Speak Early: Speculative Speech Generation for Low-Latency SpeechLLMs

Tianyuan Jiang
Tsinghua University
Beijing, China
jty23@mails.tsinghua.edu.cn

Yongheng Deng*
Tsinghua University
Beijing, China
dengyh1013@gmail.com

Ju Ren
Tsinghua University
Beijing, China
renju@tsinghua.edu.cn

Abstract

Speech large language models (SpeechLLMs) have emerged as a new interface for human–AI interaction, enabling real-time spoken responses. Time-to-First-Speech (TTFS) is a critical latency metric in such interactive SpeechLLM systems, directly determining perceived responsiveness. While prior work reduces TTFS by accelerating autoregressive decoding, we show that diffusion-based speech synthesis contributes a comparable portion of the latency. This reveals a fundamental limitation of existing approaches that optimize decoding alone, and indicates that optimizing either LLM decoding or speech synthesis in isolation is insufficient. This paper proposes SpecSpeak, a new SpeechLLM inference paradigm that overlaps LLM decoding with speech synthesis to reduce the TTFS latency. SpecSpeak introduces a lightweight draft model to generate a prefix quickly. Instead of waiting for fully verified tokens like speculative decoding, SpecSpeak triggers speech synthesis early using draft tokens, while a target model performs parallel verification and correction. To handle inconsistencies, SpecSpeak introduces a verification-aware reuse mechanism that adaptively preserves intermediate diffusion states based on prefix agreement, enabling efficient correction without restarting synthesis. Extensive experiments demonstrate that SpecSpeak reduces the TTFS latency of SpeechLLMs by 18%–30%, while preserving response correctness and speech quality. The code is available at https://github.com/jtyissix/specspeak.

1 Introduction

Speech is becoming a primary interface for human-computer interaction. Intelligent speech systems can significantly enhance both interaction efficiency and user experience. Recent advances in large language models (LLMs) have further accelerated this trend, giving rise to SpeechLLM systems that unify language understanding and speech generation within a single framework[1–3]. Such systems are rapidly becoming the backbone of interactive AI, enabling real-time voice assistants, conversational agents, and embodied intelligence[4–7].

Traditional speech systems typically adopt a cascaded architecture that decomposes the pipeline into automatic speech recognition (ASR), language modeling, and text-to-speech (TTS)[8–11]. While modular, such designs suffer from cumulative latency and error propagation across components,

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

limiting their effectiveness[12–14]. To address these issues, recent work has moved toward end-to-end SpeechLLM architectures, where speech inputs are directly processed by an LLM and outputs are generated as speech tokens or intermediate representations, followed by neural speech synthesis modules such as diffusion models[15–18]. This unified design significantly improves the intelligence and naturalness of the generated speech.

In interactive SpeechLLM systems, users are particularly sensitive to the delay before hearing the first audio response. We refer to this delay as *Time-to-First-Speech* (TTFS), which directly determines the perceived responsiveness of the system and is particularly critical to user experience. To reduce latency, existing efforts have primarily focused on accelerating LLM token generation through techniques such as multi-token prediction (MTP)[14, 19]. While effective for improving decoding efficiency, this paper reveals that LLM generation is not the only dominant bottleneck in end-to-end SpeechLLM systems.

Modern SpeechLLM systems increasingly adopt diffusion-based synthesizers due to their superior ability to generate natural and expressive speech[17, 18, 15]. However, as shown in Figure 1, diffusion-based speech synthesis contributes comparable, and often dominant latency to TTFS. More importantly, these two computationally intensive stages execute sequentially, resulting in a long *combined critical path*. As a result, optimizing either component in isolation is incomplete.

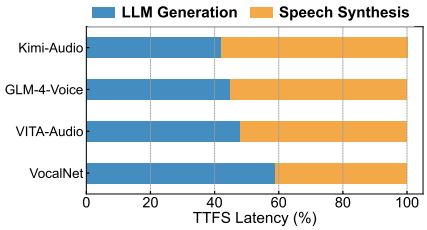


Figure 1: TTFS latency breakdown in SpeechLLMs. Speech synthesis accounts for a substantial fraction of TTFS.

This observation prompts a fundamental question: *can we break the sequential dependency between LLM generation and diffusion synthesis to reduce TTFS?* A natural idea is to pipeline the two stages and overlap their execution. However, this is non-trivial due to a fundamental tension between early execution and output correctness. As shown in Figure 2, we observe that diffusion-based speech synthesis becomes effective only after a sufficiently long token prefix is available. Therefore, starting speech synthesis early with an incomplete prefix can lead to degraded speech quality, whereas delaying synthesis until the prefix stabilizes limits the achievable overlap and reduces the potential for latency reduction.

To address this challenge, we propose `SpecSpeak`, a speculative co-generation framework that overlaps LLM decoding with speech synthesis to reduce TTFS. The key idea is to incorporate a lightweight draft model and trigger speech synthesis early using speculative draft tokens, while performing parallel verification and correction with a stronger model. Crucially, `SpecSpeak` enables efficient correction without restarting synthesis. Based on the agreement between draft and verified tokens, it adaptively reuses intermediate diffusion states when the prefix is sufficiently accurate, and selectively revises them otherwise. This design allows verification and correction to proceed in parallel with synthesis, forming an iterative co-generation process that progressively refines both text and audio outputs. By transforming the conventional sequential pipeline into an overlapping and adaptive execution process, `SpecSpeak` effectively shortens the critical path that determines TTFS.

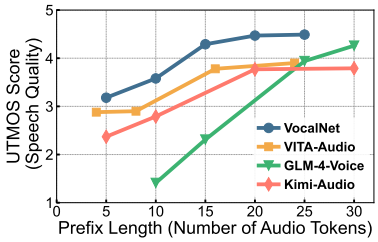
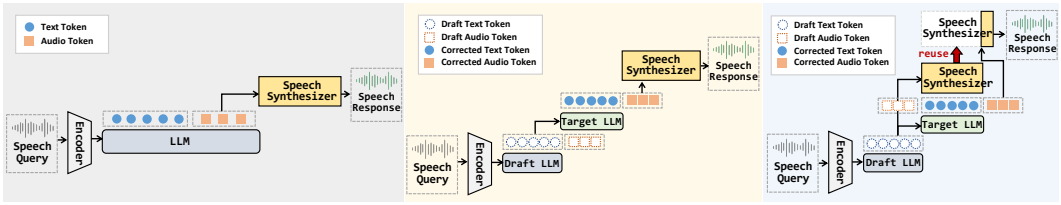


Figure 2: Impact of prefix length on synthesized speech quality.

We conduct extensive experiments to evaluate the effectiveness of `SpecSpeak` across multiple SpeechLLM architectures and benchmarks. Our evaluation covers four speech-based question answering datasets, measuring TTFS latency, response correctness, and perceptual speech quality. Results show that `SpecSpeak` consistently reduces the TTFS latency by over 30% compared to VocalNet [14] and VITA-Audio [19], while also outperforming Speculative Decoding by 18%–23% in average latency. Notably, these gains are achieved without compromising output quality, with accuracy differences within 1% and negligible changes in WER and perceptual scores.

We summarize our contributions as follows: (1) We identify TTFS as a critical-path bottleneck in SpeechLLM systems and reveal the limitation of LLM-centric acceleration methods. (2) We propose a speculative co-generation framework that breaks the sequential dependency between LLM decoding and speech synthesis. (3) We introduce a verification-aware reuse mechanism that enables efficient



(a) SpeechLLM Inference Pipeline (b) Acceleration with Speculative Decoding (c) Acceleration with SpecSpeak

Figure 3: Comparison of inference pipelines for SpeechLLMs. (a) Conventional SpeechLLM inference is strictly sequential, where speech synthesis is triggered only after full token generation. (b) Speculative decoding introduces a draft–verify paradigm that accelerates token generation but still maintains a sequential synthesis pipeline. (c) SpecSpeak (ours) enables early synthesis with draft tokens in parallel with speculative verification and selective diffusion-state reuse, allowing decoding and speech generation to overlap in a unified co-generation framework.

correction under speculative execution. (4) We demonstrate through extensive experiments that our method significantly reduces TTFS while preserving output speech quality.

2 Problem Formulation and Related Work

SpeechLLM Inference Pipeline. Modern SpeechLLM systems unify language modeling and speech generation within a single end-to-end framework [20–22]. Despite variations in architectural design, their inference procedures share a common computational structure: an LLM first produces text and speech tokens, which are subsequently consumed by a downstream speech synthesizer to generate the final waveform, as illustrated in Figure 3 (a).

Formally, given an input speech query $\mathbf{x}^s$, a *preprocessing module* encodes it into latent representations: $\mathbf{z} = (z_1, z_2, \ldots, z_n) = \mathcal{E}(\mathbf{x}^s)$. Conditioned on $\mathbf{z}$, the *multimodal LLM* autoregressively generates text and/or speech tokens $(a_k^{\mathrm{t}}, a_k^{\mathrm{s}}) = \mathcal{M}(\mathbf{a}_{<k}^{\mathrm{t}}, \mathbf{a}_{<k}^{\mathrm{s}}, \mathbf{z})$, where a^{t} and a^{s} denote text and speech tokens, respectively. Existing SpeechLLM architectures differ in how these tokens are generated, including cascaded approaches that first produce text tokens followed by speech tokens [12, 23, 14], and unified approaches that interleave text and speech token generation [13, 19]. The generated speech tokens are then fed into a *speech synthesizer* to produce the final waveform $\mathbf{y}$ through a conditional diffusion-based acoustic modeling process. Specifically, speech tokens $\mathbf{a}^s$ serve as conditioning signals for diffusion. Starting from Gaussian noise $\Gamma_1 \sim \mathcal{N}(\mathbf{0}, \mathcal{I})$, the model iteratively refines the acoustic representation via: $\Gamma_{k+1} = \mathcal{V}(\Gamma_k \mid \mathbf{a}^s)$, where $\mathcal{V}$ denotes one step of the conditional diffusion denoising process. After n_d iterations, we obtain the final acoustic representation Γ_{n_d}, which is then converted into the final waveform using a neural vocoder such as HiFi-GAN [24]: $\mathbf{y} = \mathrm{Vocoder}(\Gamma_{n_d})$.

Objective. This paper aims to minimize the latency to produce the first audible response, referred to as *Time-to-First-Speech* (TTFS) [13, 19, 23, 25]. The TTFS can be decomposed into the time required to generate a prefix of speech tokens and the time to synthesize the corresponding waveform:

$$\mathrm{TTFS} = t_{\mathrm{LLM}}(k) + t_{\mathrm{synth}}(k), \tag{1}$$

where k denotes the minimum number of speech tokens required to initiate stable speech synthesis, $t_{\mathrm{LLM}}(k)$ denotes the time required for the LLM to autoregressively generate k speech tokens, and $t_{\mathrm{synth}}(k)$ denotes the time to synthesize corresponding speech segment conditioned on these tokens.

Existing SpeechLLM Acceleration Approaches. Existing efforts to reduce latency in SpeechLLM systems are relatively limited and primarily focus on optimizing the token generation process within the LLM. Several recent works aim to accelerate autoregressive decoding by reducing the number of forward passes required to generate speech tokens. For instance, VITA-Audio [19] and VocalNet [14] introduce multi-token prediction (MTP) mechanisms that enable the model to generate multiple speech tokens in a single forward step, thereby improving decoding efficiency. VocalNet-MDM [25] further incorporates masked diffusion modeling within the speech decoder to enhance generation efficiency. These approaches effectively reduce the latency of producing the initial speech token sequence by accelerating LLM-side computation. Beyond LLM-side acceleration, speculative decoding [26–28] has been widely explored to improve autoregressive generation efficiency in language models. As

shown in Figure 3 (b), this paradigm introduces a lightweight draft model to generate candidate tokens, which are then verified by a stronger target model, effectively reducing the decoding latency. Another line of work explores reducing the generation burden by lowering the effective speech token rate. For example, Fun-Audio-Chat [29] generates low-frame-rate speech tokens within the LLM and reconstructs full-resolution tokens using a lightweight model, thereby decreasing the number of tokens that need to be generated and processed. Despite these efforts, existing approaches share a common **limitation**: they focus exclusively on optimizing the token generation stage within the LLM. However, optimizing LLM generation alone is insufficient, as speech synthesis remains on the critical path and contributes comparable latency to TTFS.

3 Methodology

We propose `SpecSpeak`, a speculative co-generation framework that overlaps LLM decoding with speech synthesis to reduce TTFS. As illustrated in Figure 3 (c), a lightweight draft model produces an early text prefix to trigger synthesis, while a target model verifies tokens in parallel, breaking the sequential dependency in conventional SpeechLLM pipelines. To handle inconsistencies between draft and verified tokens, we introduce a verification-aware reuse strategy: diffusion states are reused when the draft prefix is sufficiently accurate, and otherwise discarded and recomputed with refined tokens. This design enables parallel generation with selective reuse, effectively shortening the critical path of speech generation.

3.1 Drafting and Early Triggering

In `SpecSpeak`, we introduce a lightweight draft model $\mathcal{M}_q$ along with the SpeechLLM $\mathcal{M}_p$ serving as the high-quality target model. The draft model is used to rapidly generate text tokens, while the target model is responsible for verification. Speech synthesis can be early triggered after a sufficiently long sequence of draft text tokens has been produced. However, we observe that the resulting speech segment often does not consume all of these tokens in SpeechLLMs. This observation provides an opportunity to trigger speech synthesis even earlier. Therefore, we introduce a triggering threshold α, representing the minimal number of text tokens required to start synthesis. Once the draft model $\mathcal{M}_q$ generates a prefix $\mathbf{a}^{\mathrm{t}}_{1:\alpha}$, we trigger a speech synthesis process denoted as $\mathcal{V}_i$, using these text tokens along with their corresponding speech tokens $\mathbf{a}^{s}$. Meanwhile, the target model $\mathcal{M}_p$ runs in parallel to verify the generated text tokens. This decouples synthesis from full-sequence generation and enables early-stage co-generation.

3.2 Speculative Verification

To ensure correctness under early triggering, we introduce a verification process that runs in parallel with synthesis. This process validates the draft text tokens generated by $\mathcal{M}_q$ using the target model $\mathcal{M}_p$. Given a draft prefix $\mathbf{a}^{\mathrm{t}}_{1:\alpha}$ generated by the draft model $\mathcal{M}_q$, the target model $\mathcal{M}_p$ performs token-level validation using a standard speculative decoding procedure, which probabilistically (controlled by threshold δ) accepts or corrects each token based on the agreement between the two models [26]. More details are provided in Appendix D. Rather than focusing on the verification mechanism itself, we summarize its outcome using a single quantity, α_{acc}, which denotes the number of tokens in the prefix that are consistent with the target model. Intuitively, α_{acc} captures the reliability of the draft prefix.

3.3 Correction and Diffusion Reuse

Early triggering enables parallel synthesis, but also introduces a key challenge: the draft tokens used to condition the synthesizer may differ from the final verified tokens. A naive solution is to discard all intermediate synthesis results after verification. However, this can significantly eliminate the benefits of overlapping computation. To address this, we observe that diffusion-based speech synthesis is an iterative refinement process, where intermediate states progressively capture coarse-to-fine acoustic structures. When the conditioning tokens are only slightly perturbed, the resulting diffusion trajectory typically remains close to a valid path, and can be corrected in subsequent refinement steps.

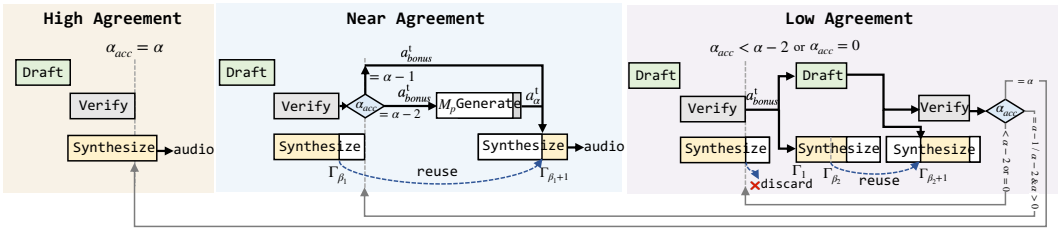


Figure 4: Co-Generation Pipeline of SpecSpeak. The inference strategy is dynamically adapted based on the agreement between draft and target models, measured by the number of verified tokens α_{acc}. *High agreement*: all draft tokens are accepted, and diffusion synthesis proceeds without modification. *Near agreement*: most tokens are correct, enabling reuse of intermediate diffusion states with minimal adjustment. *Low agreement*: significant mismatch triggers redraft and regeneration.

Based on this observation, we use α_{acc}—the number of verified tokens—as a proxy for the agreement between draft and target models, and partition the synthesis behavior into three regimes, as illustrated in Figure 4:

High agreement. When $\alpha_{\text{acc}} = \alpha$, all draft tokens are accepted. In this case, the original conditioning remains valid, and the ongoing synthesis trajectory is fully consistent with the final sequence. Therefore, the synthesis process $\mathcal{V}_i$ proceeds without modification.

Near agreement. When $\alpha_{\text{acc}} = \alpha - 1$ or $\alpha - 2$ and $\alpha_{\text{acc}} > 0$, the draft prefix is largely correct, and only 1-2 tokens require correction. In this regime, the induced perturbation to the conditioning is limited, and the current diffusion trajectory remains within a stable region. Therefore, we reuse the intermediate diffusion state and continue synthesis under the corrected conditioning, avoiding redundant computation.

Based on α_{acc}, we construct the corrected prefix $\tilde{\mathbf{a}}^{\text{t}}_{1:\alpha}$ as follows. When $\alpha_{\text{acc}} = \alpha - 1$, only the last token is rejected. In this case, we can directly obtain a corrective token $a^{\text{t}}_{\text{bonus}}$ (i.e., bonus token [26]) from the target model during the verification process at position α, after which the corrected prefix is obtained by appending this token: $\tilde{\mathbf{a}}^{\text{t}}_{1:\alpha} = \text{Concat}(\mathbf{a}^{\text{t}}_{1:(\alpha-1)}, a^{\text{t}}_{\text{bonus}})$. When $\alpha_{\text{acc}} = \alpha - 2$, we first incorporate the bonus token to form $\tilde{\mathbf{a}}^{\text{t}}_{1:(\alpha-1)} = \text{Concat}(\mathbf{a}^{\text{t}}_{1:(\alpha-2)}, a^{\text{t}}_{\text{bonus}})$, and then directly generate the final token using the target model:

$$a^{\text{t}}_{\alpha} = \mathcal{M}_p(\tilde{\mathbf{a}}^{\text{t}}_{1:(\alpha-1)}). \tag{2}$$

Then, $\tilde{\mathbf{a}}^{\text{t}}_{1:\alpha} = \text{Concat}(\tilde{\mathbf{a}}^{\text{t}}_{1:(\alpha-1)}, a^{\text{t}}_{\alpha})$. In both cases, we obtain a complete corrected prefix $\tilde{\mathbf{a}}^{\text{t}}_{1:\alpha}$, which is subsequently mapped to speech tokens $\tilde{\mathbf{a}}^{\text{s}}$.

During verification, the synthesis process $\mathcal{V}_i$ has already progressed to an intermediate diffusion state Γ_{β_1} under the draft conditioning. Since the mismatch is limited to at most two tokens, the induced perturbation remains small, and the current trajectory stays within a locally stable region. Therefore, we reuse the intermediate state Γ_{β_1} and continue the diffusion process under the corrected conditioning:

$$\Gamma_{\beta_1+1} \sim \mathcal{V}_i(\Gamma_{\beta_1} \mid \tilde{\mathbf{a}}^{\text{s}}), \ \ldots, \Gamma_{n_d} \sim \mathcal{V}_i(\Gamma_{n_d-1} \mid \tilde{\mathbf{a}}^{\text{s}}). \tag{3}$$

Low agreement. When $\alpha_{\text{acc}} < \alpha - 2$ or $\alpha_{\text{acc}} = 0$, the draft prefix significantly deviates from the target model. In this case, the corresponding speech tokens provide insufficient and inaccurate conditioning for the diffusion model. As a result, the generated trajectory deviates substantially from the correct path, and cannot be effectively corrected in later refinement steps. Therefore, the intermediate state Γ_{β_1} produced under draft conditioning is discarded, and synthesis is restarted from scratch to ensure correctness. Instead of immediately switching to full target generation, which is computationally expensive, we continue speculative drafting to progressively refine the prefix. Starting from the accepted tokens $\mathbf{a}^{\text{t}}_{1:\alpha_{\text{acc}}}$, the verification step provides an additional bonus token, extending the prefix to $\mathbf{a}^{\text{t}}_{1:(\alpha_{\text{acc}}+1)}$, which serves as a more reliable starting point for subsequent drafting. The draft model then continues generation:

$$\mathbf{a}^{\text{t}}_{(\alpha_{acc}+2):\alpha} = \mathcal{M}_q(\mathbf{a}^{\text{t}}_{1:(\alpha_{\text{acc}}+1)}). \tag{4}$$

Meanwhile, to maximize parallelism and effort of speculative diffusion, the extended prefix $\mathbf{a}^{\text{t}}_{1:(\alpha_{\text{acc}}+1)}$ is mapped to speech tokens $\mathbf{a}^{\text{s}}_{trial}$ to initiate a new synthesis process $\mathcal{V}_{i+1}$. Since the last conditioning

remains unreliable, the new synthesis process is initialized from scratch:

$$\Gamma_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \Gamma_2 \sim \mathcal{V}_{i+1}(\Gamma_1 \mid \mathbf{a}^{\mathrm{s}}_{trial}), \ \ldots, \Gamma_{n_d} \sim \mathcal{V}_{i+1}(\Gamma_{n_d-1} \mid \mathbf{a}^{\mathrm{s}}_{trial}). \tag{5}$$

Once generating $\mathbf{a}^{\mathrm{t}}_{(\alpha_{acc}+2):\alpha}$, verification is performed in parallel with a new synthesis task $\mathcal{V}_{i+2}$. As the prefix becomes progressively more accurate with the aid of bonus token, the conditioning quality improves to a level where reuse becomes reliable again. In subsequent synthesis stages, intermediate states can again be reused; however, to mitigate the risk introduced by earlier unreliable conditioning, the reuse is performed from earlier diffusion steps (denoted by Γ_{β_2} with $\beta_2 < \beta_1$), followed by continued denoising conditioned on the updated speech tokens $\mathbf{a}^{\mathrm{s}}$ (mapped from the redraft prefix $\mathbf{a}^{\mathrm{t}}_{1:\alpha}$):

$$\Gamma_{\beta_2+1} \sim \mathcal{V}_{i+2}(\Gamma_{\beta_2} \mid \mathbf{a}^{\mathrm{s}}), \ \ldots, \Gamma_{n_d} \sim \mathcal{V}_{i+2}(\Gamma_{n_d-1} \mid \mathbf{a}^{\mathrm{s}}). \tag{6}$$

At the same time, parallel verification recomputes α_{acc} for the redraft prefix, which again serves as a signal to dynamically select the subsequent execution regime, forming an iterative refinement loop. In this way, the co-generation process proceeds iteratively until a complete prefix of length α is established, upon which the corresponding audio segment is synthesized. This design avoids committing to unreliable trajectories while preserving efficiency. By incorporating verification-derived corrections (i.e., bonus tokens) and continuing speculative drafting, the system gradually improves conditioning quality and enables a smooth transition back to reusable diffusion states, resulting in a more conservative yet stable reuse strategy.

4 Experiments

4.1 Experimental Setup

Datasets. Following prior SpeechLLM work [25, 1, 14, 30, 23], we evaluate on the English subsets of OpenAudioBench [1], including **TriviaQA** [31], **LLaMA Questions** [32], **WebQuestions** [33], and **AlpacaEval** [34]. These datasets are organized as speech-based question answering benchmarks, where each sample consists of a spoken query and requires the model to generate a spoken response. Such settings closely reflect real-world human–AI interaction scenarios, where users communicate via speech and responsiveness is critical, making TTFS latency a key evaluation factor. Additional dataset details are provided in Appendix B.

Baselines. Existing SpeechLLM acceleration methods primarily optimize model architectures. For example, **VocalNet** [14] and **VITA-Audio** [19] leverage multi-token prediction (MTP) to improve decoding efficiency. Another line of work explores model-agnostic acceleration via **Speculative Decoding (SD)** [26]. Our method falls into this category and can be applied without modifying model architectures. Therefore, we adopt VocalNet (8B) and VITA-Audio (7B) as base models, representing two canonical SpeechLLM paradigms: VocalNet follows a cascaded pipeline that first generates text tokens and then speech tokens, while VITA-Audio employs a unified backbone that interleaves text and speech token generation. On top of these base models, we compare standard speculative decoding and our proposed method `SpecSpeak`. We also compare `SpecSpeak` with the base model using fewer diffusion steps in Appendix G.5. Details of the base models and speculative decoding are provided in Appendix C and Appendix D.

Implementation Details. VocalNet provides both 1B and 8B checkpoints[2], therefore, we use the 1B model as the draft model and the 8B model as the target model. For VITA-Audio, only a 7B model is publicly available[3], which we use as the target model, and we train an additional 0.5B model as the draft model. Training details of the draft model and the model configurations are included in Appendix F. For `SpecSpeak`, we set hyperparameters to $\alpha = 3, \beta_1 = 4, \beta_2 = 3$ for VocalNet, and $\alpha = 2, \beta_1 = 4, \beta_2 = 2$ for VITA-Audio. The probability threshold δ is randomly sampled from $[0, 1]$ following [26]. These serve as default settings in the main experiments, and sensitivity analyses with fixed δ values and other hyperparameters are provided in Section 4.4. All experiments are conducted on a single NVIDIA A800 GPU. For the TTFS latency evaluation, each sample is measured 10 times and we report the average.

[2] `https://huggingface.co/VocalNet/VocalNet-1B`, `https://huggingface.co/VocalNet/VocalNet-8B`

[3] `https://huggingface.co/collections/VITA-MLLM/vita-audio`

Evaluation Metrics. We evaluate three aspects: **(1) TTFS Latency.** We report the *average*, *median (P50)*, and *90th percentile (P90)* TTFS latency. The average latency reflects overall latency efficiency, while P50 captures the typical user experience. P90 measures tail latency, which is critical in interactive systems where occasional delays significantly impact perceived responsiveness. **(2) Response Correctness.** Following prior work [14, 25], we adopt an LLM-as-a-judge paradigm using Qwen3-30B-A3B [35]. We report *accuracy* on TriviaQA, WebQuestions, and LLaMA Questions, and evaluate AlpacaEval responses with *normalized scores* in the range of 1 to 10. **(3) Speech Quality.** We assess speech-text alignment using *word error rate (WER)* computed by Qwen3-ASR-1.7B [36]. In addition, we use *UTMOS* [37] to estimate mean opinion scores (MOS), reflecting perceptual audio quality. We further provide qualitative analysis on speech quality in Appendix G.6. The prompts and details for LLM-based evaluation are provided in Appendix H.

4.2 Main Results

We compare `SpecSpeak` with the baselines across four datasets. Table 1 reports TTFS latency, and Table 2 summarizes response correctness and speech quality.

Table 1: TTFS latency (ms) comparison across datasets. We report average, median (P50), and tail latency (P90). `SpecSpeak` consistently reduces latency over both base models (VocalNet and VITA-Audio) and Speculative Decoding (SD).

Dataset (→)	**TriviaQA**			**LLaMA Questions**			**WebQuestions**			**AlpacaEval**		
Method (↓)	Avg. ↓	P50 ↓	P90 ↓	Avg. ↓	P50 ↓	P90 ↓	Avg. ↓	P50 ↓	P90 ↓	Avg. ↓	P50 ↓	P90 ↓
VocalNet	311.82	309.31	335.57	275.00	271.56	278.87	314.39	273.58	369.55	311.90	287.64	347.52
VocalNet+SD	272.24	254.84	345.78	246.52	246.04	276.81	257.03	256.35	321.91	277.28	271.88	321.43
VocalNet+`SpecSpeak`	**199.12**	**173.42**	**285.29**	**191.75**	**172.56**	**255.24**	**212.18**	**183.18**	**314.00**	**216.15**	**172.99**	**272.02**
VITA-Audio	334.42	326.57	352.10	328.48	320.61	344.08	328.76	316.74	343.20	369.89	355.48	455.03
VITA-Audio+SD	273.30	261.49	347.82	270.83	235.53	343.34	290.29	269.93	350.88	295.95	272.65	364.15
VITA-Audio+`SpecSpeak`	**235.81**	**220.48**	**285.09**	**220.54**	**211.67**	**270.41**	**242.81**	**219.87**	**283.04**	**233.03**	**222.17**	**313.60**

Table 2: Response correctness and speech quality comparison. Metrics include accuracy/score (correctness), WER (alignment), and UTMOS (perceptual quality). `SpecSpeak` maintains comparable or better quality compared to base models (VocalNet and VITA-Audio) and Speculative Decoding (SD).

Dataset (→)	**TriviaQA**			**LLaMA Questions**			**WebQuestions**			**AlpacaEval**		
Method (↓)	Acc. ↑	WER ↓	UTMOS ↑	Acc. ↑	WER ↓	UTMOS ↑	Acc. ↑	WER ↓	UTMOS ↑	Score ↑	WER ↓	UTMOS ↑
VocalNet	**0.65**	**8.89**	**4.22**	0.58	**7.60**	**4.22**	0.48	7.54	**4.19**	3.45	5.58	**4.10**
VocalNet+SD	0.63	9.97	4.20	**0.59**	8.16	**4.22**	**0.49**	7.27	**4.19**	3.61	5.41	**4.10**
VocalNet+`SpecSpeak`	0.63	10.67	4.13	**0.59**	9.08	4.15	**0.49**	**7.17**	4.16	**3.70**	**5.39**	4.09
VITA-Audio	**0.28**	**4.05**	**3.76**	**0.46**	3.88	3.77	**0.25**	**3.50**	**3.72**	2.55	**9.88**	**3.56**
VITA-Audio+SD	**0.28**	4.61	3.71	0.45	3.71	**3.79**	**0.25**	4.67	3.69	**2.76**	10.32	**3.56**
VITA-Audio+`SpecSpeak`	0.27	4.77	3.71	0.43	**3.48**	**3.79**	0.24	4.19	3.67	2.66	10.54	3.52

TTFS Latency. Table 1 shows that applying `SpecSpeak` to VocalNet reduces average TTFS by 33%, P50 by 39%, and P90 by 15%. Compared to Speculative Decoding, `SpecSpeak` achieves further reductions of 23%, 32%, and 10% in average, P50, and P90 latency, respectively. Similarly, for VITA-Audio, `SpecSpeak` reduces average latency by 31%, P50 by 33%, and P90 by 22% over the base model, and outperforms Speculative Decoding by 18%, 15%, and 18%, respectively. These results demonstrate that `SpecSpeak` effectively shortens the critical path of TTFS.

Response Correctness and Speech Quality. As shown in Table 2, `SpecSpeak` maintains comparable performance to baselines in both correctness and speech quality. On average, accuracy decreases by only 0.8% compared to base models and 0.67% compared to Speculative Decoding. In some cases, `SpecSpeak` even achieves slight improvements, suggesting that draft-assisted generation can be beneficial for certain queries. Similarly, speech quality remains stable: UTMOS differences are consistently marginal across all settings, while WER shows only limited fluctuations. The results indicate that `SpecSpeak` preserves perceptual quality while remaining competitive in intelligibility.

Overall, these results demonstrate that SpecSpeak achieves significant TTFS latency reduction while maintaining high-quality speech generation.

4.3 Ablation Studies

Table 3: Ablation study on VITA-Audio (TriviaQA). We evaluate both coarse-grained and fine-grained variants. Removing key components consistently increases TTFS latency, while having minimal impact on correctness and speech quality, validating the effectiveness of SpecSpeak's design.

Method	TTFS Latency (ms)			Response Correctness and Quality		
	Avg ↓	P50 ↓	P90 ↓	Acc. ↑	WER ↓	UTMOS ↑
SpecSpeak	**235.81**	**220.48**	**285.09**	0.274	4.77	3.71
w/o Early Triggering	273.30 (↑ 16%)	261.49 (↑ 19%)	347.82 (↑ 22%)	0.276 (↑ 0.7%)	4.61 (↓ 3.3%)	3.71 (↓ 0.0%)
w/o Diffusion Reuse	271.41 (↑ 15%)	223.39 (↑ 1.3%)	303.42 (↑ 6.4%)	0.274 (↑ 0.0%)	4.61 (↓ 3.3%)	**3.72** (↑ 0.2%)
w/o Drafting	291.95 (↑ 24%)	272.10 (↑ 23%)	313.74 (↑ 10%)	**0.283** (↑ 3.2%)	4.74 (↓ 0.6%)	3.06 (↓ 18%)
Normal Early Triggering	279.51 (↑ 19%)	295.94 (↑ 34%)	319.35 (↑ 12%)	0.274 (↑ 0.0%)	**4.35** (↓ 8.8%)	3.71 (↓ 0.0%)
Draft-based Correction	245.30 (↑ 4.0%)	230.67 (↑ 4.6%)	305.03 (↑ 7.0%)	0.274 (↑ 0.0%)	4.69 (↓ 1.7%)	**3.72** (↑ 0.2%)
No Parallel Redraft Synthesis	252.42 (↑ 7.0%)	259.75 (↑ 18%)	287.87 (↑ 9.8%)	0.274 (↑ 0.0%)	4.77 (↓ 0.0%)	3.71 (↓ 0.0%)

We conduct ablation studies on VITA-Audio using the TriviaQA dataset to understand the contribution of each component in SpecSpeak. Table 3 summarizes the results. Ablation studies on other datasets are available in Appendix G.1.

Coarse-Grained Ablations. We first examine three key design choices at a coarse level: *(1) w/o Early Triggering.* Disables early triggering speech synthesis with draft tokens and starts diffusion only after verification and correction (equivalent to speculative decoding). *(2) w/o Diffusion Reuse.* Discards all intermediate synthesis results and restarts diffusion after verification from scratch. *(3) w/o Drafting.* Removes the draft model and relies solely on the target model for early triggering. Specifically, in VITA-Audio we start pre-diffusion for 4 steps when 2 speech tokens are available. Then, we complete the rest 6 steps of diffusion starting from previous intermediate state after all 8 speech tokens are generated.

Fine-Grained Design Ablations. We then analyze several finer-grained design decisions within SpecSpeak: *(4) Normal Early Triggering.* To evaluate the effectiveness of α in early triggering, this variant increases the threshold from $\alpha = 2$ to $\alpha = 4$, matching the default prefix length requirement of the base model before starting synthesis. *(5) Draft-based Correction.* For the near-agreement case, we replace the target-based correction with draft-based correction followed by verification, testing whether the target model is more efficient for the final rejected token generation. *(6) No Parallel Redraft Synthesis.* For the low-agreement case, we disable parallel synthesis during redrafting and instead restart synthesis only after the refined prefix is obtained.

Results. Table 3 shows that the TTFS latency increases significantly without largely changing the response correctness and speech quality across all ablations. The results confirm the effectiveness of the key components in SpecSpeak, and demonstrate the soundness of the system design.

4.4 Sensitivity Analysis

We analyze the impact of four key hyperparameters in SpecSpeak using VITA-Audio on TriviaQA, including the draft acceptance threshold δ, early triggering threshold α, and the diffusion reuse steps β_1 (near agreement) and β_2 (low agreement). Figure 5 reports average TTFS latency and accuracy (additional metrics are provided in Appendix G.2). We observe clear and consistent trade-offs across all parameters.

Impact of δ. As the acceptance threshold δ increases, latency rises monotonically while accuracy improves steadily. This is because a stricter acceptance criterion reduces the likelihood of accepting incorrect draft tokens, thereby improving correctness. However, it also leads to more frequent rejections and redrafting, introducing additional generation overhead and increasing latency.

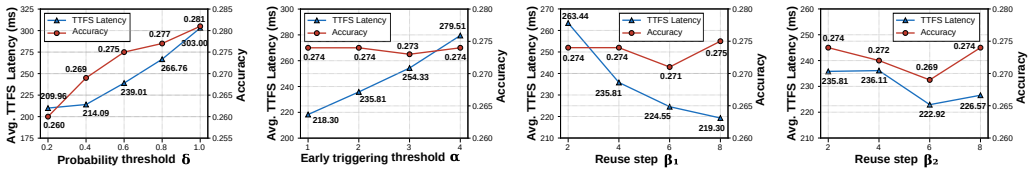


Figure 5: Sensitivity analysis of key hyperparameters in SpecSpeak. We report the impact of δ, α, β_1, and β_2 on the average TTFS latency and accuracy.

Impact of α. The early triggering threshold α primarily affects execution timing rather than output quality. Increasing α delays the start of diffusion, causing the system to wait longer before initiating pre-diffusion, which increases average latency. Meanwhile, accuracy remains nearly unchanged, indicating that α controls when synthesis starts but does not significantly impact token correctness.

Impact of β_1 and β_2. For diffusion reuse, both β_1 and β_2 exhibit similar trends. Larger β_1 allows reuse from later intermediate states in near-agreement cases, reducing redundant diffusion steps and lowering latency. Similarly, increasing β_2 enables more aggressive reuse in low-agreement regimes, further reducing diffusion cost. In both cases, accuracy remains stable, as these parameters operate purely on the synthesis side without affecting text-level verification. Appendix results show that increasing β_1 and β_2 introduces speech quality degradation under more aggressive reuse.

4.5 Impact of Draft Quality

We further analyze how the quality of draft tokens affects the efficiency of SpecSpeak. In SpecSpeak, the number of accepted tokens α_{acc} serves as an indicator of draft quality. Specifically, when $\alpha_{\text{acc}} = \alpha$ (high agreement), all draft tokens are accepted, corresponding to the best-case scenario. When $\alpha_{\text{acc}} = \alpha - 1$ or $\alpha - 2$ and $\alpha_{\text{acc}} > 0$ (near agreement), only a small number of tokens require correction. When $\alpha_{\text{acc}} < \alpha - 2$ or $\alpha_{\text{acc}} = 0$ (low agreement), the draft significantly deviates from the target model, representing the worst case.

We evaluate SpecSpeak under these regimes on TriviaQA using VocalNet as the base model, measuring TTFS latency and speech quality. As shown in Table 4, the acceleration gain strongly correlates with draft quality. In the high-agreement case, SpecSpeak achieves the largest improvement, reducing average TTFS by 46.5% compared to the baseline. Even in the low-agreement regime, SpecSpeak still provides a 6% latency reduction, demonstrating robustness under unfavorable conditions. Across all regimes, both WER and UTMOS remain largely unchanged, indicating that speech quality is preserved. The results demonstrate that SpecSpeak can effectively leverage high-quality drafts for maximum acceleration while remaining robust to imperfect drafting.

Table 4: Impact of draft token quality on SpecSpeak. Results are grouped by agreement level (high, near, low) based on α_{acc}. SpecSpeak achieves the largest TTFS latency reduction under the high agreement case (best case) and remains effective even in the low agreement case (worst case), while preserving speech quality across all regimes.

Case	TTFS Latency (ms)			Speech Quality	
	Avg ↓	P50 ↓	P90 ↓	WER ↓	UTMOS ↑
Base Model	311.82	309.31	335.57	**8.89**	**4.22**
High Agreement ($\alpha_{acc} = \alpha = 3$)	**166.87** (↓ 46%)	**166.47** (↓ 46%)	**181.28** (↓ 46%)	9.45 (↑ 6.3%)	4.19 (↓ 0.7%)
Near Agreement ($\alpha_{acc} = \alpha - 1$)	198.42 (↓ 36%)	187.69 (↓ 39%)	221.51 (↓ 34%)	10.12 (↑ 13%)	4.08 (↓ 3.3%)
Near Agreement ($\alpha_{acc} = \alpha - 2$)	246.55 (↓ 21%)	249.76 (↓ 19%)	289.71 (↓ 14%)	10.63 (↑ 19%)	4.05 (↓ 4.0%)
Low Agreement ($\alpha_{acc} < \alpha - 2$)	292.79 (↓ 6.1%)	298.26 (↓ 3.6%)	333.94 (↓ 0.5%)	10.75 (↑ 21%)	4.01 (↓ 5.0%)

4.6 Batched Serving Results

To further evaluate the effectiveness of SpecSpeak in serving scenarios, we conduct additional batched-serving experiments using VITA-Audio on the LLaMA Questions dataset, with batch sizes ranging from 1 to 128. We compare SpecSpeak against the original VITA-Audio baseline and conventional Speculative Decoding (SD), reporting the average, P50, and P90 TTFS latency under different batch sizes.

As shown in Table 5, SpecSpeak consistently achieves the lowest TTFS latency across all evaluated batch sizes and metrics. Specifically, SpecSpeak reduces the average TTFS latency by 23.0%–36.4% compared to the VITA-Audio baseline and 8.1%–22.7% compared to SD. Notably, even at a batch size of 128, SpecSpeak achieves an average TTFS of 5923.94 ms, corresponding to reductions of 25.0% and 10.3% over the baseline and SD, respectively. Moreover, SpecSpeak consistently achieves lower P90 latency, indicating improved tail-latency performance under batched execution. These results demonstrate that the latency benefits of SpecSpeak persist as batch size increases, highlighting its potential for efficient large-scale speech generation serving.

Table 5: Comparison of TTFS(ms) across different batch sizes using VITA-Audio on LLaMA Questions dataset. SpecSpeak maintains the advantage compared to VITA-Audio baseline and Speculative Decoding (SD).

Batch Size	VITA-Audio			VITA-Audio+SD			VITA-Audio+SpecSpeak		
	Avg.	P50	P90	Avg.	P50	P90	Avg.	P50	P90
1	328.48	320.61	344.08	270.83	235.53	343.34	**220.54**	**211.67**	**270.41**
8	601.78	587.65	630.14	504.28	477.19	740.31	**463.28**	**403.86**	**587.97**
16	1211.81	1199.47	1228.80	937.13	922.71	1856.29	**770.79**	**717.10**	**1045.15**
32	2290.46	2097.30	2848.45	2063.34	2029.95	3120.35	**1595.81**	**1446.62**	**2798.39**
64	4248.13	4012.62	4627.98	3698.32	3544.68	5049.62	**3106.43**	**2869.75**	**3872.89**
128	7896.20	7642.89	8113.75	6602.60	6574.46	8054.97	**5923.94**	**5529.74**	**7166.23**

5 Limitations and Future Work

Despite the effectiveness of SpecSpeak, we identify several potential limitations: **(1) Draft Quality.** The effectiveness of SpecSpeak depends on the quality of draft tokens. When the draft model produces low-quality prefixes, the system may fall back to more conservative execution, reducing the achievable acceleration. **(2) Computational Overhead.** SpecSpeak introduces additional coordination between drafting, verification, and diffusion reuse. Although our empirical results (Appendix G.3) show that this design incurs only modest computational overhead, it increases system complexity and may require careful implementation for efficient deployment. **(3) End-to-end Latency.** SpecSpeak is primarily designed to optimize Time-to-First-Speech (TTFS), which captures perceived responsiveness. While our experiments (Appendix G.4) show that the approach can also yield improvements in end-to-end latency, it does not explicitly optimize for overall generation time or throughput. Extending speculative co-generation to jointly optimize TTFS and end-to-end latency is an important direction for future work. **(4) Diffusion Synthesizer Applicability.** SpecSpeak is designed for SpeechLLMs with an LLM backbone and a diffusion-based speech synthesizer. For SpeechLLMs employing non-diffusion speech synthesizers, the current SpecSpeak pipeline is not directly applicable, since the agreement mechanism explicitly leverages the iterative refinement property of diffusion models.

6 Conclusion

We study the problem of reducing Time-to-First-Speech (TTFS) in SpeechLLM systems and identify a fundamental limitation of existing approaches: optimizing LLM decoding alone is insufficient due to the sequential dependency with diffusion-based speech synthesis. To address this, we propose SpecSpeak, a speculative co-generation framework that overlaps LLM decoding with speech synthesis. By combining early triggering, parallel verification, and verification-aware reuse of diffusion states, SpecSpeak transforms the conventional sequential pipeline into an adaptive co-generation process that shortens the critical path of TTFS. Extensive experiments demonstrate that SpecSpeak consistently reduces TTFS across multiple SpeechLLM architectures while preserving response correctness and speech quality.

7 Acknowledgments

This research was supported by the National Natural Science Foundation of China under Grant No. 62402267.

References

[1] Yadong Li, Jun Liu, Tao Zhang, Song Chen, Tianpeng Li, Zehuan Li, Lijun Liu, Lingfeng Ming, Guosheng Dong, Da Pan, et al. Baichuan-omni-1.5 technical report. *arXiv preprint arXiv:2501.15368*, 2025.

[2] Boyong Wu, Chao Yan, Chen Hu, Cheng Yi, Chengli Feng, Fei Tian, Feiyu Shen, Gang Yu, Haoyang Zhang, Jingbei Li, et al. Step-audio 2 technical report. *arXiv preprint arXiv:2507.16632*, 2025.

[3] Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, et al. Qwen2-audio technical report. *arXiv preprint arXiv:2407.10759*, 2024.

[4] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*, 2024.

[5] Wenbin Ding, Jun Chen, Mingjia Chen, Fei Xie, Qi Mao, and Philip Dames. Pfea: An llm-based high-level natural language planning and feedback embodied agent for human-centered ai. *arXiv preprint arXiv:2510.24109*, 2025.

[6] Heting Gao, Hang Shao, Xiong Wang, Chaofan Qiu, Yunhang Shen, Siqi Cai, Yuchen Shi, Zihan Xu, Zuwei Long, Yike Zhang, et al. Lucy: Linguistic understanding and control yielding early stage of her. *arXiv preprint arXiv:2501.16327*, 2025.

[7] Keyu An, Qian Chen, Chong Deng, Zhihao Du, Changfeng Gao, Zhifu Gao, Yue Gu, Ting He, Hangrui Hu, Kai Hu, et al. Funaudiollm: Voice understanding and generation foundation models for natural interaction between humans and llms. *arXiv preprint arXiv:2407.04051*, 2024.

[8] Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. Audiogpt: Understanding and generating speech, music, sound, and talking head. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 23802–23804, 2024.

[9] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

[10] Peng Wang, Songshuo Lu, Yaohua Tang, Sijie Yan, Wei Xia, and Yuanjun Xiong. A full-duplex speech dialogue scheme based on large language model. *Advances in Neural Information Processing Systems*, 37:13372–13403, 2024.

[11] Pan Zhang, Xiaoyi Dong, Yuhang Cao, Yuhang Zang, Rui Qian, Xilin Wei, Lin Chen, Yifei Li, Junbo Niu, Shuangrui Ding, et al. Internlm-xcomposer2. 5-omnilive: A comprehensive multimodal system for long-term streaming video and audio interactions. *arXiv preprint arXiv:2412.09596*, 2024.

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

[13] Aohan Zeng, Zhengxiao Du, Mingdao Liu, Kedong Wang, Shengmin Jiang, Lei Zhao, Yuxiao Dong, and Jie Tang. Glm-4-voice: Towards intelligent and human-like end-to-end spoken chatbot. *arXiv preprint arXiv:2412.02612*, 2024.

[14] Yuhao Wang, Heyang Liu, Ziyang Cheng, Ronghua Wu, Qunshan Gu, Yanfeng Wang, and Yu Wang. Vocalnet: Speech llms with multi-token prediction for faster and high-quality generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 19595–19612, 2025.

[15] Yushen Chen, Zhikang Niu, Ziyang Ma, Keqi Deng, Chunhui Wang, JianZhao JianZhao, Kai Yu, and Xie Chen. F5-tts: A fairytaler that fakes fluent and faithful speech with flow matching. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6255–6271, 2025.

[16] 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.

[17] Zhihao Du, Changfeng Gao, Yuxuan Wang, Fan Yu, Tianyu Zhao, Hao Wang, Xiang Lv, Hui Wang, Chongjia Ni, Xian Shi, et al. Cosyvoice 3: Towards in-the-wild speech generation via scaling-up and post-training. *arXiv preprint arXiv:2505.17589*, 2025.

[18] Shivam Mehta, Ruibo Tu, Jonas Beskow, Éva Székely, and Gustav Eje Henter. Matcha-tts: A fast tts architecture with conditional flow matching. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11341–11345. IEEE, 2024.

[19] Zuwei Long, Yunhang Shen, Chaoyou Fu, Heting Gao, Lijiang Li, Peixian Chen, Mengdan Zhang, Hang Shao, Jian Li, Jinlong Peng, et al. Vita-audio: Fast interleaved cross-modal token generation for efficient large speech-language model. *arXiv preprint arXiv:2505.03739*, 2025.

[20] Siddhant Arora, Kai-Wei Chang, Chung-Ming Chien, Yifan Peng, Haibin Wu, Yossi Adi, Emmanuel Dupoux, Hung-Yi Lee, Karen Livescu, and Shinji Watanabe. On the landscape of spoken language models: A comprehensive survey. *arXiv preprint arXiv:2504.08528*, 2025.

[21] Jing Peng, Yucheng Wang, Bohan Li, Yiwei Guo, Hankun Wang, Yangui Fang, Yu Xi, Haoyu Li, Xu Li, Ke Zhang, et al. A survey on speech large language models for understanding. *IEEE Journal of Selected Topics in Signal Processing*, 2025.

[22] Wenqian Cui, Dianzhi Yu, Xiaoqi Jiao, Ziqiao Meng, Guangyan Zhang, Qichao Wang, Steven Y Guo, and Irwin King. Recent advances in speech language models: A survey. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13943–13970, 2025.

[23] Qingkai Fang, Yan Zhou, Shoutao Guo, Shaolei Zhang, and Yang Feng. Llama-omni 2: Llm-based real-time spoken chatbot with autoregressive streaming speech synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18617–18629, 2025.

[24] 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.

[25] Ziyang Cheng, Yuhao Wang, Heyang Liu, Ronghua Wu, Qunshan Gu, Yanfeng Wang, and Yu Wang. Vocalnet-mdm: Accelerating streaming speech llm via self-distilled masked diffusion modeling. *arXiv preprint arXiv:2602.08607*, 2026.

[26] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[27] Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925, 2023.

[28] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, pages 28935–28948, 2024.

[29] Tongyi Fun Team, Qian Chen, Luyao Cheng, Chong Deng, Xiangang Li, Jiaqing Liu, Chao-Hong Tan, Wen Wang, Junhao Xu, Jieping Ye, et al. Fun-audio-chat technical report. *arXiv preprint arXiv:2512.20156*, 2025.

[30] Yuhao Wang, Ziyang Cheng, Heyang Liu, Ronghua Wu, Qunshan Gu, Yanfeng Wang, and Yu Wang. Vocalnet-m2: Advancing low-latency spoken language modeling via integrated multi-codebook tokenization and multi-token prediction. *arXiv preprint arXiv:2511.10232*, 2025.

[31] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, 2017.

[32] Eliya Nachmani, Alon Levkovitch, Roy Hirsch, Julian Salazar, Chulayuth Asawaroengchai, Soroosh Mariooryad, Ehud Rivlin, RJ Skerry-Ryan, and Michelle Tadmor Ramanovich. Spoken question answering and speech continuation using spectrogram-powered llm. *arXiv preprint arXiv:2305.15255*, 2023.

[33] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544, 2013.

[34] Li Xuechen, Zhang Tianyi, Dubois Yann, Taori Rohan, Gulrajani Ishaan, Guestrin Carlos, et al. Alpacaeval: An automatic evaluator of instruction-following models, 2023.

[35] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[36] Xian Shi, Xiong Wang, Zhifang Guo, Yongqi Wang, Pei Zhang, Xinyu Zhang, Zishan Guo, Hongkun Hao, Yu Xi, Baosong Yang, et al. Qwen3-asr technical report. *arXiv preprint arXiv:2601.21337*, 2026.

[37] 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.

[38] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[39] Binbin Zhang, Hang Lv, Pengcheng Guo, Qijie Shao, Chao Yang, Lei Xie, Xin Xu, Hui Bu, Xiaoyu Chen, Chenchen Zeng, et al. Wenetspeech: A 10000+ hours multi-domain mandarin corpus for speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6182–6186. IEEE, 2022.

[40] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[41] Daniel Galvez, Greg Diamos, Juan Ciro, Juan Felipe Cerón, Keith Achorn, Anjali Gopi, David Kanter, Maximilian Lam, Mark Mazumder, and Vijay Janapa Reddi. The people's speech: A large-scale diverse english speech recognition dataset for commercial usage. *arXiv preprint arXiv:2111.09344*, 2021.

[42] Vineel Pratap, Qiantong Xu, Anuroop Sriram, Gabriel Synnaeve, and Ronan Collobert. Mls: A large-scale multilingual dataset for speech research. *arXiv preprint arXiv:2012.03411*, 2020.

[43] Guoguo Chen, Shuzhou Chai, Guanbo Wang, Jiayu Du, Wei-Qiang Zhang, Chao Weng, Dan Su, Daniel Povey, Jan Trmal, Junbo Zhang, et al. Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio. *arXiv preprint arXiv:2106.06909*, 2021.

[44] Changhan Wang, Morgane Riviere, Ann Lee, Anne Wu, Chaitanya Talnikar, Daniel Haziza, Mary Williamson, Juan Pino, and Emmanuel Dupoux. Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 993–1003, 2021.

[45] Linhan Ma, Dake Guo, Kun Song, Yuepeng Jiang, Shuai Wang, Liumeng Xue, Weiming Xu, Huan Zhao, Binbin Zhang, and Lei Xie. Wenetspeech4tts: A 12,800-hour mandarin tts corpus for large speech generation model benchmark. *arXiv preprint arXiv:2406.05763*, 2024.

[46] 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. *arXiv preprint arXiv:1904.02882*, 2019.

[47] Wenbin Wang, Yang Song, and Sanjay Jha. Globe: A high-quality english corpus with global accents for zero-shot speaker adaptive text-to-speech. *arXiv preprint arXiv:2406.14875*, 2024.

[48] Haorui He, Zengqiang Shang, Chaoren Wang, Xuyuan Li, Yicheng Gu, Hua Hua, Liwei Liu, Chen Yang, Jiaqi Li, Peiyang Shi, et al. Emilia: An extensive, multilingual, and diverse speech dataset for large-scale speech generation. In *2024 IEEE Spoken Language Technology Workshop (SLT)*, pages 885–890. IEEE, 2024.

[49] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

[50] Yunhang Shen, Chaoyou Fu, Shaoqi Dong, Xiong Wang, Yi-Fan Zhang, Peixian Chen, Mengdan Zhang, Haoyu Cao, Ke Li, Shaohui Lin, et al. Long-vita: Scaling large multi-modal models to 1 million tokens with leading short-context accuracy. *arXiv preprint arXiv:2502.05177*, 2025.

[51] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021, 2023.

[52] Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*, 2024.

[53] Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

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

[55] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

A Broader Impacts

Potential Positive Societal Impacts. This work aims to improve the responsiveness of SpeechLLM systems by reducing Time-to-First-Speech (TTFS), a key latency metric for real-time human–AI interaction. Lower latency can significantly enhance user experience in latency-sensitive applications such as voice assistants, accessibility tools, and interactive conversational agents, where timely feedback is essential.

Potential Negative Societal Impacts. Despite these benefits, improved responsiveness may also introduce risks. Reducing latency makes synthesized speech appear more immediate and natural, which could increase the risk of misuse in applications such as impersonation.

Future Works. This work points to several important directions for future research. First, improving robustness and controllability in low-latency speech generation remains an open challenge, particularly under partial or uncertain intermediate outputs. Second, incorporating safety mechanisms, such as content verification and misuse detection, is critical for deploying responsive SpeechLLM systems in real-world settings. Finally, exploring principled trade-offs between latency, quality, and reliability will be essential for advancing practical and trustworthy speech-based human–AI interaction.

B Dataset Details

We evaluate our framework on the English subsets of OpenAudioBench [1], following prior SpeechLLM studies [25, 1, 14, 30, 23]. This benchmark suite covers four representative speech-based question answering tasks—TriviaQA [31], WebQuestions [33], LLaMA Questions [32], and AlpacaEval [34]. These datasets provide a comprehensive evaluation of SpeechLLMs under diverse interaction patterns and response requirements. We provide detailed descriptions of each dataset as follows.

- **TriviaQA** is a knowledge-intensive question answering benchmark consisting of factoid questions collected from trivia and quiz-league websites. The task requires the model to answer open-domain factual questions based on general world knowledge. It evaluates the factual correctness and robustness of SpeechLLMs under relatively complex reasoning and long-form response settings.
- **WebQuestions** is a question answering benchmark built from real user queries issued to search engines and grounded on Freebase knowledge. The task focuses on short factual questions involving entities, relations, and common knowledge. It is used to assess the model's capability in handling concise spoken queries with high response precision.
- **LLaMA Questions** is a spoken question answering benchmark introduced for evaluating end-to-end speech interaction with large language models. It contains conversational and knowledge-seeking spoken queries with diverse linguistic styles and varying complexity. This dataset emphasizes robustness under natural spoken inputs and better reflects interactive dialogue scenarios.
- **AlpacaEval** is an instruction-following benchmark adapted for spoken interaction evaluation. It contains diverse open-ended user instructions covering reasoning, explanation, generation, and general assistance tasks. This dataset evaluates the generalization ability of SpeechLLMs beyond factual QA and measures their capability in handling complex real-world spoken instructions.

C Base Model Details

This paper selects two representative open-source SpeechLLMs as the base models: **VocalNet** [14] and **VITA-Audio** [19]. These two models represent two mainstream paradigms of end-to-end speech generation: text then audio cascaded speech generation and unified interleaved text-audio generation.

C.1 VocalNet

VocalNet is an aligned multimodal SpeechLLM that follows a cascaded speech generation framework. It consists of four major components: a speech encoder, a pretrained LLM backbone, a speech

decoder, and a speech vocoder. The input speech is first processed by a frozen speech encoder (Whisper-large-v3) and a downsample adaptor to produce compressed acoustic representations, which are then fed into the LLM backbone for semantic understanding and text generation. In each loop, the LLM generates 5 text tokens, and the corresponding hidden states are further passed to an autoregressive speech decoder to generate 15 discrete speech tokens, which are finally converted into waveform outputs using a chunk-aware flow matching vocoder together with HiFi-GAN.

A key feature of VocalNet is the use of *Multi-Token Prediction (MTP)* for speech token decoding. Instead of standard next-token prediction, VocalNet introduces sequential MTP modules that enable multiple speech tokens to be predicted within a single inference step, significantly improving decoding efficiency while preserving temporal dependencies among speech tokens. This design mainly accelerates the speech generation stage after text generation has already started, but the first speech chunk still depends on the completion of initial text decoding, leaving TTFS latency suboptimal.

In our experiments, we use `VocalNet-8B` as the target model and `VocalNet-1B` as the draft model for speculative decoding, following the officially released checkpoints. We use 2 MTP modules in our experiments.

C.2 VITA-Audio

VITA-Audio adopts a unified interleaved generation paradigm for speech interaction. Unlike cascaded systems, it alternates between text token generation and audio token generation within a single autoregressive sequence, enabling speech output during the early stages of response generation.

The model consists of an audio encoder, an audio decoder, an LLM backbone, and a set of lightweight *Multiple Cross-modal Token Prediction (MCTP)* modules. During each forward pass, the LLM predicts one text token, while the MCTP modules directly generate multiple corresponding audio tokens from the LLM hidden states and token embeddings.

VITA-Audio provides several inference modes (*Turbo*, *Boost*, *Balance*, and *Vanilla*) with different speed-quality trade-offs. Among them, the *Balance* mode maintains a stable text-audio ratio and provides better speech quality for spoken dialogue tasks. For the first audio chunk, it generates 1 text token, followed by 4 speech tokens, then 3 text tokens and 8 speech tokens, the collected 8 speech tokens (the 12 speech tokens include 4 special tokens of audio start and stop). After that, it generates 4 text tokens and 10 speech tokens in each loop. In our experiments, we use the released `VITA-Audio Balance 7B` model as the target model and train an additional `0.5B draft model` (the training details are provided in Appendix F) to support speculative decoding.

Compared with VocalNet, VITA-Audio already reduces first-audio latency through interleaved decoding and MCTP. We adopt them as base models, and apply Speculative Decoding techniques and `SpecSpeak` on them for comparison.

D Speculative Decoding Details

`SpecSpeak` introduces a draft model M_q and serves the high-quality SpeechLLM as the target model M_p. The drafting and verification follow a standard speculative decoding procedure [26]. The details are illustrated as follows.

For an input prefix $\mathbf{y}_{<i} = (y_1, \ldots, y_{i-1})$, the target model defines the next-token distribution

$$p_i(v) = P_{M_p}(y_i = v \mid \mathbf{y}_{<i}),$$

while the draft model defines

$$q_i(v) = P_{M_q}(y_i = v \mid \mathbf{y}_{<i}).$$

Here, $p_i(\cdot)$ is the desired distribution and $q_i(\cdot)$ is its approximation.

Drafting. Given the current prefix $\mathbf{y}_{<t}$, the draft model autoregressively proposes α candidate text tokens

$$\hat{y}_t, \hat{y}_{t+1}, \ldots, \hat{y}_{t+\alpha-1},$$

where each token is sampled from

$$\hat{y}_{t+k} \sim q_{t+k}(\cdot), \quad k = 0, 1, \ldots, \alpha - 1.$$

Then the draft sequence is

$$\hat{\mathbf{y}} = (\hat{y}_t, \hat{y}_{t+1}, \ldots, \hat{y}_{t+\alpha-1}).$$

Verification. Instead of verifying drafted tokens one by one, the target model performs only a single forward pass over the drafted prefix

$$\mathbf{y}_{<t} \oplus (\hat{y}_t, \hat{y}_{t+1}, \ldots, \hat{y}_{t+\alpha-1}),$$

and simultaneously obtains the conditional distributions for all drafted positions:

$$p_t(\cdot),\ p_{t+1}(\cdot),\ \ldots,\ p_{t+\alpha}(\cdot).$$

For each drafted token $\hat{y}_{t+k}$, we compute its acceptance probability

$$A_{t+k} = \min\left(1,\ \frac{p_{t+k}(\hat{y}_{t+k})}{q_{t+k}(\hat{y}_{t+k})}\right).$$

Then, a random variable threshold $\delta_{t+k} \sim \mathcal{U}(0,1)$ is sampled, and the draft token is accepted if $\delta_{t+k} \leq A_{t+k}$. Suppose the first rejected position is $t+n$, then the first n drafted tokens are accepted:

$$\hat{y}_t, \hat{y}_{t+1}, \ldots, \hat{y}_{t+n-1}.$$

Correction. At the first rejected token position, the target model can sample a bonus token from the residual distribution

$$p'_{t+n}(v) = \mathrm{norm}\left(\max\left(0,\ p_{t+n}(v) - q_{t+n}(v)\right)\right),$$

where $\mathrm{norm}(\cdot)$ denotes normalization into a valid probability distribution. The bonus token is sampled as

$$y_{t+n} \sim p'_{t+n}(\cdot).$$

If all drafted tokens are accepted, the target model directly samples the bonus token from

$$y_{t+\alpha} \sim p_{t+\alpha}(\cdot).$$

Therefore, each speculative decoding process yields at least one token (0 accepted tokens and 1 bonus token) and at most $\alpha + 1$ tokens (α accepted tokens and 1 bonus token).

E Algorithm Details

Algorithm 1: `Speculative Decoding Verification (SD)`

Input: Accepted Prefix $\mathbf{a}'$; Draft $\mathbf{a}$; Draft logits $\mathbf{q}$; Target model M_p; Draft length d.
Output: Verified draft $\mathbf{a}$; Total length of accepted tokens α_{acc}.
Initialize $\alpha_{\mathrm{acc}} \leftarrow$ length of $\mathbf{a}'$
// Target Verification
Target model runs one forward pass over the drafted prefix: $\mathbf{p}(\cdot) \leftarrow M_p(\mathbf{a})$
for $k = 1$ **to** d **do**
  Compute acceptance probability: $A_k \leftarrow \min\left(1, \frac{p_k(a_k)}{q_k(a_k)}\right)$
  Sample $\delta_k \sim \mathcal{U}(0,1)$
  if $\delta_k \leq A_k$ **then**
    Accept a_k and set $\alpha_{\mathrm{acc}} \leftarrow \alpha_{\mathrm{acc}} + 1$
  else
    Reject a_k and stop verification
    Construct the residual target distribution: $p'_k(x) \leftarrow \frac{\max(0, p_k(x) - q_k(x))}{\sum_{x'} \max(0, p_k(x') - q_k(x'))}$
    Sample one correction token: $a_{\mathrm{corr}} \sim p'_k(\cdot)$
    Construct updated prefix: $\mathbf{a}_{1:(\alpha_{\mathrm{acc}}+1)} \leftarrow \mathrm{Concat}(\mathbf{a}', \mathbf{a}_{1:k}, a_{\mathrm{corr}})$
    return $\mathbf{a}_{1:(\alpha_{\mathrm{acc}}+1)}$,$\alpha_{acc}$
// All Draft Tokens Accepted
Sample one bonus token from the target model: $a_{\mathrm{bonus}} \sim p_d(\cdot)$
Construct updated prefix: $\mathbf{a}_{1:(\alpha_{\mathrm{acc}}+1)} \leftarrow \mathrm{Concat}(\mathbf{a}', \mathbf{a}_{1:d}, a_{\mathrm{bonus}})$
return $\mathbf{a}_{1:(\alpha_{\mathrm{acc}}+1)}$,$\alpha_{acc}$

Algorithm 2: `SpecSpeak`

Input: Input speech $\mathbf{x}^s$; Encoder $\mathcal{E}$; Draft model M_q; Target model M_p; Diffusion model $\mathcal{V}$; Vocoder; Threshold α; Reuse steps β_1, β_2
Output: First speech chunk $\mathbf{y}$
Initialize verified text tokens $\mathbf{a}^t \leftarrow \emptyset$, draft text tokens $\tilde{\mathbf{a}}^t \leftarrow \emptyset$
Initialize diffusion step $i \leftarrow 1$, verified text tokens length $m \leftarrow 0$, $ReuseFlag \leftarrow False$
// Speech Encoding
$\mathbf{z} = \mathcal{E}(\mathbf{x}^s)$
while $m < \alpha$ **do**
  // Text Tokens Drafting
  Draft model generates text tokens autoregressively until $len(\tilde{\mathbf{a}}^t) = \alpha$: $\tilde{\mathbf{a}}^t_{(m+1):\alpha}, \mathbf{q}_{(m+1):\alpha} \sim M_q(\mathbf{a}^t_{:m}, \mathbf{z})$
  $\tilde{\mathbf{a}}^t_{1:\alpha} \leftarrow \mathrm{Concat}(\mathbf{a}^t_{1:m}, \tilde{\mathbf{a}}^t_{(m+1):\alpha})$
  Parallel do
    // Branch1: Early Triggering Speech Synthesis
    Generate speech tokens $\tilde{\mathbf{a}}^s$ following $\tilde{\mathbf{a}}^t_{1:\alpha}$
    if *$ReuseFlag$ is False* **then**
      Start diffusion synthesis $\mathcal{V}_i$ from $\Gamma_1 \sim \mathcal{N}(0, I)$ and generate intermediate states $\mathbf{\Gamma}_{2:n_d}$ conditioned on $\tilde{\mathbf{a}}^s$
    else
      Start diffusion synthesis $\mathcal{V}_i$ from Γ_{β_2} in $\mathcal{V}_{i-1}$ and generate $\mathbf{\Gamma}_{(\beta_2+1):n_d}$ conditioned on $\tilde{\mathbf{a}}^s$
    // Branch2: Parallel Verification (SD)
    $\tilde{\mathbf{a}}^t_{1:(\alpha_{acc}+1)}, \alpha_{\text{acc}} \leftarrow \mathtt{SD}(\mathbf{a}^t_{:m}, \tilde{\mathbf{a}}^t_{(m+1):\alpha}, \mathbf{q}_{(m+1):\alpha}, M_p, \alpha - m)$ // Algorithm 1
    $m \leftarrow \alpha_{acc} + 1$, $\mathbf{a}^t_{1:m} \leftarrow \tilde{\mathbf{a}}^t_{1:(\alpha_{acc}+1)}$
  // High Agreement
  if $\alpha_{\text{acc}} = \alpha$ **then**
    Accept Γ_{n_d} as the final mel spectrum
    Break
  else
    // Near Agreement
    if $\alpha_{\text{acc}} \in \{\alpha - 1, \alpha - 2\}$ *and* $\alpha_{\text{acc}} > 0$ **then**
      if $\alpha_{acc} = \alpha - 1$ **then**
        Generate speech tokens $\mathbf{a}^s$ following $\mathbf{a}^t_{1:\alpha}$
        Start diffusion synthesis $\mathcal{V}_{i+1}$ from Γ_{β_1} in $\mathcal{V}_i$ and generate $\mathbf{\Gamma}_{(\beta_1+1):n_d}$ conditioned on $\mathbf{a}^s$
        Break
      else
        Generate one text token with the target model: $a^t_\alpha \sim M_p(\mathbf{a}^t_{1:(\alpha-1)}, \mathbf{z})$
        $\mathbf{a}^t_{1:\alpha} \leftarrow \mathrm{Concat}(\mathbf{a}^t_{1:(\alpha-1)}, a^t_\alpha)$
        Generate speech tokens $\mathbf{a}^s$ following $\mathbf{a}^t_{1:\alpha}$
        Start diffusion synthesis $\mathcal{V}_{i+1}$ from Γ_{β_1} in $\mathcal{V}_i$ and generate $\mathbf{\Gamma}_{(\beta_1+1):n_d}$ conditioned on $\mathbf{a}^s$
        Break
    else
      // Low Agreement
      Parallel do
        // Branch1: Asynchronously Launch Speech Synthesis
        Generate speech tokens $\mathbf{a}^s_{trial}$ following $\tilde{\mathbf{a}}^t_{1:m}$
        Start diffusion synthesis $\mathcal{V}_{i+1}$ from $\Gamma_1 \sim \mathcal{N}(0, I)$ conditioned on $\mathbf{a}^s_{trial}$
        // Branch2: Continue Drafting
        $i \leftarrow i + 1$
        $ReuseFlag \leftarrow True$
        Continue
$\mathbf{y} = \mathrm{Vocoder}(\Gamma_{n_d})$
return $\mathbf{y}$

F Implementation Details

Base Model Configurations. We set the number of MTP modules to 2 in VocalNet. Other parameters of VocalNet are illustrated in Table 6. We adopt the balance mode of VITA-Audio[4] to maintain audio quality while preserving speedup performance and disabling probabilistic sampling following its default configurations. For both base models, we enable FlashAttention [38] and convert Pytorch checkpoint of diffusion model to TensorRT engine, which is consistent with real scenario inference deployment.

[4] `https://huggingface.co/VITA-MLLM/VITA-Audio-Balance`

Table 6: The generation configurations of VocalNet.

Module	Configuration
Backbone LLM	Temperature = 0.1 Top-p = 0.1
Speech Decoder	Top-k = 1 Penalty window size = 6 Penalty = 2

Training Details of the 0.5B Draft Model. To train a lightweight draft model for VITA-Audio-7B, we start with Qwen2.5-0.5B-Instruct[5] and perform a four-step training process using the same method and datasets introduced in Vita-Audio [19]. In Stage 1, audio–text alignment is achieved via large-scale ASR [39–44], TTS [45–48], and text-only [49–53] continual pretraining with frozen audio encoder and decoder. In Stage 2, a single multiple cross-modal token prediction (MCTP) module is trained for next-token prediction based on LLM outputs and hidden states. In Stage 3, this module is extended to multiple MCTP modules with iterative prediction and shared initialization. In Stage 4, supervised fine-tuning on speech QA [54, 6] data with interleaved outputs enables efficient speech-to-speech dialogue.

The training is conducted with a global batch size of 64 over a single epoch on 4xA800 GPUs. Distinct learning rate schedules are adopted across stages: the LLM is optimized with a learning rate of 6×10^{-5} during Stage 1 for 8000 steps, while the MCTP modules are trained with 1×10^{-3} in Stages 2 and 3 for 4000 and 8000 steps separately. In the final stage, the learning rates are adjusted to 5×10^{-5} for the LLM and 4×10^{-5} for the MCTP modules with 4000 steps. Optimization is performed using AdamW [55] with $\beta_1 = 0.9$ and $\beta_2 = 0.95$.

G Additional Experimental Results and Analysis

G.1 Additional Ablation Studies

Table 7: Ablation study on VITA-Audio (LLaMA Questions).

Method	TTFS Latency (ms)			Response Correctness and Quality		
	Avg ↓	P50 ↓	P90 ↓	Acc. ↑	WER ↓	UTMOS ↑
SpecSpeak	220.54	**211.67**	**270.41**	0.43	3.48	**3.79**
w/o Early Triggering	270.83 (↑ 23%)	235.53 (↑ 11%)	343.34 (↑ 27%)	0.45 (↑ 4.7%)	3.71 (↑ 6.6%)	**3.79** (↑ 0.0%)
w/o Diffusion Reuse	260.49 (↑ 18%)	235.30 (↑ 11%)	272.41 (↑ 0.74%)	0.43 (↑ 0.0%)	**3.30** (↓ 5.2%)	3.78 (↓ 0.26%)
w/o Drafting	290.38 (↑ 32%)	291.19 (↑ 38%)	326.62 (↑ 21%)	**0.46** (↑ 7.0%)	3.37 (↓ 3.2%)	3.04 (↓ 20%)
Normal Early Triggering	239.37 (↑ 8.5%)	227.69 (↑ 7.6%)	290.20 (↑ 7.3%)	0.45 (↑ 4.7%)	3.82 (↑ 9.8%)	3.71 (↓ 2.1%)
Draft-based Correction	**213.45** (↓ 3.2%)	212.02 (↑ 0.17%)	272.92 (↑ 0.93%)	0.41 (↓ 4.7%)	3.54 (↑ 1.7%)	3.77 (↓ 0.53%)
No Parallel Redraft Synthesis	223.23 (↑ 1.2%)	217.58 (↑ 2.8%)	271.44 (↑ 0.38%)	0.43 (↑ 0.0%)	3.86 (↑ 11%)	3.76 (↓ 0.79%)

Table 8: Ablation study on VITA-Audio (WebQuestions).

Method	TTFS Latency (ms)			Response Correctness and Quality		
	Avg ↓	P50 ↓	P90 ↓	Acc. ↑	WER ↓	UTMOS ↑
SpecSpeak	242.81	**219.87**	**283.04**	0.24	4.19	3.67
w/o Early Triggering	290.29 (↑ 20%)	269.93 (↑ 23%)	350.88 (↑ 24%)	**0.25** (↑ 4.2%)	4.67 (↑ 11%)	**3.69** (↑ 0.54%)
w/o Diffusion Reuse	268.93 (↑ 11%)	229.67 (↑ 4.5%)	310.91 (↑ 9.8%)	0.24 (↑ 0.0%)	**3.77** (↓ 10%)	**3.69** (↑ 0.54%)
w/o Drafting	271.71 (↑ 12%)	280.62 (↑ 28%)	283.22 (↑ 0.06%)	**0.25** (↑ 4.2%)	4.92 (↑ 17%)	3.11 (↓ 15%)
Normal Early Triggering	256.20 (↑ 5.5%)	258.27 (↑ 17%)	339.94 (↑ 20%)	**0.25** (↑ 4.2%)	4.02 (↓ 4.1%)	3.68 (↑ 0.27%)
Draft-based Correction	**241.52** (↓ 0.53%)	221.75 (↑ 0.85%)	291.15 (↑ 2.9%)	0.21 (↓ 13%)	4.97 (↑ 19%)	3.61 (↓ 1.6%)
No Parallel Redraft Synthesis	244.31 (↑ 0.62%)	222.75 (↑ 1.3%)	284.07 (↑ 0.36%)	0.24 (↑ 0.0%)	4.16 (↓ 0.72%)	3.68 (↑ 0.27%)

[5] https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct

As shown in Table 7 and Table 8, we extend the ablation studies to WebQuestions and LLaMA Questions based on VITA-Audio. The results show that removing any core component more or less increases TTFS latency, while causing only marginal changes in response correctness and speech quality across both the LLaMA Questions and WebQuestions benchmarks. This demonstrates that the major benefit of SpecSpeak lies in substantially improving generation efficiency without sacrificing answer accuracy or speech naturalness.

Specifically, disabling Early Triggering results in the most significant latency degradation, confirming that early response initiation is critical for reducing first-token delay. Removing Diffusion Reuse also introduces clear latency overhead, indicating that reusing intermediate acoustic states effectively avoids redundant computation during speech synthesis. Eliminating Drafting causes the largest overall latency increase and also leads to a clear degradation in UTMOS, showing that the drafting mechanism not only improves efficiency but also helps maintain speech quality. Among the fine-grained variants, replacing speculative early triggering with normal early triggering, removing parallel redraft synthesis, or adopting draft-based correction all result in slower response generation, further validating the importance of the proposed parallel drafting and correction design.

Across both datasets, the accuracy and WER remain relatively stable under different variants, suggesting that the latency improvements mainly come from better system scheduling and generation strategies rather than sacrificing response correctness. These results collectively verify the effectiveness of the key components in SpecSpeak and confirm the soundness of the overall system design.

G.2 Additional Sensitivity Analysis

We include more sensitivity analysis results of δ, α, β_1, and β_2 on SpecSpeak in terms of tail latency (P50 latency and P90 latency) and generated speech quality (WER and UTMOS).

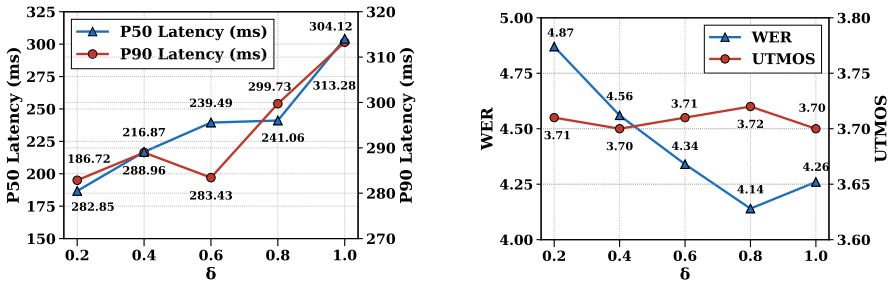


Figure 6: The impact of probability threshold δ on SpecSpeak's tail latency (P50 latency and P90 latency) and generated speech quality (WER and UTMOS).

Figure 6 shows that, as δ increases, both P50 and P90 latency increase consistently with average latency due to stricter acceptance criteria. WER decreases with larger δ, indicating closer alignment between the draft and target models. UTMOS remains largely unchanged, suggesting that speech quality is not sensitive to δ.

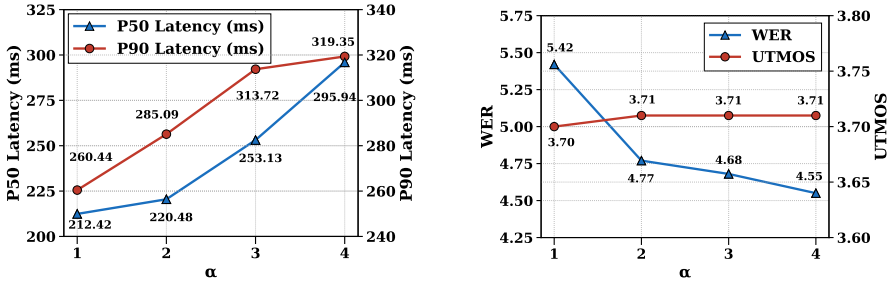


Figure 7: The impact of early triggering threshold α on SpecSpeak's tail latency (P50 latency and P90 latency) and generated speech quality (WER and UTMOS).

Figure 7 shows that, as α increases, both P50 and P90 latency increase due to delayed initiation of speech synthesis. WER decreases notably when α increases from 1 to 2, highlighting the benefit of a minimally sufficient prefix for stable generation. UTMOS remains stable, indicating preserved speech quality.

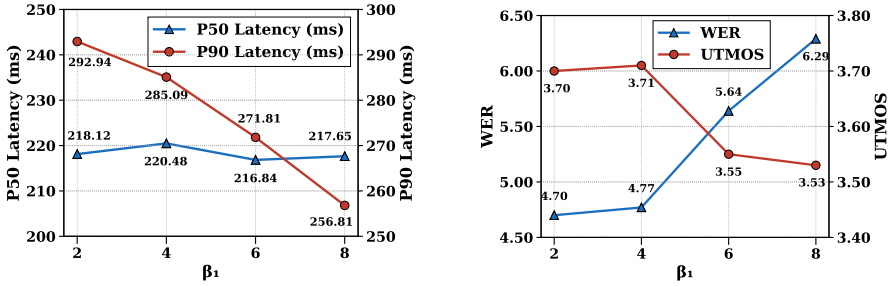


Figure 8: The impact of reuse step β_1 on SpecSpeak's tail latency (P50 latency and P90 latency) and generated speech quality (WER and UTMOS).

Figure 8 shows that, as β_1 increases, P50 latency remains stable while P90 latency decreases, indicating that reuse primarily mitigates tail latency caused by redrafting and re-diffusion. However, UTMOS degrades with larger β_1, reflecting the trade-off introduced by more aggressive reuse, where diffusion errors may not be fully corrected.

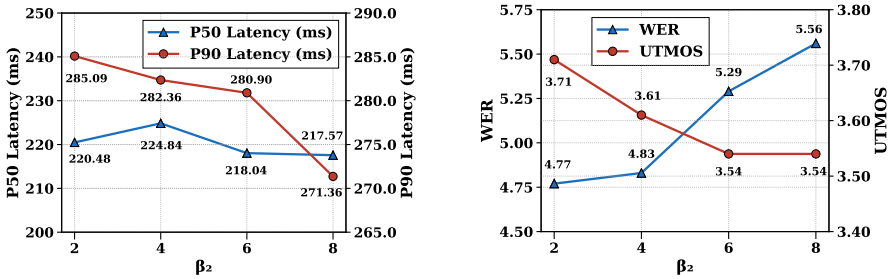


Figure 9: The impact of reuse step β_2 on SpecSpeak's tail latency (P50 latency and P90 latency) and generated speech quality (WER and UTMOS).

Figure 9 shows that, similar to β_1, increasing β_2 keeps P50 latency stable while reducing P90 latency by alleviating tail latency. However, more aggressive reuse leads to degradation in UTMOS, consistent with the trade-off between efficiency and correction fidelity.

G.3 Cost Analysis

We analyze the hardware overhead introduced by SpecSpeak beyond latency improvements. Specifically, we measure **GPU memory usage**, **power consumption**, and **GPU utilization**, which together reflect the deployment cost of the proposed co-generation framework. Experiments are conducted using VITA-Audio on the LLaMA Questions dataset under the same environment settings as described in Appendix F. For each query, we run the full inference pipeline and record GPU statistics at a 2-second interval, reporting average values across all samples.

Table 9: Resource cost analysis of SpecSpeak on VITA-Audio. We report average GPU memory usage, power consumption, and utilization across the LLaMA Questions dataset. The additional runtime overhead introduced by SpecSpeak is limited.

Method	GPU Memory Usage	Power Consumption	GPU Utilization
VITA-Audio	34.65 GiB	193.13 W	82.70%
VITA-Audio+SpecSpeak	41.30 GiB	196.47 W	83.50%

Table 9 summarizes the results. Compared to the original VITA-Audio, integrating SpecSpeak increases average GPU memory usage from 34.65 GiB to 41.30 GiB. This overhead primarily arises from maintaining both draft and target models, as well as storing intermediate diffusion states for reuse during speculative correction. In contrast, the additional computational overhead is minimal. Average power consumption increases only slightly from 193.13 W to 196.47 W, and GPU utilization remains stable (82.70% vs. 83.50%). This indicates that SpecSpeak does not substantially increase computational pressure during inference, despite introducing additional coordination across components.

Overall, these results suggest that SpecSpeak achieves significant TTFS reduction with moderate memory overhead, while incurring minimal additional energy and compute cost. This enables improving perceived responsiveness without introducing substantial runtime burden.

G.4 End-to-End Latency Analysis

While SpecSpeak is primarily designed to reduce Time-to-First-Speech (TTFS), we further evaluate its impact on end-to-end speech generation latency.

We conduct experiments using VITA-Audio on the AlpacaEval dataset. To measure end-to-end efficiency, we report *Tokens Per Second* (TPS) and *Real-Time Factor* (RTF), defined as the number of generated tokens divided by total generation time, and the ratio between total generation time and speech duration, respectively. We additionally evaluate response correctness and speech quality using the same metrics as in the main experiments, including Score, WER, and UTMOS.

Table 10: End-to-end latency and speech quality comparison on AlpacaEval. TPS (Tokens Per Second) is defined as the number of generated tokens divided by total generation time, and RTF (Real-Time Factor) is defined as the ratio between total generation time and speech duration.

Method	**Efficiency**		**Response Correctness and Quality**		
	RTF ↓	TPS ↑	Score ↑	WER ↓	UTMOS ↑
VITA-Audio	0.40	56.13	**2.55**	**9.05**	**3.56**
VITA-Audio+SD	**0.38**	56.83	2.28	9.95	3.34
VITA-Audio+SpecSpeak	**0.38**	**59.81**	2.23	9.67	3.24

Table 10 presents the results. Compared to the base model, SpecSpeak improves efficiency, achieving higher TPS (59.81 vs. 56.13) and lower RTF (0.38 vs. 0.40). However, the improvement is less pronounced than the gains observed in TTFS. In terms of response correctness and speech quality, SpecSpeak remains competitive with the baselines, with only minor differences across Score, WER, and UTMOS.

Discussion. The relatively modest improvement in end-to-end latency is expected. TTFS is dominated by the critical path before the first audio is produced, where LLM decoding and diffusion synthesis are executed sequentially. SpecSpeak effectively shortens this critical path by overlapping the two stages. However, as generation progresses, the base model itself increasingly exhibits natural overlap between LLM decoding and speech synthesis, diminishing the relative advantage introduced by SpecSpeak. These results highlight that SpecSpeak is particularly effective at improving *perceived responsiveness* (TTFS), while still providing moderate gains in end-to-end efficiency. Optimizing both TTFS and full-generation latency jointly remains an important direction for future work.

G.5 Comparison with Few-Step Diffusion Baselines

To further investigate the latency–quality trade-off in SpeechLLM inference, we compare SpecSpeak against a straightforward acceleration strategy, i.e., reducing the number of diffusion steps in speech synthesis. Unlike SpecSpeak, which overlaps LLM decoding with speech synthesis through speculative co-generation and verification-aware diffusion reuse, reducing diffusion steps directly decreases the computational cost of speech synthesis. We conduct experiments using VITA-Audio on the TriviaQA dataset, evaluating the original 10-step diffusion configuration and three reduced-step variants with 7, 4, and 1 denoising steps. Following the evaluation protocol in Section 4.1, we report the average, P50, and P90 TTFS latency, together with WER and UTMOS. We additionally measure

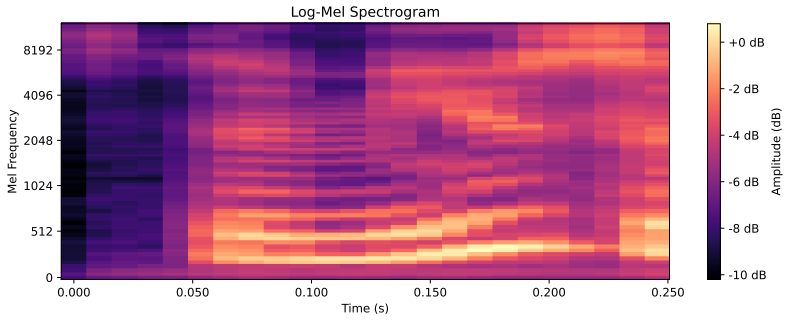
Table 11: Comparison with reduced diffusion-step baselines using VITA-Audio on TriviaQA. We report TTFS latency (ms), WER, UTMOS, and first-chunk UTMOS. The best results are in bold. The second best results are underlined. Few-step diffusion leads to drastic quality degradation, while SpecSpeak balances latency and quality.

Method	TTFS Latency (ms)			Speech Quality		
	Avg. ↓	P50 ↓	P90 ↓	WER ↓	UTMOS ↑	First-Chunk UTMOS ↑
VITA-Audio (10 steps)	334.42	326.57	352.10	**4.05**	**3.76**	**3.47**
VITA-Audio (7 steps)	296.97	297.32	329.51	6.12	3.70	3.39
VITA-Audio (4 steps)	256.72	259.18	282.96	7.34	3.64	3.22
VITA-Audio (1 step)	**230.08**	242.73	**260.38**	9.41	3.55	2.96
VITA-Audio+SpecSpeak	235.81	**220.48**	285.09	4.77	3.71	3.37

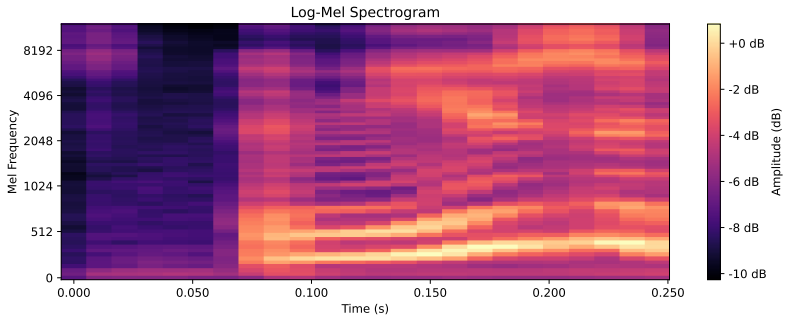
the UTMOS of the first generated speech chunk to assess the perceptual quality of the initial audible response.

As shown in Table 11, reducing the number of diffusion steps consistently decreases TTFS latency but introduces noticeable degradation in speech quality. Specifically, reducing the number of steps from 10 to 1 decreases the average TTFS from 334.42 ms to 230.08 ms, while increasing WER from 4.05 to 9.41. Meanwhile, UTMOS decreases from 3.76 to 3.55, with a more pronounced degradation in first-chunk UTMOS, which drops from 3.47 to 2.96. These results indicate that aggressively reducing diffusion steps compromises both speech intelligibility and perceptual quality, particularly for the initial speech chunk. In contrast, SpecSpeak achieves substantial latency reduction without reducing diffusion steps or compromising speech quality, thereby offering a more favorable latency–quality trade-off.

G.6 Qualitative Speech Quality Analysis

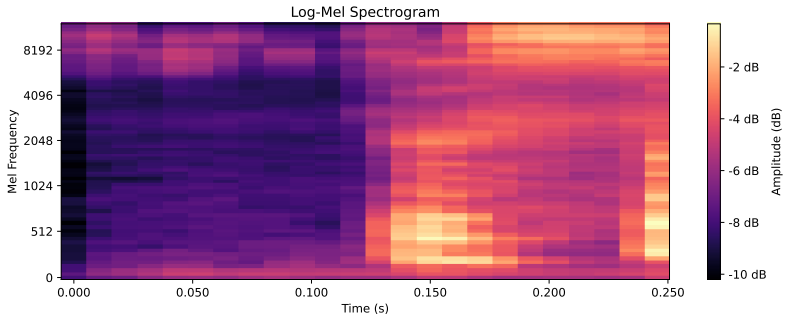


(a) VocalNet output.

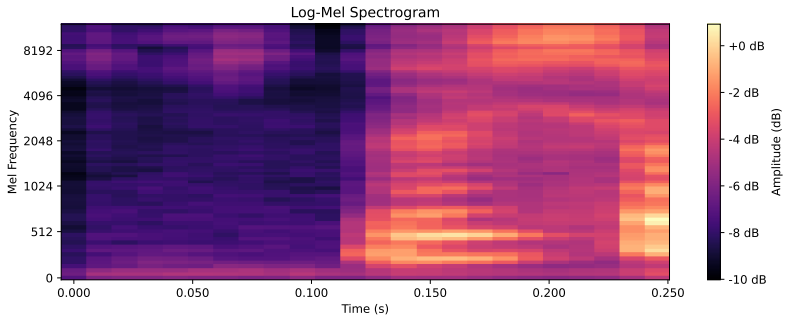


(b) SpecSpeak output.

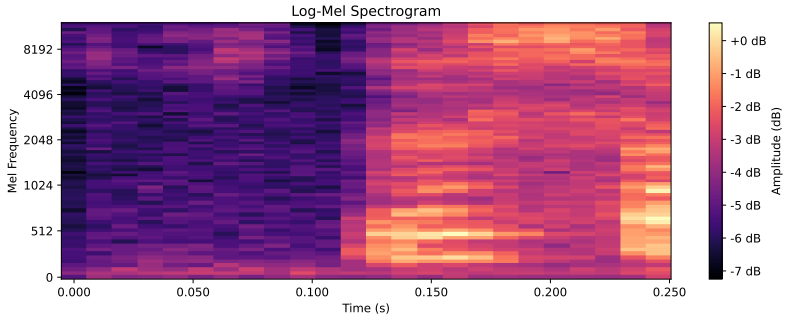
Figure 10: Mel-spectrogram comparison under the high-agreement case. The output generated by SpecSpeak closely matches the original VocalNet output, indicating that SpecSpeak can effectively preserve acoustic structures when draft tokens are fully accepted.



(a) VocalNet output.

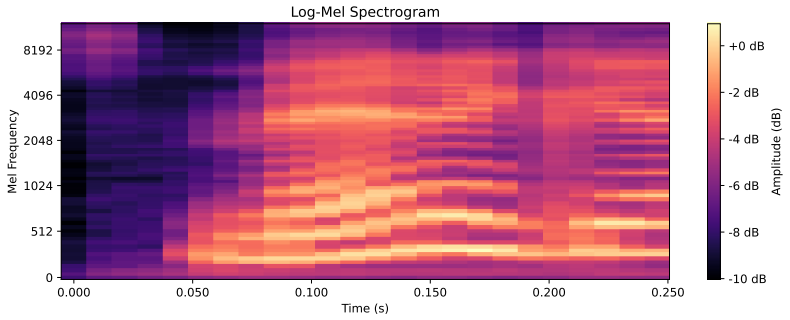


(b) SpecSpeak output.

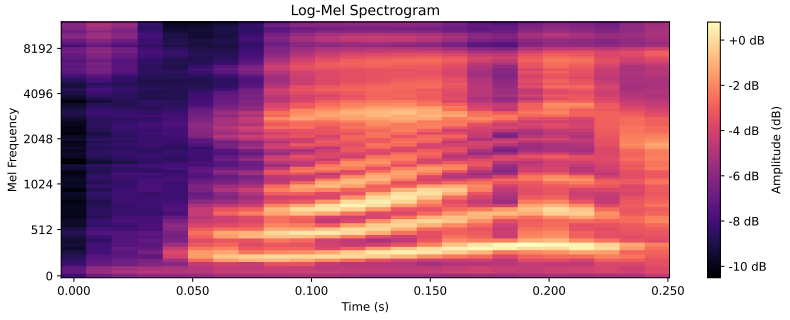


(c) Draft model output.

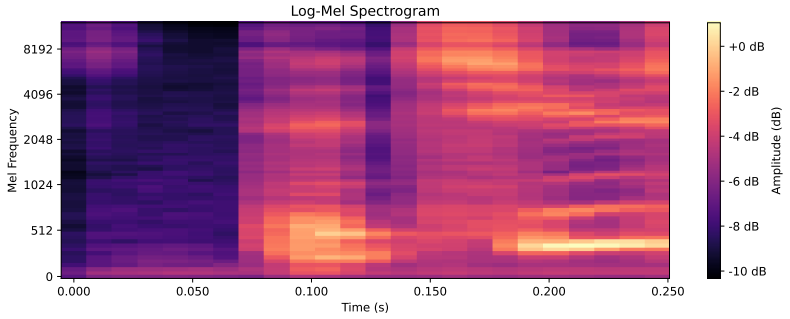
Figure 11: Mel-spectrogram comparison under the near-agreement case. The output generated by SpecSpeak closely matches the original VocalNet output, while the draft model outputs exhibit more noticeable deviations, indicating that SpecSpeak effectively refines the generated speech through verification-aware correction and diffusion reuse.



(a) VocalNet output.



(b) `SpecSpeak` output.



(c) Draft model output.

Figure 12: Mel-spectrogram comparison under the low-agreement case. `SpecSpeak` successfully corrects the speculative generation and preserves spectral structures and temporal patterns comparable to the original VocalNet output, while the draft model outputs exhibit more noticeable deviations.

WER and UTMOS provide quantitative evaluation of intelligibility and perceptual quality for the generated speech. To further examine whether speculative co-generation introduces noticeable acoustic distortion, we additionally provide qualitative speech quality analysis by visualizing the mel-spectrograms produced by different methods. A mel-spectrogram represents the time-frequency structure of speech signals, where the horizontal axis corresponds to time and the vertical axis represents Mel-scaled frequency bins. The color intensity indicates spectral energy at different time-frequency regions. Since mel-spectrograms capture harmonic structures, temporal transitions, and energy distributions of speech, visually similar spectrograms generally indicate similar acoustic characteristics and perceptual quality.

Figures 10, 11, and 12 visualize representative examples under the high-agreement, near-agreement, and low-agreement cases, respectively. For the high-agreement case, we compare the generated mel-spectrograms between the original VocalNet pipeline and `SpecSpeak`. For the near-agreement and low-agreement cases, we additionally visualize the draft model output to illustrate the correction effect introduced by `SpecSpeak`'s speculative verification and diffusion reuse.

Overall, the mel-spectrograms generated by `SpecSpeak` remain highly consistent with those produced by the original SpeechLLM pipeline across all agreement regimes. The overall spectral structures, harmonic patterns, and temporal energy distributions are well preserved, even when speculative corrections are required. In contrast, the draft model outputs in near-agreement and low-agreement cases exhibit more noticeable deviations from the base model outputs. These results demonstrate that the verification-aware correction and diffusion reuse mechanisms in `SpecSpeak` can effectively recover from imperfect draft tokens while preserving the acoustic quality of generated speech.

H Prompts

To evaluate the response correctness, following prior work [14, 25], we adopt an LLM-as-a-judge paradigm using Qwen3-30B-A3B [35]. We report *accuracy* on TriviaQA, WebQuestions, and LLaMA Questions, and evaluate AlpacaEval responses with *normalized scores* in the range of 1 to 10. The prompts for LLM judge are illustrated as follows.

To evaluate the alignment between speech and text responses, we transcribe the response speech by Qwen3-ASR-1.7B [36] and calculate the word error rate (WER), regarding the response answer as the reference text. To assess the overall speech quality, we use the UTMOS [37] to predict mean opinion scores (MOS).

LLM Judge Prompt for TriviaQA, LLaMA Questions and WebQuestions

```
You are a strict and fair judge for QA evaluation.

Given a QUESTION, a list of GOLD ANSWERS (aliases allowed), and a MODEL ANSWER, decide
     whether the model answer should be counted as correct.

Rules:
1. Accept if the model answer clearly provides the same factual answer as ANY gold
     answer, even if phrased differently.
2. Reject if it is ambiguous, contradicts, or gives a different entity/value.
3. If the model answer contains extra text but includes the correct answer unambiguously,
      accept.
4. If the question expects a short entity/value, do NOT accept long unrelated passages
     unless the answer is explicit.
5. Output ONLY valid JSON with key "verdict". No other text.

Return exactly one of:
{"verdict":"YES"}
or
{"verdict":"NO"}

QUESTION:
{question}

GOLD ANSWERS:
{gold_block}
```

```
MODEL ANSWER:
{model_answer}

Now judge correctness. Output ONLY JSON.
```

LLM score prompt for AlpacaEval

```
You are an expert evaluator judging the quality of an AI assistant's answer.

Your task is to rate the answer on a scale from 1 to 10.

Evaluation criteria:
1. Correctness (does it match the expected answer?)
2. Helpfulness (does it provide useful information?)
3. Relevance to the question

Scoring rules:
10 = completely correct and helpful
9 = correct with minor extra text
8 = mostly correct
7 = reasonably helpful but not perfect
6 = partially correct
4-5 = weak answer
1-3 = clearly incorrect

Important judging guidelines:
- Be reasonably generous.
- Minor wording differences should NOT reduce the score.
- If the correct answer appears anywhere in the response, the score should usually be at
      least 8.
- Extra explanation should NOT reduce the score.
- Only give very low scores if the answer is clearly wrong.

Output ONLY JSON.
Return exactly: {"score": X}

QUESTION:
{question}

REFERENCE ANSWERS:
{gold_block}

MODEL ANSWER:
{model_answer}

Provide the score.
```

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 have explicitly claimed our contributions and scope in the abstract and introduction.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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 have discussed the limitations in Sec. 5.

   Guidelines:

   - The answer [N/A] 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: [N/A]

Justification: We include no theoretical results in this paper.

Guidelines:

- The answer [N/A] 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 have provided implementation details in Section 4.1 and Appendix F, describing the datasets, models, evaluation metrics, hardware, and hyperparameters required to reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] 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: The experiments use publicly available models and datasets, and the code will be released to support reproducibility.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We have provided implementation details in Section 4.1 and Appendix F, including the datasets, models, evaluation metrics, hardware, hyperparameters, and model training details, etc.

Guidelines:

- The answer [N/A] 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: [No]

Justification: The paper reports averaged latency over multiple runs, but does not include error bars.

Guidelines:

- The answer [N/A] 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: Appendix G.3 reports the GPU memory usage, power consumption, and GPU utilization of our proposed method.

   Guidelines:

   - The answer [N/A] 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: The research follows the NeurIPS Code of Ethics, and we do not involve human subjects, private data collection, or unsafe data release.

   Guidelines:

   - The answer [N/A] 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 it in Appendix A.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work focuses on improving inference efficiency for SpeechLLM systems and does not introduce new capabilities for generating disallowed or harmful content. The proposed method operates on top of existing models without modifying their underlying training data or alignment mechanisms.

    Guidelines:

    - The answer [N/A] 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: The paper credits the existing models, datasets, evaluation metrics, and baselines used in the experiments through citations.

    Guidelines:

    - The answer [N/A] 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: The new asset introduced by this paper is the system implementation details.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We do not use LLMs for important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.