{A} = \alpha_1 \cdot \text{MHSA}(\hat{x}). \tag{9}$$

Unlike vanilla DiT residual connections, we introduce an learnable adaptive scaling rate ρ_1, which is restricted between 10^{-4} and 1 before applying it for robust training. This adaptive rate modulates the residual update with the difference:

$$x \leftarrow x + \rho_1 \cdot \sigma(\mathbf{A} - x), \tag{10}$$

where we utilize additional LN, ensuring controlled information flow and stable optimization for the difference. Following the MHSA computation, we apply an AdaLN as:

$$\hat{x} = \sigma(x) \cdot (1 + \gamma_2) + \beta_2 \tag{11}$$

before passing it through the FFN, which is further scaled by α_2, leading to

$$\mathbf{F} = \alpha_2 \cdot \text{FFN}(\hat{x}). \tag{12}$$

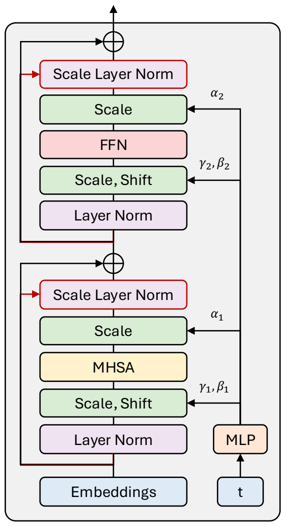


Figure 4: Scale-DiT architecture of StreamFlow.

To maintain training stability and prevent extreme gradient updates, we introduce a separate adaptive scaling rate ρ_2 for the FFN layer. Finally, the residual connection is updated as:

$$x \leftarrow x + \rho_2 \cdot \sigma(\mathbf{F} - x). \tag{13}$$

By scale layer normalization, Scale-DiT improves training stability, optimizes representation learning efficiency, and enhances overall optimization dynamics.

Table 1: Objective evaluation results from EnCodec tokens using universal speech test samples. RFWave uses CFG2.

Model	Streaming	Params.	M-STFT ↓	PESQ ↑	Period ↓	V/UV ↑	UTMOS ↑	MOS ↑
GT	-	-	-	-	-	-	3.423	4.07±0.02
Vocos [46]	✗	7M	1.074	3.051	0.086	0.957	3.100	3.98±0.02
MBD [13]	✗	411M	1.612	2.645	0.108	0.946	3.300	3.95±0.03
RFWave [32]	✗	18M	1.280	3.020	0.078	0.957	2.988	3.99±0.02
StreamFlow	✗	170M	0.997	3.473	0.080	0.957	3.450	4.03±0.02
Encodec [9]	✓	15M	1.170	2.643	0.112	0.941	2.542	3.74±0.03
StreamFlow-Tiny	✓	11M	1.111	3.027	0.107	0.947	3.060	3.99±0.02
StreamFlow-Small	✓	44M	1.072	3.207	0.096	0.950	3.206	3.99±0.03
StreamFlow-Base	✓	175M	1.061	3.335	0.102	0.948	3.325	4.03±0.02

3.4 Waveform Transformation

STFT/iSTFT Conventional waveform generation models used high-resolution features with transposed convolutional layer. However, increasing the resolution of features significantly increases the inference speed. Vocos [46] introduced iSTFT head to directly transform the feature into waveform signals on the low-resolution features. Unlike one-step generation models, we utilize multi-step generation frameworks using CFM so we first extract the STFT-based features of an input $\mathbf{x}_t$ and prompt and iSTFT-based vector field estimation. However, we found that STFT and iSTFT require additional computation cost for each sampling step, and larger receptive field (window size) than the quantization size of the token (hop size) so it is not proper for the streaming generation system because it is essential to use future tokens for robust waveform generation.

Linear-Reshape Transformation To reduce this problem, we adopt a linear reshape transformation proposed in WaveNeXt [36]. We first reshape the 1d waveform-level input $\mathbf{x}_t$ and prompt of length T into 2d data of height T/h and width h, where h denotes hop size like STFT. After reshaping the data, we apply the linear projection to map it into the feature space for Scale-DiT. For vector field prediction, we also utilize linear projection to the data space, and reshape 2d data into 1d original reshape for loss calculation. It is worth noting that it does not require additional future frames and reshape function during sampling steps unlike STFT-based models, resulting in better efficient and faster streaming generation. Specifically, we reshape the waveform signal by h of 160.[2]

3.5 Fine-tuning with Adversarial Training

For high-fidelity waveform generation, many works utilize adversarial training by introducing well-designed discriminators. Although leveraging multi-scale of various discriminator could improve the performance of waveform generation models, it requires too much time to train the models and it is difficult to optimize their losses together. To reduce the entire training time, we first train the StreamFlow using the proposed SFM objective, and fine-tune the pre-trained StreamFlow with adversarial training following [28]. For adversarial training, we utilized the multi-period discriminator (MPD) of HiFi-GAN [20] to reflect the different periodic features, multi-scale short time Fourier transform discriminator (MS-STFTD) of [9] to reflect the magnitude and phase from different STFTs, and multi-scale sub-band constant-Q transform (MS-SB-CQTD) of [13] to reflect the detailed frequency features. Furthermore, we also utilize multi-scale STFT losses of BigVGAN-v2 together.

4 Experiment and Result

4.1 Experimental Setup

Dataset We utilized LibriTTS [60] to train all models including the ablation study. LibriTTS is a high-quality speech dataset with a sampling rate of 24,000 Hz. We used EnCodec and Mimi

[2]We have compared the size of h at our preliminary study. Decreasing h could improve the performance but increase the computation complexity. We recommend decreasing the dimension size when using smaller h. This has the advantage of a smaller parameter size with similar performance.

Table 2: Ablation study on Streaming StreamFlow using the LibriTTS-dev subset.

Model	M-STFT ↓	PESQ ↑	Period ↓	V/UV ↑	Pitch ↓	UTMOS ↑
Streaming (online)						
StreamFlow	1.088	3.430	0.088	0.955	26.843	3.792
w/o In-Context Learning	1.099	3.337	0.088	0.954	26.772	3.748
w/o Adversarial Fine-tuning	1.388	2.669	0.101	0.950	21.099	3.178
w/o SFM Pre-training	1.919	1.153	0.353	0.812	574.55	1.308
w/o Scale-DiT	1.593	2.557	0.099	0.946	27.330	3.033
w/o RoPE	2.049	1.598	0.176	0.898	68.891	1.904
Non-streaming (offline)						
StreamFlow	1.017	3.499	0.085	0.955	29.087	3.888
w/o iSTFT	1.097	3.139	0.082	0.956	22.636	3.619

as speech tokenizers to compare the streaming reconstruction performance because they consist of causal convolutional layers for encoding and decoding the waveform signal. We utilize eight RVQ of each model to train the model.[3] We first validate the models using LibriTTS-dev clean and test subsets, and then we evaluate the performance of each model using universal speech datasets consisting of 300 samples from various datasets including Expresso, HiFiTTS, LibriTTS, Aishell3, JVS, and CML-TTS following RFWave [32].

Training For streaming models, we pre-train StreamFlow models with a learning rate of 2×10^{-4}, batch size of 512 for 1M steps on four NVIDIA A6000 GPUs. We utilize sliced window training by randomly segmenting the waveform signal by 10,240 frames (32 tokens of EnCodec). Then, we fine-tune StreamFlow with a learning rate of 2×10^{-5}, batch size of 64 for 0.25M steps on four NVIDIA A6000 GPUs. We utilize sliced window training by randomly segmenting the waveform signal by 20,480 frames (64 tokens of EnCodec). The architecture details are described in Appendix A. The parallel models are described in Appendix B.

Sampling We utilize the Euler method as the ODE method. We compared the sampling steps for pre-trained models in Table 9. For streaming models, we fine-tuned the model with the fixed-step generator by four steps of parallel models and eight steps of streaming models. For Mimi reconstruction, we fixed two steps for minimal latency.

4.2 EnCodec Token Reconstruction

We compared the performance of EnCodec token reconstruction with the strong parallel baselines including Vocos, Multi-Band Diffusion (MBD), and RFWave. Furthermore, we compare with the streaming baselines including causal EnCodec with the same latency to obtain robust streaming generation. Table 1 demonstrated the effectiveness of our methods in that streaming models still outperformed the powerful parallel models including MBD and RFWave. Furthermore, StreamFlow-T also shows better performance in terms of PESQ. This indicated that our new streaming generation frameworks could improve the streaming generation performance significantly. Table 1 shows that our models have much better performance in terms of MOS compared to previous streaming methods, and also better performance compared to parallel models.

4.3 Ablation Study

In-Context Learning When we prepend the generated samples as prompts, the models could learn useful information from them. Table 2 also showed much better performance than the model without prompt even with the same hyperparameter.

Adversarial Fine-tuning The results demonstrated the efficiency and effectiveness of two-stage training using SFM and adversarial training. Although we only fine-tuned the models with small training steps of 0.25M, the performance improved significantly.

[3]We found that StreamFlow using eight RVQ token embedding for training can still generate waveform signal from any number of RVQ token due to iterative refinement.

Table 3: Scalability with respect to model size

Model	Params.	Input Dim.	Hidden	Head	M-STFT ↓	PESQ ↑	Period ↓	V/UV ↑	Pitch ↓	UTMOS ↑
StreamFlow-Tiny	11M	256	1024	4	1.126	3.183	0.097	0.951	27.846	3.550
StreamFlow-Small	44M	512	2048	8	1.099	3.307	0.089	0.955	27.740	3.690
StreamFlow-Base	175M	1024	4096	16	1.088	3.430	0.088	0.955	26.843	3.792

Table 4: Additional comparison between DiT and Scale-DiT before adversarial fine-tuning. We use 16 steps of Euler method for sampling. Objective evaluation results from Mimi tokens using LibriTTS-dev subsets.

Model	Training Steps	WER ↓	STOI ↑	PESQ ↑	SPK-SIM ↑	UTMOS ↑
DiT	150k	11.76	0.85	1.71	0.58	2.72
DiT + REPA	150k	8.76	0.86	1.74	0.59	2.85
Scale-DiT	150k	9.68	0.86	1.78	0.59	2.83
Scale-DiT + REPA	150k	8.34	0.87	1.78	0.60	2.92
DiT	300k	9.41	0.87	1.84	0.62	3.04
DiT + REPA	300k	8.17	0.87	1.84	0.64	3.10
Scale-DiT	300k	7.56	0.87	1.90	0.64	3.16
Scale-DiT + REPA	300k	7.18	0.88	1.92	0.65	3.26
DiT	700k	9.59	0.87	1.92	0.64	3.20
DiT + REPA	700k	6.88	0.88	1.92	0.67	3.28
Scale-DiT	700k	6.23	0.88	2.07	0.68	3.45
Scale-DiT + REPA	700k	5.99	0.89	2.09	0.68	3.52

SFM Pre-training Without SFM pre-training, the model with adversarial training could not emerge streaming generation ability with ODE sampling at the early steps, resulting in discriminator collapse and slow training speed.

Scale-DiT We found that training the vanilla DiT model with a linear-reshape transformation using a high-resolution waveform signal as a target is unstable. Compared to the vanilla DiT, Table 2 indicated that Scale-DiT could improve the performance. We also observed that the learning process exhibits increased stability with reduced fluctuations during training.

RoPE For streaming generation, it is essential to use positional embeddings. The model without RoPE could not generate the waveform signal well due to a lack of ability to learn the positional information of high-resolution waveform signal.

Linear-Reshape Transformation Table 2 confirmed that STFT/iSTFT-based models are better than Linear-Reshape transformation for parallel generation models so we used STFT/iSTFT-based methods for parallel models. However, as we described in section 3.4, STFT/iSTFT is not proper for streaming generation because it requires a larger receptive field for their transformation so we used Linear-Reshape transformation for streaming methods.

Scalability We train three different models as described in Table 3. The results demonstrate consistent scalability with respect to model size across all evaluation metrics.

4.4 Further Analysis of Scale-DiT

As a further analysis of Scale-DiT, we first compare it with the vanilla DiT baseline to investigate its effectiveness and stability. As shown in Table 4, Scale-DiT consistently achieves better performance across all evaluation metrics, demonstrating enhanced stability and stronger capability for hierarchical feature modeling. These improvements suggest that the proposed scaling mechanism effectively stabilizes training and optimizes feature representations compared to the vanilla DiT. Building upon these results, we conducted additional experiments to investigate whether Scale-DiT can be further enhanced through feature regularization. Specifically, we incorporated representation alignment for generation (REPA) [58] into the architecture to examine its extensibility and robustness. In this setting, the 7th-layer representation from Wav2Vec 2.0 was used as the target semantic embedding, replacing the DINOv2 features employed in the original REPA, and a cosine similarity based REPA loss was applied at the 6th block of our 12-block model. When combined with REPA (Scale-DiT + REPA), the model achieved additional gains indicating that Scale-DiT provides a solid and stable foundation for structured feature learning, while REPA serves as an effective auxiliary regularization technique for further refinement.

Table 5: Objective evaluation results from Mimi tokens using LibriTTS-dev subsets. Note that StreamFlow performs inference in a two-step process.

Model	N_q	Bitrate	F (Hz)	CER ↓	WER ↓	M-STFT ↓	PESQ ↑	Period. ↓	V/UV ↑	Pitch ↓	UTMOS ↑
GT	-	-	-	1.12	3.06	-	-	-	-	-	3.862
Mimi	4	550	50	7.42	12.72	1.552	1.657	0.210	0.880	77.575	3.019
StreamFlow	4	550	50	5.22	9.45	1.410	1.584	0.212	0.876	73.891	3.093
Mimi	6	825	75	5.10	9.00	1.426	2.012	0.180	0.901	60.142	3.347
StreamFlow	6	825	75	3.78	6.87	1.272	2.043	0.177	0.906	55.830	3.719
Mimi	8	1100	100	3.05	6.93	1.352	2.266	0.165	0.910	50.686	3.506
StreamFlow	8	1100	100	3.26	6.16	1.217	2.306	0.162	0.915	45.640	3.910

Table 6: Inference details and subjective evaluation results from Mimi tokens using universal speech test samples. Note that StreamFlow performs inference in a two-step process.

Model	N_q	Bitrate	F (Hz)	Params.	Latency ↓	xRTF ↑	Avg. Memory ↓	CMOS ↑	UTMOS ↑
Mimi	4	550	50	79M	80ms	-	-	-	2.62
StreamFlow	4	550	50	175M	160ms	-	-	+0.058	2.76
Mimi	6	825	75	79M	80ms	-	-	-	2.91
StreamFlow	6	825	75	175M	160ms	-	-	+0.022	3.34
Mimi	8	1100	100	79M	80ms	3.224	502MB	-	3.06
StreamFlow	8	1100	100	175M	160ms	8.319	1176MB	+0.085	3.55

4.5 Mimi Token Reconstruction

We evaluated the Mimi decoder and StreamFlow on Mimi token reconstruction using LibriTTS-dev subsets and universal speech test samples. As detailed in Table 5, StreamFlow outperforms Mimi decoder in terms of WER, M-STFT, PESQ, Pitch, and UTMOS, and has a comparable performance in other metrics. Inference details and subjective scores are summarized in Table 6. Furthermore, StreamFlow directly generated high-resolution waveform signal on the low-resolution features by linear-reshape transformation, resulting in a much faster inference speed compared to the Mimi decoder. This indicates that our model could replace the Mimi decoder for full-duplex real-time speech language models of Moshi. We provide a detailed description of the system's operation in a full-duplex setting in Appendix H.

5 Potential Broader Impact

Practical Application We introduce a new generative model, streaming flow matching (SFM) for streaming generation, and a new diffusion architecture, Scale-DiT. This can be adopted for sequential data generation including video generation, audio generation, and stock prediction. We successfully leveraged our methods to real-time streaming waveform generation by proposing StreamFlow. This can accelerate the research of text-to-speech [27, 29], text-to-audio, real-time voice conversion [6], and speech language models with high-quality audio generation.

Limitation In this work, we focus real-time generation system using SFM. However, we observed that increasing the sampling steps of SFM can further improve the quality. Therefore, we aim to minimize the trade-off between real-time processing and quality even with more sampling steps. While SFM pre-training significantly reduces the overall training time and emerge new ability for streaming generation, adversarial fine-tuning is still required for high-fidelity waveform generation, resulting in slow training speed due to various discriminators.

6 Conclusion

We presented a novel steaming generative models, streaming flow matching (SFM) for real-time streaming generation. Additionally, we introduce Scale-DiT, a robust diffusion Transformers architecture. We verified the effectiveness of our proposed methods in real-time high-resolution waveform signal generation tasks. To achieve this, we carefully designed a new waveform generation model, StreamFlow, which incorporates SFM, Scale-DiT, and linear-reshape transformation for high-resolution waveform modeling using EnCodec and Mimi tokens. Furthermore, we demonstrated the scalability of our models, and we have a plan to train much larger models for better generalization. Additionally, we see that our methods can be applied to directly train the neural audio codec more efficiently.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2025-02283048, Developing the Next-Generation General AI with Reliability, Ethics, and Adaptability, IITP-2025-RS-2023-00255968, the Artificial Intelligence Convergence Innovation Human Resources Development, No.RS-2021-II212068, Artificial Intelligence Innovation Hub), National Research Foundation of Korea(NRF) grant funded by the Korea government (MSIT) (RS-2025-16069227) and Artificial intelligence industrial convergence cluster development project funded by the Ministry of Science and ICT(MSIT, Korea)&Gwangju Metropolitan City.

References

[1] Edresson Casanova, Ryan Langman, Paarth Neekhara, Shehzeen Hussain, Jason Li, Subhankar Ghosh, Ante Jukić, and Sang-gil Lee. Low frame-rate speech codec: a codec designed for fast high-quality speech llm training and inference. *arXiv preprint arXiv:2409.12117*, 2024.

[2] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=NsMLjcFaO8O`.

[3] Qian Chen, Yafeng Chen, Yanni Chen, Mengzhe Chen, Yingda Chen, Chong Deng, Zhihao Du, Ruize Gao, Changfeng Gao, Zhifu Gao, et al. Minmo: A multimodal large language model for seamless voice interaction. *arXiv preprint arXiv:2501.06282*, 2025.

[4] Yushen Chen, Zhikang Niu, Ziyang Ma, Keqi Deng, Chunhui Wang, Jian Zhao, Kai Yu, and Xie Chen. F5-tts: A fairytaler that fakes fluent and faithful speech with flow matching. *arXiv preprint arXiv:2410.06885*, 2024.

[5] Ha-Yeong Choi and Jaehan Park. Voiceprompter: Robust zero-shot voice conversion with voice prompt and conditional flow matching. *arXiv preprint arXiv:2501.17612*, 2025.

[6] Ha-Yeong Choi, Sang-Hoon Lee, and Seong-Whan Lee. Dddm-vc: Decoupled denoising diffusion models with disentangled representation and prior mixup for verified robust voice conversion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17862–17870, 2024.

[7] 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. *arXiv preprint arXiv:2410.00037*, 2024.

[8] Zhihao Du, Yuxuan Wang, Qian Chen, Xian Shi, Xiang Lv, Tianyu Zhao, Zhifu Gao, Yexin Yang, Changfeng Gao, Hui Wang, et al. Cosyvoice 2: Scalable streaming speech synthesis with large language models. *arXiv preprint arXiv:2412.10117*, 2024.

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

[10] Sefik Emre Eskimez, Xiaofei Wang, Manthan Thakker, Canrun Li, Chung-Hsien Tsai, Zhen Xiao, Hemin Yang, Zirun Zhu, Min Tang, Xu Tan, et al. E2 tts: Embarrassingly easy fully non-autoregressive zero-shot tts. *arXiv preprint arXiv:2406.18009*, 2024.

[11] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *arXiv preprint arXiv:2407.15595*, 2024.

[12] Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

[13] Yicheng Gu, Xueyao Zhang, Liumeng Xue, and Zhizheng Wu. Multi-scale sub-band constant-q transform discriminator for high-fidelity vocoder. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 10616–10620. IEEE, 2024.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[15] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[16] 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.

[17] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. Crepe: A convolutional representation for pitch estimation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165. IEEE, 2018.

[18] Sungwon Kim, Sang-Gil Lee, Jongyoon Song, Jaehyeon Kim, and Sungroh Yoon. Flowavenet: A generative flow for raw audio. In *International Conference on Machine Learning*, pages 3370–3378. PMLR, 2019.

[19] Akio Kodaira, Chenfeng Xu, Toshiki Hazama, Takanori Yoshimoto, Kohei Ohno, Shogo Mitsuhori, Soichi Sugano, Hanying Cho, Zhijian Liu, and Kurt Keutzer. Streamdiffusion: A pipeline-level solution for real-time interactive generation. *arXiv preprint arXiv:2312.12491*, 2023.

[20] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in neural information processing systems*, 33:17022–17033, 2020.

[21] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=a-xFK8Ymz5J.

[22] Kundan Kumar, Rithesh Kumar, Thibault De Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre De Brebisson, Yoshua Bengio, and Aaron C Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in neural information processing systems*, 32, 2019.

[23] 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.

[24] Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, et al. Voicebox: Text-guided multilingual universal speech generation at scale. *Advances in neural information processing systems*, 36, 2024.

[25] Sang-gil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon. BigVGAN: A universal neural vocoder with large-scale training. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=iTtGCMDEzS_.

[26] Sang-gil Lee, Zhifeng Kong, Arushi Goel, Sungwon Kim, Rafael Valle, and Bryan Catanzaro. Etta: Elucidating the design space of text-to-audio models. *arXiv preprint arXiv:2412.19351*, 2024.

[27] Sang-Hoon Lee, Seung-Bin Kim, Ji-Hyun Lee, Eunwoo Song, Min-Jae Hwang, and Seong-Whan Lee. Hierspeech: Bridging the gap between text and speech by hierarchical variational inference using self-supervised representations for speech synthesis. *Advances in Neural Information Processing Systems*, 35:16624–16636, 2022.

[28] Sang-Hoon Lee, Ha-Yeong Choi, and Seong-Whan Lee. Accelerating high-fidelity waveform generation via adversarial flow matching optimization. *arXiv preprint arXiv:2408.08019*, 2024.

[29] Sang-Hoon Lee, Ha-Yeong Choi, Seung-Bin Kim, and Seong-Whan Lee. Hierspeech++: Bridging the gap between semantic and acoustic representation of speech by hierarchical variational inference for zero-shot speech synthesis. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

[30] Sang-Hoon Lee, Ha-Yeong Choi, and Seong-Whan Lee. Periodwave: Multi-period flow matching for high-fidelity waveform generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tQ1PmLfPBL.

[31] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[32] Peng Liu, Dongyang Dai, and Zhiyong Wu. RFWave: Multi-band rectified flow for audio waveform reconstruction. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=gRmWtOnTLK.

[33] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[34] Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized transformer with representation learning on the hypersphere. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=se4vjm7h4E.

[35] Max Morrison, Rithesh Kumar, Kundan Kumar, Prem Seetharaman, Aaron Courville, and Yoshua Bengio. Chunked autoregressive GAN for conditional waveform synthesis. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=v3aeIsY_vVX.

[36] Takuma Okamoto, Haruki Yamashita, Yamato Ohtani, Tomoki Toda, and Hisashi Kawai. Wavenext: Convnext-based fast neural vocoder without istft layer. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1–8. IEEE, 2023.

[37] Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International conference on machine learning*, pages 3918–3926. PMLR, 2018.

[38] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[39] Julian D Parker, Anton Smirnov, Jordi Pons, CJ Carr, Zack Zukowski, Zach Evans, and Xubo Liu. Scaling transformers for low-bitrate high-quality speech coding. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4YpMrGfldX.

[40] Wei Ping, Kainan Peng, and Jitong Chen. Clarinet: Parallel wave generation in end-to-end text-to-speech. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HklY120cYm.

[41] K R Prajwal, Bowen Shi, Matthew Le, Apoorv Vyas, Andros Tjandra, Mahi Luthra, Baishan Guo, Huiyu Wang, Triantafyllos Afouras, David Kant, and Wei-Ning Hsu. Musicflow: Cascaded flow matching for text guided music generation. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=kOczKjmYum.

[42] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[43] David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling diffusion models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=a9bzTv9SzO.

[44] Takaaki Saeki, Detai Xin, Wataru Nakata, Tomoki Koriyama, Shinnosuke Takamichi, and Hiroshi Saruwatari. Utmos: Utokyo-sarulab system for voicemos challenge 2022. *arXiv preprint arXiv:2204.02152*, 2022.

[45] Robin San Roman, Yossi Adi, Antoine Deleforge, Romain Serizel, Gabriel Synnaeve, and Alexandre Défossez. From discrete tokens to high-fidelity audio using multi-band diffusion. *Advances in neural information processing systems*, 36:1526–1538, 2023.

[46] Hubert Siuzdak. Vocos: Closing the gap between time-domain and fourier-based neural vocoders for high-quality audio synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vY9nzQmQBw.

[47] Christian J Steinmetz and Joshua D Reiss. auraloss: Audio focused loss functions in pytorch. In *Digital music research network one-day workshop (DMRN+ 15)*, 2020.

[48] Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Conditional flow matching: Simulation-free dynamic optimal transport. *arXiv preprint arXiv:2302.00482*, 2(3), 2023.

[49] A Vasuki and PT Vanathi. A review of vector quantization techniques. *IEEE Potentials*, 25(4): 39–47, 2006.

[50] Simon Welker, Matthew Le, Ricky T. Q. Chen, Wei-Ning Hsu, Timo Gerkmann, Alexander Richard, and Yi-Chiao Wu. Flowdec: A flow-based full-band general audio codec with high perceptual quality. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=uxDFlPGRLX.

[51] Haibin Wu, Naoyuki Kanda, Sefik Emre Eskimez, and Jinyu Li. TS3-Codec: Transformer-Based Simple Streaming Single Codec. In *Interspeech 2025*, pages 604–608, 2025. doi: 10.21437/Interspeech.2025-921.

[52] Yijing Wu, SaiKrishna Rallabandi, Ravisutha Srinivasamurthy, Parag Pravin Dakle, Alolika Gon, and Preethi Raghavan. Heysquad: A spoken question answering dataset, 2023.

[53] 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.

[54] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203. IEEE, 2020.

[55] 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.

[56] Jixun Yao, Yuguang Yan, Yu Pan, Ziqian Ning, Jiaohao Ye, Hongbin Zhou, and Lei Xie. Stablevc: Style controllable zero-shot voice conversion with conditional flow matching. *arXiv preprint arXiv:2412.04724*, 2024.

[57] Zhen Ye, Peiwen Sun, Jiahe Lei, Hongzhan Lin, Xu Tan, Zheqi Dai, Qiuqiang Kong, Jianyi Chen, Jiahao Pan, Qifeng Liu, et al. Codec does matter: Exploring the semantic shortcoming of codec for audio language model. *arXiv preprint arXiv:2408.17175*, 2024.

[58] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

[59] 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.

[60] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech. In *Proc. Interspeech 2019*, pages 1526–1530, 2019. doi: 10.21437/Interspeech.2019-2441.

[61] 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*, 2024.

[62] Jialong Zuo, Shengpeng Ji, Minghui Fang, Ziyue Jiang, Xize Cheng, Qian Yang, Wenrui Liu, Guangyan Zhang, Zehai Tu, Yiwen Guo, et al. Enhancing expressive voice conversion with discrete pitch-conditioned flow matching model. *arXiv preprint arXiv:2502.05471*, 2025.

A Implementation Details for Streaming Models

We describe the hyperparameter details of StreamFlow for streaming models at Table 7. 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 7: 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
	Context Prompt Ratio γ	3	3	3	3
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 8. 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 8: 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 9 provides additional objective evalutation 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 9: 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 [9], which is the popular neural audio codec using RVQ and adversarial training.[4] 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 [46], which is an iSTFT-based waveform generation model with adversarial training.[5] 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 comparision, 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) [45] as a strong baseline for EnCodec token reconstruction.[6] 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 [32], which a strong baseline using conditional flow matching for multi-band parallel generation.[7] 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 Mimi[8] from Moshi [7]. 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.

[4] https://github.com/facebookresearch/encodec
[5] https://github.com/gemelo-ai/vocos
[6] https://github.com/facebookresearch/audiocraft
[7] https://github.com/bfs18/rfwave
[8] 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 combinded 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

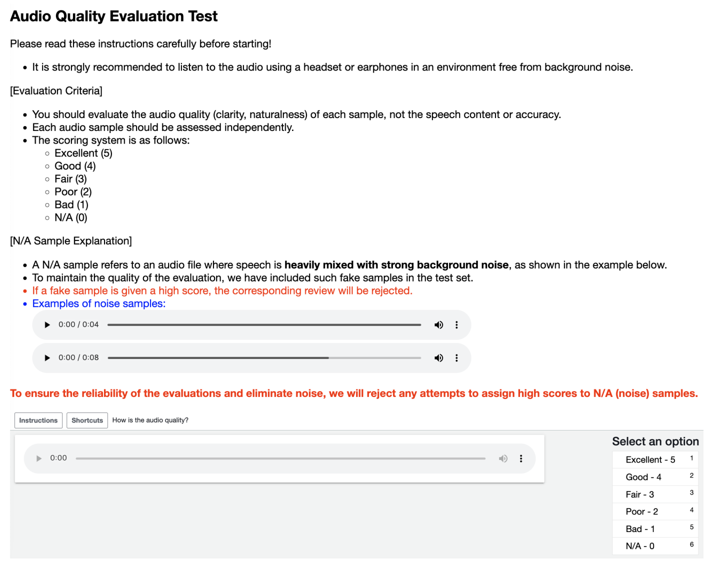
M-STFT We employed the multi-resolution Short-Time Fourier Transform (M-STFT) distance implemented in the open-source Auraloss [47].[9] Originally proposed in Parallel WaveGAN [54], the M-STFT quantifies the distances between ground-truth and generated audio samples across multiple STFT resolutions, 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) metric [10]. 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 [35] regarding the perceptual degradation caused by periodicity artifacts, we measured periodicity errors using the Periodicity Root Mean Square Error (RMSE)[11]. 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 [17]. We used the pytorch implementation of CREPE.[12]

UTMOS To assess the naturalness of the generated samples, we utilized the open-source Mean Opinion Score (MOS) prediction model, UTMOS [44].[13] 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.

[9] https://github.com/csteinmetz1/auraloss
[10] https://github.com/ludlows/PESQ
[11] https://github.com/descriptinc/cargan
[12] https://github.com/maxrmorrison/torchcrepe
[13] https://github.com/tarepan/SpeechMOS


Audio Quality Evaluation Test

Please read these instructions carefully before starting!

- It is strongly recommended to listen to the audio using a headset or earphones in an environment free from background noise.

[Evaluation Criteria]

- You should evaluate the audio quality (clarity, naturalness) of each sample, not the speech content or accuracy.
- Each audio sample should be assessed independently.
- The scoring system is as follows:
  - Excellent (5)
  - Good (4)
  - Fair (3)
  - Poor (2)
  - Bad (1)
  - N/A (0)

[N/A Sample Explanation]

- A N/A sample refers to an audio file where speech is **heavily mixed with strong background noise**, as shown in the example below.
- To maintain the quality of the evaluation, we have included such fake samples in the test set.
- If a fake sample is given a high score, the corresponding review will be rejected.
- Examples of noise samples:

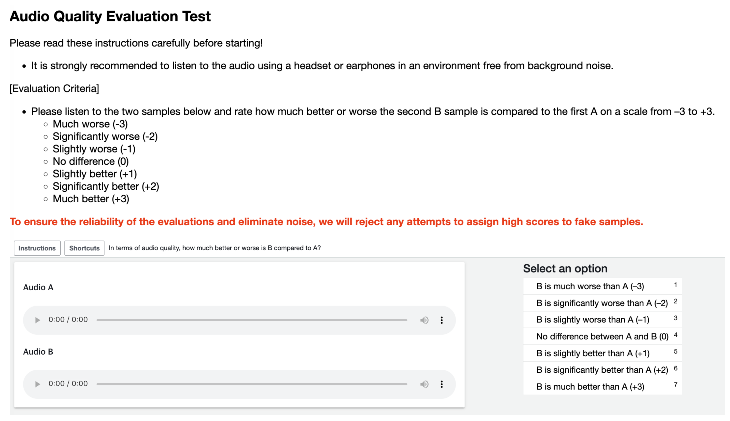
To ensure the reliability of the evaluations and eliminate noise, we will reject any attempts to assign high scores to N/A (noise) samples.

How is the audio quality?

Select an option

Excellent - 5	1
Good - 4	2
Fair - 3	3
Poor - 2	4
Bad - 1	5
N/A - 0	6

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


Audio Quality Evaluation Test

Please read these instructions carefully before starting!

- It is strongly recommended to listen to the audio using a headset or earphones in an environment free from background noise.

[Evaluation Criteria]

- Please listen to the two samples below and rate how much better or worse the second B sample is compared to the first A on a scale from −3 to +3.
  - Much worse (-3)
  - Significantly worse (-2)
  - Slightly worse (-1)
  - No difference (0)
  - Slightly better (+1)
  - Significantly better (+2)
  - Much better (+3)

To ensure the reliability of the evaluations and eliminate noise, we will reject any attempts to assign high scores to fake samples.

In terms of audio quality, how much better or worse is B compared to A?

Audio A

Audio B

Select an option

B is much worse than A (−3)	1
B is significantly worse than A (−2)	2
B is slightly worse than A (−1)	3
No difference between A and B (0)	4
B is slightly better than A (+1)	5
B is significantly better than A (+2)	6
B is much better than A (+3)	7

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)[14]. 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.

[14] https://www.mturk.com/

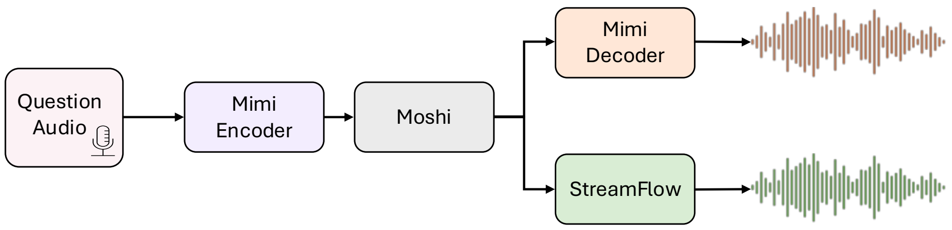


Figure 7: Inference pipeline comparing the Mimi decoder and the proposed StreamFlow, given identical Moshi output.

Table 10: 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 code[15], and conducted experiments using question audio from HeySQuAD [52][16] 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 10, our proposed StreamFlow consistently outperformed the Mimi decoder.

As shown in Figure 7, the pipeline of [Question Audio $\rightarrow$ Mimi Encoder$\rightarrow$ LLM $\rightarrow$ 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.

[15]https://github.com/kyutai-labs/moshi/blob/main/moshi/moshi/run_inference.py
[16]https://huggingface.co/datasets/yijingwu/HeySQuAD_human

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly highlight the main contributions of the proposed method in the abstract and introduction, which are well aligned with the main content of the paper. The scope of our contributions is described accurately without exaggeration, and necessary limitations and assumptions are appropriately acknowledged.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitation of our works in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We present the prior theory, our proposed theoretical framework, and the corresponding results using equations and tables, with clear explanations and cross-references.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We clearly describe the model training and inference procedures, network architecture, detailed implementation specifics, datasets, and training settings throughout the main text and appendix. We also plan to release the code and model after the paper is accepted. Even though the full code is not yet provided, the detailed architecture is illustrated in figures to enable reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We will release all source code and checkpoints after paper notification.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We used the open dataset, LibriTTS. We describe the dataset details, and hyper-parameter, and type of optimizer.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We reported the MOS with confidence interval.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe the computation resources and training time.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We all read the Code of Ethics provided in above link.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss both potential positive societal impacts and negative societal impacts in Section 5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper does not involve the release of any models or datasets that pose a high risk of misuse. Therefore, safeguards are not applicable in this context.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We will release the source code and check point under the license of CC-BY-NC 4.0.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release the source code and check point, and include details about training, license, limitations, etc.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: We described the Crowdsourcing details in Appendix.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We only used Amazon Mturk for evaluation.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We do not use LLMs for research.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.