Benchmarking the Reponsiveness of Open-Source Text-to-Speech Systems

Ha Pham Thien Dinh , Rutherford Agbeshi Patamia , Ming Liu * , Akansel Cosgun

Posted Date: 11 August 2025

doi: 10.20944/preprints202508.0654.v1

Keywords: Text-to-Speech; voice assistant; responsiveness; benchmark; latency; trade-off; real-time

 doi:10.20944/preprints202508.0654.v1

Article

Benchmarking the Reponsiveness of Open-Source Text-to-Speech Systems

Ha Pham Thien Dinh, Rutherford Agbeshi Patamia, Ming Liu* and Akansel Cosgun

Deakin University, Burwood, VIC, Australia 3125
* Correspondence: m.liu@deakin.edu.au

Abstract

This study addresses a significant gap in voice assistant research by evaluating the responsiveness - the speed at which a TTS system generates speech in reaction to input, crucial for maintaining natural, real-time interactions - of open-source text-to-speech (TTS) models—an often overlooked yet critical component for real-time applications. While extensive benchmarking has been performed on speech-to-text and large language models, little work has focused on how efficiently TTS systems respond in live settings—largely because TTS research has historically prioritized subjective quality metrics like naturalness and intelligibility, which are easier to assess through human listening tests than real-time performance; additionally, the lack of standardized, reproducible tools for measuring latency and responsiveness has further limited progress in this area. This work presents the first comprehensive benchmark focused on responsiveness—assessing TTS latency, tail latency, and real-time processing performance across 13 prominent open-source, readily available models, in contrast to commercial systems like Amazon Polly or Google Cloud TTS, which are closed-source and paywalled. Using a standardized single-stream evaluation inspired by MLPerf Inference, the study measures model responsiveness under controlled conditions and also investigates trade-offs between speed and audio quality. Results reveal substantial variability across models, with some achieving sub-second latency suitable for interactive systems, while others fall short of real-time standards. The benchmark highlights performance bottlenecks in autoregressive architectures and identifies parallel and flow-based models as more efficient for low-latency scenarios. Importantly, the proposed framework provides a reproducible foundation for comparing TTS models in latency-sensitive environments and sets a baseline for future research. By focusing on responsiveness, this work contributes to the development of more effective and natural voice interfaces.

Keywords: Text-to-Speech; voice assistant; responsiveness; benchmark; latency; trade-off, real-time

1. Introduction

Voice assistants have become increasingly prevalent in modern society, integrating seamlessly into smartphones, smart speakers, and various IoT devices. The widespread adoption of virtual assistants like Amazon's Alexa, Apple's Siri, and Google Assistant underscores the growing demand for intuitive and hands-free interaction with technology [1]. This trend is propelled by advancements in speech recognition and synthesis technologies, enabling more natural and efficient human-computer communication [2].

The primary and most widely adopted approach for designing voice assistant systems is the modular pipeline architecture. This structure divides the system into specialized components that each handle a distinct task. First, a speech-to-text (STT) engine transcribes the user's spoken input into text. Next, a Natural Language Processing (NLP) component—typically powered by a Large Language Model (LLM)—interprets the text and generates a response. Finally, a text-to-speech (TTS) engine converts the generated text back into natural-sounding speech for the user [3]. This clear division allows each part of the system to leverage domain-specific advances: STT models focus on acoustic

 doi:10.20944/preprints202508.0654.v1

and linguistic accuracy, LLMs handle semantics and intent reasoning, and TTS models synthesize expressive, human-like speech.

An emerging alternative to this modular design is an integrated, end-to-end trainable system that combines all stages—recognition, understanding, and synthesis—into a single model. One prominent example is Moshi [4], a real-time speech-to-speech dialogue model developed by Kyutai [1]. Although Moshi eliminates explicit interfaces between components (no separate ASR (Automatic Speech Recognition), NLP (Natural Language Processing), or TTS (text-to-Speech) modules), its architecture remains logically modular: it uses a large language model core to reason over internally predicted text tokens, while audio tokenizers encode and decode the speech signals. The entire system is trained end-to-end, allowing for tight integration and low latency, but it still relies on internal text-based reasoning and separate token streams for input and output speech.

Overall, the base quality of the integrated approach remains behind that of modular pipelines — Moshi, for example, prioritizes conversational fluidity and low latency over raw quality, and while it achieves decent intelligibility and speaker similarity, it is not yet intended to be competitive against state-of-the-art commercial TTS systems in terms of naturalness and expressiveness, as acknowledged by its authors [4], making the more common modular approach the preferred choice for voice assistant systems. This approach, with distinct components for speech-to-text (STT), natural language processing (NLP), and text-to-speech (TTS), allows for higher-quality outputs by leveraging advancements in each field. Benchmarking the performance of these individual components is critical not only for selecting the most effective models, but also for ensuring system reliability and maintainability—since failures or bottlenecks in a modular pipeline can be isolated and resolved at the component level without retraining the entire system. This enables more targeted optimization and efficient integration of new models, which is essential for building robust voice assistant applications. Moreover, because the system is not end-to-end but modular, any quality or latency issues can be systematically traced to and addressed within the responsible module, streamlining both development and debugging processes.

Among the three core components of voice assistants—speech-to-text (STT), large language models (LLMs), and text-to-speech (TTS) — LLMs have received the most research attention due to rapid advancements in AI. Several benchmarks have been developed to assess their capabilities, such as BIG-Bench, which evaluate over 200 tasks spanning linguistics, logic, and world knowledge to test generalization limits [5], while MMLU measures accuracy across 57 academic subjects to assess reasoning depth [6]. HumanEval focuses on code generation accuracy using pass@k execution metrics [7], and MLPerf Inference benchmarks model latency and throughput under standardized deployment scenarios [8]. Project MPG introduces a combined “Goodness” and “Fastness” score to balance correctness with query-per-second efficiency [9]. Holistic chatbot evaluations such as the E2E Chatbot Benchmark assess semantic similarity against expert responses [10], MT-Bench-101 scores multi-turn conversational coherence [11], and Malode analyzes deployment feasibility in domain-specific contexts [12]. Finally, FACTS Grounding measures faithful, document-grounded generation using multi-judge evaluations across real-world enterprise domains [13]. Together, these benchmarks reflect a shift from accuracy-only metrics toward comprehensive evaluations incorporating latency, grounding, and real-world applicability.

There are also benchmarks that evaluate voice assistants holistically, assessing the entire end-to-end system rather than isolating individual modules, such as AudioBench [14], End-to-End Speech Benchmark (ESB) [15], S2S-Bench [16], VoiceBench [17], and CURATe [18], which will be discussed in subsequent sections. While these works provide critical insights into the LLM component and end-to-end systems’ performance, they leave a major gap in evaluating TTS performance — particularly its responsiveness, which is essential for real-time voice interactions. Unlike LLMs, TTS systems have received relatively little benchmarking attention, despite their direct impact on perceived latency, fluidity of conversation, and overall user experience.

[1] https://kyutai.org/

 doi:10.20944/preprints202508.0654.v1

This work addresses that gap by introducing a benchmark dedicated to evaluating TTS responsiveness for real-time voice assistants, focusing on latency, tail latency, and processing efficiency rather than naturalness or intelligibility. It provides a standardized framework for fair comparisons, establishes performance baselines, and informs future optimization of low-latency TTS systems. Section 2 reviews related evaluations on LLMs, STT, and TTS; Section 3 details the benchmarking methodology; Section 4 describes the experimental setup; Section 5 presents results; and Section 6 discusses broader trends and future directions.

2. Related Works

In this section, we review prior work on text-to-speech (TTS) systems and broader benchmarks designed to evaluate voice assistants end to end, including their TTS components. We trace how benchmarking methodologies have evolved—shifting from a narrow focus on accuracy toward more comprehensive evaluations that account for responsiveness, latency, and real-time performance.

2.1. Text-to-Speech (TTS)

Benchmarking Text-to-Speech (TTS) models has historically focused on assessing speech quality, naturalness, and intelligibility rather than computational efficiency or real-time responsiveness. Many existing evaluations rely on subjective metrics such as Mean Opinion Scores (MOS) - a commonly used metric in speech synthesis evaluation recommended to be the measure of synthesized speech quality by an early work on benchmarking TTS relied on MOS [19], where human listeners rate the naturalness and quality of generated speech on a scale (typically from 1 to 5), with higher scores indicating more human-like and intelligible speech - preference tests, or crowdsourced human assessments. However, as TTS models become increasingly advanced, these methods face limitations in scalability, reproducibility, and responsiveness benchmarking. Several works have attempted to improve TTS evaluation by introducing new benchmarking methodologies, but most remain limited in scope.

Hugging Face's TTS Arena [20,21] is one of the most well-known benchmarking efforts, offering a crowdsourced evaluation platform where users compare TTS models by listening to side-by-side samples and voting on which sounds more natural, clear, and expressive. While this approach provides valuable human feedback, it does not explicitly or systematically assess output accuracy; however, inaccuracies in word generation can indirectly affect scores since they influence user preference. The platform also has notable limitations: evaluations are inherently subjective and may shift with changing user preferences, it does not measure latency or computational efficiency, and it relies on participant activity—causing newer models to be underrepresented or unevenly rated.

Picovoice [2] [22] takes a different approach, specifically focusing on TTS inference speed. This Picovoice TTS Latency Benchmark provides a lightweight and extensible benchmarking tool to evaluate TTS models' response times in virtual assistants and edge computing environments. It measures latency at various processing stages, including text preprocessing, model inference, and waveform generation. However, while Picovoice's methodology offers a useful tool for assessing TTS speed, it is limited in scope. The framework does not provide a comprehensive quality assessment, meaning models that generate speech quickly but sound unnatural may still score highly. Additionally, its benchmarking process is primarily designed for local and embedded systems, which focus solely on on-device processing and therefore do not account for factors like network latency or scalability—potentially making it less relevant for evaluating cloud-based or server-side TTS applications.

Artificial Analysis [3] [23] presents a more holistic evaluation framework, assessing both quality and performance. Their methodology benchmarks various commercial and open-source TTS models across key dimensions, including naturalness, intelligibility, and latency. They employ serverless API-based evaluation, where models are tested in real-world conditions to measure their inference time and quality trade-offs. However, like TTS Arena, Artificial Analysis relies on human evaluations for speech

[2] https://github.com/Picovoice/tts-latency-benchmark
[3] https://artificialanalysis.ai/text-to-speech

 doi:10.20944/preprints202508.0654.v1

quality, which introduces subjectivity. Moreover, their focus on commercial API-based models may not fully capture open-source or offline TTS systems, limiting its applicability in resource-constrained environments.

Another work [24] extends benchmarking efforts by providing a comparative analysis of six leading TTS models, using a structured dataset of 500 diverse text prompts. Each output was rated by three expert human evaluators using structured criteria, ensuring a controlled and consistent evaluation process. Unlike previous works, this study incorporates structured feedback on intelligibility, pronunciation accuracy, and audio quality. However, it does not explicitly benchmark latency or responsiveness, making it less relevant for real-time applications. Additionally, since Labelbox focuses on commercial solutions, open-source models are not evaluated, leaving gaps in benchmarking freely available TTS systems.

Finally, Minixhofer et al. (2024) provides one of the most comprehensive attempts to establish an objective, scalable evaluation metric for TTS models [25]. The Text-to-Speech Distribution Score (TTSDS) benchmark quantifies synthetic speech quality by analyzing how closely a model's output distribution aligns with real human speech. Unlike MOS-based benchmarks, it removes reliance on subjective ratings, enabling reproducible and cross-study comparisons. Intelligibility is evaluated by passing the generated speech through speech-to-text (STT) models (wav2vec 2.0 and Whisper) to obtain transcripts, which are then compared with the original text using Word Error Rate (WER). Alongside intelligibility, TTSDS incorporates four additional factors—prosody, speaker identity, environmental artifacts, and general distribution similarity—summarized below.

- **Intelligibility** – Measured using Word Error Rate (WER) from STT models.
- **Prosody** – Evaluated via pitch variation and rhythm consistency.
- **Speaker Identity** – Measures how closely the generated voice matches natural human speech by comparing voice features (e.g., tone and timbre) between synthetic and real recordings.
- **Environmental Artifacts** – Measures unwanted noise or distortions in the generated audio, with higher artifact levels indicating lower audio quality.
- **General Distribution Similarity** – Uses self-supervised embeddings to compare TTS outputs with real speech data.

It benchmarks 35 TTS systems developed between 2008 and 2024, demonstrating that objective factor-based scoring correlates strongly with human evaluations. However, TTSDS does not explicitly evaluate latency, meaning it does not fully address responsiveness, which is critical for real-time TTS applications.

2.2. End-to-End Systems

In addition to benchmarks that evaluate the three modular components separately, several recent works have introduced benchmarks that assess speech-enabled systems in an end-to-end manner.

AudioBench [14] was introduced as a benchmark for audio large language models (AudioLLMs) that take audio as input and generate textual responses to natural-language instructions. It unifies evaluation across speech semantics, environmental audio understanding, and paralinguistic voice traits through 8 tasks and 26 datasets, 7 of which are newly collected or adapted. These tasks span areas such as speech understanding (e.g., automatic speech recognition and question answering), audio-scene understanding (e.g., audio captioning), and voice-related attributes (e.g., emotion and accent recognition). The benchmark tests robustness by varying prompt templates and input lengths (from seconds to minutes), totaling over 400 hours and 100k samples. Since the benchmark outputs are free-form text rather than fixed choices, AudioBench uses a model-as-judge approach, where large language models automatically evaluate the generated answers instead of relying on human annotators. The authors report high agreement between different judge models, validating the reliability of this automated scoring method.

S2S-Bench [16] is designed to measure "intelligence degradation," or the drop in reasoning ability when a model processes audio tokens instead of text tokens. It compares two conditions: speech-

 doi:10.20944/preprints202508.0654.v1

to-text (S→T), where audio input is tokenized and processed by the model, and a text-to-text (T→T) upper bound using the same semantic content. Each test sample contains two continuations — a correct (positive) and an incorrect (negative) option — and the model's task is to prefer the positive one. Perplexity, a standard language modeling metric indicating how confident the model is about a sequence (lower means more confident), is used to score each option. Accuracy is defined as the proportion of times the positive continuation receives lower perplexity than the negative, and the benchmark also analyzes the perplexity "gap" between the two options to gauge how strongly the model prefers the correct answer. Results show a consistent performance drop in the S→T setting across all tested reasoning tasks, confirming that audio tokenization introduces a measurable degradation relative to the text-only upper bound.

VoiceBench [17] introduces a benchmark for assessing how effectively LLM-based voice assistants interpret spoken commands in realistic conditions. Unlike traditional speech-to-text (STT) tests, it examines robustness to factors like speaker variation, background noise, and linguistic complexity, covering tasks in knowledge retrieval, instruction following, and safety adherence. Results show notable performance gaps between end-to-end voice assistants and modular ASR-LLM pipelines, with current models struggling under noisy or diverse inputs, underscoring the need for greater real-world resilience.

Conversely, CURATe [18] evaluates the ability of conversational AI to sustain personalized alignment during interactions. It benchmarks ten leading LLM-based assistants across 337 scenarios, focusing on their capacity to respect user-specific safety constraints and contextual preferences. Findings highlight recurring issues such as neglecting safety-critical instructions, excessive agreement (sycophancy), and inattentiveness to personalized context, while demonstrating that prompting models to explicitly acknowledge safety cues can enhance adherence. CURATe thus offers a structured framework for measuring consistency and trustworthiness in conversational AI.

Overall, while significant progress has been made in benchmarking TTS models, existing evaluations remain fragmented. Crowdsourced evaluations like TTS Arena [20,21] provide valuable user feedback but lack latency measurements. Speed-focused frameworks like Picovoice's [22] benchmark effectively assess inference time but do not consider speech quality. API-driven benchmarks such as Artificial Analysis [23] and Labelbox [24] provide structured evaluations of commercial TTS models but exclude open-source solutions. Finally, TTSDS [25] represents a strong step toward objective evaluation but does not address responsiveness—the primary focus of this study. Given these limitations, this research aims to establish a dedicated, open-source benchmark for evaluating TTS responsiveness, filling the gap left by prior methodologies and providing insights tailored for real-time voice assistant applications.

3. Benchmarking Methodology

This project is influenced by the MLPerf Inference framework [8], a widely recognized benchmark suite for evaluating machine learning systems across different deployment scenarios. Among its four defined scenarios—single-stream, multistream, server, and offline—we adopt the single-stream setting, which best reflects real-time voice assistant usage where queries are processed sequentially for a single user. In our evaluation, we feed each model datasets of text from different categories - which will be discussed in the experiment section and record the latency for every input. We then summarize these results using descriptive statistics, primarily median latency and tail latency (e.g., 90th percentile), to capture both typical and worst-case performance across the distribution. This approach reflects realistic usage patterns rather than single-point measurements and provides a more reliable picture of model responsiveness.

This paper defines TTS input processing as the sequence where text is received and converted into an output audio file. Because streaming support varies—some models offer native streaming while others rely on external tools—evaluating only file generation time ensures fair, consistent measurement of responsiveness across all models.

 doi:10.20944/preprints202508.0654.v1

For speech processing models, the task is to transcribe audio input into text accurately and efficiently. The following metrics are chosen for evaluating model responsiveness and quality:

- **Latency:** Measures the time from when the model begins processing the text input to when the generated audio file is fully produced. This study reports the **median latency**, which summarizes typical synthesis speed across the dataset while minimizing the effect of outliers. Lower median latency directly indicates faster audio generation, which is critical for real-time voice assistant performance.

$$\text{Latency} = t_{\text{audio\_ready}} - t_{\text{text\_received}} \tag{1}$$

  where:

  - $t_{\text{text\_received}}$ is the timestamp when the text input is passed to the TTS model.
  - $t_{\text{audio\_ready}}$ is the timestamp when the audio file generation is completed.

- **Tail Latency:** Captures worst-case performance by measuring the 90th percentile latency across all samples. This reflects edge-case delays relevant to voice assistant interactions; keeping this value low is essential for maintaining smooth and responsive user experiences.

$$\text{Tail Latency} = P_{90} \tag{2}$$

  where:

  - P_{90} is the 90th percentile latency across the dataset.

In addition to latency metrics, this paper evaluates **Audio Quality** as a primary measure of synthesis performance. These metrics directly assess the clarity, naturalness, and intelligibility of the generated speech, focusing on phrases and sentences rather than isolated single-word utterances. In this study, audio quality is further examined alongside responsiveness to investigate whether there is any trade-off between synthesis speed and perceived quality.

To evaluate audio quality, this study uses a speech-to-text (STT) model to transcribe the audio generated by each TTS system back into text. The resulting transcription is then compared against the original input text to measure how accurately the TTS model preserved the intended content. Both the ground-truth input text and the STT-produced transcription undergo normalization following Whisper's evaluation guidelines [26], which include lowercasing, punctuation removal, whitespace standardization, contraction normalization, and numeric formatting. This preprocessing ensures that the comparison focuses on semantic accuracy rather than superficial formatting differences, providing a fair assessment of how faithfully each TTS model conveys the intended speech content.

- **Percentage of Incorrect Audio Files:** The proportion of generated audio files whose transcriptions, after being converted back to text via STT and normalized as described above, do not exactly match the original input text. In other words, this metric is based on strict string equality between the normalized input and output texts.
- **Overall WER:** A cumulative measure of word error rate (WER) across all transcriptions. Word Error Rate (WER) is a common metric used to evaluate transcription accuracy, calculated for a single sample as:

$$WER = \frac{S + D + I}{N} \tag{3}$$

  where:

  - S is the number of substitutions (incorrectly transcribed words),
  - D is the number of deletions (missing words),
  - I is the number of insertions (extra words added),
  - N is the total number of words in the reference transcript.

 doi:10.20944/preprints202508.0654.v1

For multiple transcriptions, the **Overall WER**, used by Whisper [26] aggregates WER across all samples by summing the total number of errors (substitutions, deletions, and insertions) across all transcriptions and dividing by the total number of words in all reference transcripts:

$$\text{Overall WER} = \frac{\sum(S_i + D_i + I_i)}{\sum N_i} \tag{4}$$

where i represents each individual sample in the dataset. This method provides a cumulative error rate that reflects the overall performance of the transcription system across a dataset rather than an average of individual WER values.

- **Median WER (Mismatched Files):** In addition to the Summation-Based WER following Whisper's methodology, we also compute the Median WER specifically for files with mismatched transcriptions, providing further insight into transcription accuracy variability.

4. Experiments - Benchmarked Text Inputs

This section describes the experiments conducted to measure each model's performance, detailing the datasets of text inputs used as well as models tested.

4.1. Evaluation Categories

First, to reflect the models' capabilities of naturally transforming text into speech in conversations, we look into various elements in human speech - where certain sentence types consist solely of words or short phrases that can function as complete, standalone utterances. We also take into account the elements that are present in dialogues, including utterances, backchannel words such as "uh-huh", "I see" or filler & transition words such as "well", "so", "anyway" to define the range of inputs for models testing. Accordingly, this project evaluates single-word utterances and two-word phrases that can function as complete sentences. Furthermore, full-length sentences are incorporated to capture typical structures observed in standard written and spoken communication. A guideline for scientific writings suggested 12-17 words to be the optimal length [27], while another one indicated today's experts recommended 15-18 words per sentence [28]. To accommodate these guidelines, we incorporate 12-word sentences and 18-word sentences into our benchmarking, and to sum up, came up with the following criterias for experimentation:

Table 1. Categorization of Benchmarked Text Input Types with Word Count, Sources, and Data Size.

Input Type	# Words	Justification	Source	# Data Points
Single Words				
1-syllable words	1	Single words containing exactly one syllable, filtered and deduplicated.	CMU Dictionary [29,30]	5,352
2-syllable words	1	Single words containing exactly two syllables, filtered and deduplicated.	CMU Dictionary [29,30]	13,549
Most common in dialogues	1	Frequently spoken standalone words (questions, answers, imperatives, etc.).	COCA [31]	501

 doi:10.20944/preprints202508.0654.v1

Table 1. *Cont.*

Input Type	# Words	Justification	Source	# Data Points
2-Word Phrases				
LLM-generated	2	Random 2-word phrases used as standalone expressions, deduplicated.	LLaMA 3 8B (LLM-generated)	5,659
Most common	2	Most frequent spoken 2-word N-grams based on raw frequency.	iWebCorpus [32]	504
Sentences with Specific Lengths				
12-word sentences	12	Minimum recommended sentence length for benchmarking.	LLaMA 3 8B (LLM-generated)	4,994
18-word sentences	18	Maximum recommended sentence length for benchmarking.	LLaMA 3 8B (LLM-generated)	5,506

To ensure a robust evaluation, while latency is computed for all input categories, audio quality metrics are assessed only on 12-word and 18-word sentences generated by Llama3-8B. This choice avoids the limitations of single-word evaluations: while the benchmark includes standalone words (e.g., "Yes," "Stop"), such cases can yield misleading accuracy scores because multiple pronunciations may be valid and the lack of surrounding context provides few cues for disambiguation. Multi-word phrases and sentences, by contrast, more closely resemble natural speech patterns and offer richer contextual information, resulting in a more reliable assessment of audio quality.

4.2. TTS Models

This study focuses on evaluating the most popular and widely-used TTS models, as identified through extensive literature and community engagement metrics—such as GitHub stars or likes—alongside models developed by reputable service providers. Since voice assistants typically do not require voice cloning or dynamic multi-voice capabilities, we prioritize models trained on single-speaker datasets. We also include models trained on multiple voices, provided that each voice is independently preset—meaning the model is trained on specific voices individually and allows users to choose from those predefined options.

In many TTS systems, the core model is responsible for converting input text into intermediate audio representations, typically mel-spectrograms. These spectrograms must then be transformed into actual audio waveforms using a vocoder. Therefore, unless the TTS model is fully end-to-end, a separate vocoder is required to synthesize the final audio output. In this study, we focus on TTS models that perform the text-to-spectrogram transformation, however still making use of vocoders where applicable. The list of models and architectures evaluated is as follows:

Table 2. Summary of TTS models evaluated in this study.

Model	Architecture / Approach	Setup (Vocoder / Dataset)	Year	Stars/Forks	Key Features
BarkTTS [33]	Transformer text-to-audio, semantic token-based	EnCodec [34]	2023	N/A	Multilingual, includes nonverbal audio and music

 doi:10.20944/preprints202508.0654.v1

Table 2. *Cont.*

Model	Architecture / Approach	Setup (Vocoder / Dataset)	Year	Stars/Forks	Key Features
VALL-E X [35]	Neural codec language model	EnCodec [34]	2023	N/A	Zero-shot voice cloning, cross-lingual synthesis
Neural HMM [36]	Probabilistic seq2seq with HMM alignment	Hifigan 2 [37] / LJSpeech [38]	2022	41.6k / 5.4k	Stable synthesis, adjustable speaking rate, low data need
Overflow [39]	Neural HMM + normalizing flows	Hifigan 2 [37] / LJSpeech [38]	2022	41.6k / 5.4k	Expressive, low WER, efficient training
Edge-TTS [40]	Cloud API	Cloud Backend	2021	N/A	Lightweight, easy integration, multilingual
FastPitch [41]	Parallel non-autoregressive, pitch-conditioned	Hifigan 2 [37] / LJSpeech [38]	2021	41.6k / 5.4k	Low latency, controllable prosody, real-time capable
VITS [42]	End-to-end VAE + adversarial learning	None / LJSpeech [38]	2021	41.6k / 5.4k	Unified pipeline, high naturalness, expressive prosody
Glow-TTS [43]	Flow-based, monotonic alignment	Multiband MelGAN [44] / LJSpeech [38]	2020	41.6k / 5.4k	Fast inference, robust alignment, pitch/rhythm control
Capacitron [45]	Tacotron 2 + VAE for prosody	Hifigan 2 [37] / Blizzard 2013 [46]	2019	41.6k / 5.4k	Fine-grained prosody modeling, expressive synthesis
Tacotron 2 (DCA/DDC) [47]	Two-stage seq2seq, attention-based	Hifigan 2 [37], Multiband MelGAN [44] / LJSpeech [38]	2018	41.6k / 5.4k	High naturalness, improved alignment with DDC/DCA
Google TTS (gTTS) [48]	Cloud API (Google Translate)	Cloud backend	2014	N/A	Lightweight, easy integration, multilingual
Microsoft Speech API 5.4 (Microsoft Speech) [49]	Concatenative / parametric synthesis	Built-in voices / Windows	2009	N/A	Commercial baseline, real-time, multilingual support

The experiments are conducted on a machine equipped with a 13th Gen Intel Core i9-13900HX processor (24 cores, 32 threads) and an NVIDIA GeForce RTX 4070 Laptop GPU with 8 GB of VRAM, providing sufficient computational resources for text-to-speech benchmarking. As this setup reflects a high-end consumer device rather than specialized server hardware, it closely mirrors real-world conditions in which virtual assistants are typically deployed, ensuring the results remain practically relevant. To ensure a fair and consistent comparison, all experiments are conducted using each model's default configurations, with no manual hyperparameter tuning or additional training. The models are evaluated exactly as provided by their official implementations or APIs, relying on pre-trained

 doi:10.20944/preprints202508.0654.v1

weights and configurations publicly released by their developers. No fine-tuning or custom training was performed in this study.

5. Results

This section provides an overview of model performance, focusing on both responsiveness and audio quality for the 12-word and 18-word sentence input types. We begin with a broad discussion of overall trends before examining the results and their implications in greater details in the subsequent section.

5.1. Responsiveness

Overall, across both median and 90th percentile rankings, FastPitch consistently dominates performance. It ranks first or second in nearly every category, achieving first place in both median and 90th percentile for most inputs such as 2-syllable words, most common words, and LLM 2-word phrases. Even in longer sequences like 12-word and 18-word sentences, it maintains top-three positions (median rank 1, P90 rank 3). This highlights FastPitch's exceptional balance between typical and worst-case latency, ensuring predictable real-time responsiveness across varying input complexities.

Microsoft Speech follows closely behind and frequently secures first or second place in both median and P90 rankings across almost all categories. It is ranked first in median latency for 1-syllable words, tied first for most common words, and first again for 12-word sentences. Even at 90th percentile, Microsoft Speech rarely falls below rank 3, showing remarkable stability despite being a non-neural, CPU-based system. Its consistent top-tier performance demonstrates how deterministic architectures can outperform many neural models under variable conditions.

GlowTTS and VITS also perform strongly, staying within the top five in most categories. GlowTTS achieves ranks as high as third for both median and P90 in shorter inputs (e.g., 1-syllable and 2-syllable words) and second for 18-word sentences at P90, though occasionally dropping to fifth in LLM 2-word phrases. VITS shows similar behavior, with steady mid-upper rankings (median rank 4–6) and slightly worse tail performance (P90 ranks 4–5), indicating minor degradation as inputs grow more complex.

The middle tier consists of NeuralHMM, Overflow, and Tacotron2-DCA. NeuralHMM ranks mid-table, for example 8th in median and 6th in P90 for 1-syllable words, and improves to 4th in most common words (median). However, its P90 scores degrade with input length, reaching 7th for 18-word sentences. Overflow follows a similar trajectory, ranking around 6th–10th across categories; for instance, it is 10th median and 7th P90 for 1-syllable words and 9th/8th for most common words. Tacotron2-DCA places better at median (2nd–4th in shorter phrases) but worsens in P90, hitting 10th for 2-syllable words and 9th for most common words, indicating occasional latency spikes.

Capacitron50 and Capacitron150 occupy mid-lower rankings. Capacitron50 is typically 6th–7th median (e.g., 6th for 1-syllable words, 7th for most common words) and similarly placed for P90. Capacitron150 fares slightly worse, with 9th median and 9th P90 for 1-syllable words and 7th–8th across other categories. The minimal performance gap between the 50- and 150-unit variants suggests increased model size does not significantly alleviate tail-latency issues.

At the bottom, Vall-E-X and BarkTTS consistently rank last, with extremely high median and P90 latencies across all input lengths. Vall-E-X sits consistently at 13th–14th for both median and P90, for example 13th median and 13th P90 in most common words, and 14th in 18-word sentences. BarkTTS is similarly poor, often tied for 14th (e.g., 14th median and 14th P90 in 1-syllable and 2-syllable words) and showing no improvement even in shorter inputs. These rankings reflect their computationally heavy architectures and lack of optimization for low-latency inference.

 doi:10.20944/preprints202508.0654.v1

Table 3. Median (M) and 90th Percentile (P90) Latencies Across All Benchmarked Text Input Categories (s).

Model	1-syllable word		2-syllable word		Most Common word		LLM 2-word phrase		Most Common 2-word		LLM 12-word sentence		LLM 18-word sentence	
	M	P90	M	P90	M	P90	M	P90	M	P90	M	P90	M	P90
BarkTTS [33]	5.73	13.01	4.41	12.11	1.49	1.90	2.01	5.05	7.45	12.42	31.05	78.86	14.75	36.00
Vall-E-X [35]	2.20	11.72	1.75	5.59	2.63	4.36	7.53	15.78	1.73	5.93	10.09	13.90	17.79	45.38
NeuralHMM [36]	0.27	0.33	0.28	0.35	0.24	0.34	**0.06**	**0.12**	0.20	0.24	0.45	0.70	0.64	0.85
Overflow [39]	0.33	0.41	0.23	0.28	0.34	0.53	0.37	0.45	0.22	0.27	0.68	0.88	1.56	1.99
EdgeTTS [40]	1.11	1.58	1.29	1.59	0.83	0.98	1.19	1.98	0.87	1.16	1.64	2.26	1.78	2.41
Fastpitch [41]	**0.06**	**0.09**	**0.04**	**0.05**	**0.05**	**0.12**	**0.04**	**0.07**	**0.04**	0.10	**0.06**	0.17	0.13	0.30
VITS [42]	0.29	0.33	0.22	0.31	0.19	0.28	0.25	0.29	0.20	0.23	0.28	0.42	0.51	0.73
GlowTTS [43]	0.10	0.14	0.09	0.13	0.09	0.14	0.23	0.29	**0.05**	0.07	0.08	**0.13**	**0.09**	**0.15**
Capacitron50 [45]	0.23	0.35	0.24	0.39	0.20	0.32	0.29	0.37	0.19	0.37	0.35	0.48	0.66	0.94
Capacitron150 [45]	0.25	0.51	0.28	0.58	0.22	0.42	0.33	0.69	0.22	0.50	0.64	0.86	0.39	0.62
Tacotron2-DCA [47]	0.10	0.53	0.18	1.16	0.16	0.70	0.09	0.46	0.18	0.50	0.44	0.62	0.53	0.77
Tacotron2-DDC [47]	0.17	16.49	0.26	15.95	0.37	8.19	0.45	21.99	0.16	8.14	0.91	1.24	0.93	1.52
GTTS [48]	0.43	0.65	0.45	0.58	0.58	0.65	0.42	0.62	0.58	0.64	0.68	0.78	0.93	1.39
Microsoft Speech [49]	**0.06**	**0.07**	**0.06**	**0.08**	**0.06**	**0.07**	0.07	**0.08**	0.06	**0.07**	**0.07**	**0.08**	**0.07**	**0.08**

A particularly striking outlier is Tacotron2-DDC. While its median ranks are mid-tier (e.g., 4th for 1-syllable words, 8th for 2-syllable words), its P90 ranks collapse to last place in almost every input (14th P90 for 1-syllable words, 13th–14th for longer inputs). This discrepancy stems from pathological inference behavior, where stop-token failures and alignment drift cause excessively long outputs even for short inputs.

Cloud-based systems like GTTS and EdgeTTS show distinct patterns. GTTS generally ranks in the 10th–11th range for both median and P90, such as 10th median and 10th P90 for 1-syllable words, reflecting moderate but stable performance. EdgeTTS fares slightly worse: 12th in both median and P90 for 1-syllable words and 11th–12th for most categories, with tail latencies particularly affected by network fluctuations. These results highlight how cloud services, while convenient, struggle to match the deterministic low-latency performance of local neural models like FastPitch or GlowTTS.

5.2. Audio Quality

In terms of output audio quality, Microsoft Windows Speech, GTTS, and EdgeTTS emerge as the best overall performers in terms of audio quality. All three achieve extremely low incorrect file rates — between 2–4% for 12-word inputs and around 4% for 18-word inputs — with overall WER consistently below 0.5. Median WER remains low (approx. 10–16), demonstrating both stability and precision. Their consistent accuracy across sentence lengths places them firmly at the top of the ranking. Notably, Microsoft Windows Speech achieves this despite being CPU-based and non-neural, highlighting the efficiency of its deterministic synthesis pipeline. GTTS and EdgeTTS, despite being cloud-based, maintain similarly high quality, with minimal degradation even as input length doubles.

Among locally hosted neural models, VITS demonstrates the strongest performance in audio quality. It records incorrect rates of just 5.11% (12-word) and 7.11% (18-word), combined with the lowest overall WER among local models (approx. 0.65) and consistently low median WER values (approx. 5–8%). FastPitch and GlowTTS follow closely, with slightly higher incorrect rates (7–20%) and WER values around 0.7–1.8, remaining highly competitive in quality despite their focus on speed.

 doi:10.20944/preprints202508.0654.v1

Mid-tier results are observed for Tacotron2-DCA, Tacotron2-DDC, and Overflow. These models maintain moderate incorrect rates (approx. 12–20%) and overall WER between 1–3. While Tacotron2-DDC suffers from severe tail-latency issues in responsiveness, its quality metrics remain stable, suggesting that overly long outputs do not necessarily lead to higher transcription mismatches. Overflow and Tacotron2-DCA show similar performance, with modest error rates but less consistency compared to top-tier models.

Table 4. TTS Models' Audio Quality for LLM-Generated 12-Word and 18-Word Sentence Inputs

	12-Word Sentence Inputs			18-Word Sentence Inputs		
Model	% Incorrect	Overall WER	Median WER	% Incorrect	Overall WER	Median WER
BarkTTS [33]	26.72	5.28	8.33	32.30	4.20	5.56
Vall-E-X [35]	26.36	4.04	8.33	32.67	3.75	5.56
NeuralHMM [36]	51.44	8.48	16.67	62.78	8.50	11.11
Overflow [39]	12.06	1.38	8.33	16.32	1.28	5.56
EdgeTTS [40]	**3.02**	0.41	16.67	**4.02**	0.38	11.11
Fastpitch [41]	7.15	0.82	8.33	9.43	0.77	5.56
VITS [42]	**5.11**	0.65	8.33	**7.11**	0.65	5.56
GlowTTS [43]	14.92	1.80	8.33	20.03	1.69	5.56
Capacitron 50 [45]	17.01	3.90	16.67	23.87	4.77	11.11
Capacitron 150 [45]	28.19	7.94	16.67	23.64	4.89	11.11
Tacotron 2 - DCA [47]	16.01	2.01	8.33	20.40	1.99	5.56
Tacotron 2 - DDC [47]	12.56	2.68	8.33	19.68	2.84	5.56
GTTS [48]	**2.84**	0.40	16.67	**4.25**	0.39	10.82
Microsoft Speech [49]	**2.86**	0.39	16.67	**4.09**	0.39	10.53

NeuralHMM and the Capacitron series perform worse in terms of quality. NeuralHMM exhibits error rates exceeding 50% and overall WER around 8.5, indicating frequent synthesis failures or alignment errors. Capacitron50 and Capacitron150 fare slightly better but still remain in the lower tier, with 17–28% incorrect rates and overall WER between 4–8, reflecting inconsistent handling of longer phrases.

Vall-E-X and BarkTTS consistently deliver the lowest quality scores. Both models show incorrect rates exceeding 25–32% and overall WER above 3.5–5, with median WER comparable to low-performing models despite significantly heavier architectures. Their degradation worsens with longer inputs, highlighting poor optimization for short prompts and limited robustness to input length scaling.

Overall, cloud-based models (GTTS, EdgeTTS) and in-built Microsoft Windows Speech dominate the quality metrics, showing the most accurate and stable outputs, while VITS leads among locally hosted neural architectures. FastPitch and GlowTTS also maintain high quality with slightly more variability, while models like Vall-E-X, BarkTTS, and Capacitron variants consistently trail behind due to higher error rates and poor robustness.

6. Discussions & Conclusion

6.1. High-Performing Models: Strong Responsiveness with Minimal Quality Trade-offs Across Input Categories

One of the primary aims of this benchmarking study was to identify whether any text-to-speech (TTS) models could deliver both low latency and high audio quality consistently across a range of input lengths. Most models exhibited trade-offs — excelling in one dimension while underperforming in the other. However, Microsoft Speech stands out as the clear overall leader. Despite being a CPU-based, non-neural system, it achieves top-tier responsiveness and audio quality simultaneously, ranking first or second in nearly every median and 90th percentile latency category and also securing the lowest error rates in audio quality metrics (approximately 2–4% incorrect files, overall WER around 0.39-0.41).

 doi:10.20944/preprints202508.0654.v1

Its deterministic design avoids pathological tail-latency spikes and maintains pristine transcription fidelity even for long inputs, making it uniquely suited for latency-critical real-time applications.

In terms of pure responsiveness, FastPitch remains the fastest neural model in this benchmark. Its feed-forward, non-autoregressive design consistently ranks first in median and tail latency across almost every category, including 1- and 2-syllable words, most common words, and short phrases. Even in longer sentences (12-word and 18-word), it maintains top-three positions, highlighting its predictability under load. However, FastPitch's audio quality lags behind the top tier: its incorrect rates are slightly higher (around 7–9%), and its overall WER (around 0.7-0.8) is above the best performers. While still competitive and suitable for real-time systems where speed is paramount, its quality ceiling prevents it from unseating Microsoft Speech as the best all-rounder.

Conversely, EdgeTTS and Google's GTTS dominate the audio quality leaderboard. Both achieve error rates in the 2–4% range and overall WER below 0.5, rivaling or even matching Microsoft Speech for transcription fidelity. However, their cloud-based nature introduces significant responsiveness issues. GTTS performs mid-tier in latency (10th-11th ranks across most inputs), while EdgeTTS frequently ranks near the bottom (11th-12th at P90) due to network variability and API response times. These results underline a key trade-off: while cloud services can deliver exceptional quality, their unpredictable latency makes them unsuitable for deterministic real-time deployments.

Among locally hosted neural architectures, VITS strikes the best balance between speed and quality. It ranks mid-upper for latency (median ranks 4-6, P90 ranks 4-5) and delivers the lowest WER among all neural models (approximately 0.65), with incorrect rates around 5-7%. This combination makes VITS a strong candidate where moderate latency is acceptable but higher audio fidelity is desired, such as offline synthesis or user-facing content generation.

Other models - including GlowTTS, NeuralHMM, Overflow, Tacotron2-DCA, and Capacitron variants - occupy the mid-tier, with moderate rankings in both responsiveness and quality. GlowTTS is competitive at low latency but slightly less stable at P90; NeuralHMM and Overflow degrade as input length grows; and Capacitron 50/150 gain little from scaling, remaining slower and less accurate than top performers.

At the bottom, Tacotron2-DDC, Vall-E-X, and BarkTTS show severe limitations. Tacotron2-DDC suffers catastrophic tail-latency failures (e.g., 16.49s P90 for 1-syllable words) due to attention/stop-token misalignment, despite decent median scores. Vall-E-X and BarkTTS, meanwhile, consistently rank last in both latency and quality: their heavy autoregressive pipelines result in extreme latencies (median >2s, P90 >12-45s) and high error rates (25-32%), making them impractical for any interactive use case.

In summary, Microsoft Speech emerges as the best all-round TTS system in this study - unrivaled in its ability to combine ultra-low latency with near-perfect quality. FastPitch is the fastest neural option, ideal when responsiveness outweighs quality, while EdgeTTS and GTTS lead in fidelity but are hampered by cloud-induced delays. For local neural solutions balancing both, VITS provides the strongest alternative, though it still trails Microsoft Speech in overall stability.

6.2. Unusual Patterns in Performance Metrics

While most models exhibit predictable behavior in terms of latencyand synthesis quality, certain models display unexpected performance trends that merit closer examination. These anomalies provide deeper insights into the internal workings of the architectures and highlight potential optimizations or inefficiencies in model design.

A particularly notable irregularity arises in the performance of Tacotron2-DDC (Double Decoder Consistency). While most models maintain predictable scaling between median and tail latencies, Tacotron2-DDC exhibits extreme instability. For example, in the 1-syllable word category it records a median latency of 0.17 s, yet its P90 latency spikes to 16.49 s. This pattern repeats across all inputs — 2-syllable words (median 0.26 s vs. P90 15.95 s) and LLM 2-word phrases (median 0.45 s vs. P90 21.99 s) — indicating severe tail-latency issues irrespective of text length or complexity. Such behavior suggests systemic alignment failures in its dual-decoder mechanism: when the coarse and fine decoders

 doi:10.20944/preprints202508.0654.v1

desynchronize, the model enters extended decoding loops, inflating inference times even on otherwise short prompts. These anomalies render Tacotron2-DDC unsuitable for latency-critical deployments, despite its competitive median responsiveness.

Beyond individual strengths at specific input lengths, the latency rankings also reveal which models maintain stable performance across all text categories. Microsoft Speech stands out as the clearest example of consistency: its median latency remains at 0.06–0.07 s across every input type, from single-syllable words to 18-word sentences, with 90th percentile latency barely exceeding 0.08 s. This indicates virtually no degradation as inputs grow longer, a rare trait among all evaluated models. Combined with its leading audio quality — consistently achieving the lowest incorrect-file rates and word-error rates across both 12-word and 18-word sentences — Microsoft Speech delivers exceptional balance between responsiveness and fidelity. These qualities make it particularly well-suited for general-purpose voice assistants, where reply lengths can vary widely yet both latency and intelligibility must remain predictably high.

FastPitch also displays relatively stable trends — for example, FastPitch's median latency only rises from 0.06 s (1-syllable words) to 0.13 s (18-word sentences). It achieves competitive audio quality, though not as consistently strong as Microsoft Speech or VITS. By contrast, autoregressive models such as Tacotron2-DDC and BarkTTS degrade sharply, with 90th percentile latencies ballooning from 16.49 s and 13.01 s (1-syllable words) to 31.99 s and 78.86 s, respectively, for 2-word and 18-word phrases. These findings highlight how Microsoft Speech — and, to a slightly lesser degree, FastPitch and GlowTTS — provide predictable real-time performance while balancing speed and quality, a critical property for interactive systems where worst-case latency and transcription accuracy both impact user experience.

A notable anomaly is observed in GlowTTS, where latency for longer inputs is sometimes comparable to — or even lower than — that of shorter ones. For instance, GlowTTS records a median latency of 0.10 s for 1-syllable words but achieves 0.09 s for the most common single words and maintains 0.23 s for LLM-generated 2-word phrases. Despite handling more content, its per-word latency does not scale upward as expected, and in some cases, longer phrases are synthesized proportionally faster. This behavior suggests that GlowTTS benefits from parallel flow-based synthesis and stable duration modeling, allowing it to maintain efficient performance even as input complexity increases. In practical applications, this implies GlowTTS can process full-sentence prompts with minimal latency overhead relative to single-word utterances, making it especially effective in scenarios where reply lengths vary dynamically.

In contrast, Vall-E-X and BarkTTS perform markedly worse than other models in terms of latency, with inefficiencies evident across every input length tested. For short single-word inputs, Vall-E-X records a median latency of 2.20 s with 90th percentile latency spiking to 11.72 s, while BarkTTS fares even worse at 5.73 s median and 13.01 s P90. Performance deteriorates further with longer sequences: for LLM-generated 18-word sentences, Vall-E-X reaches 17.79 s median and 45.38 s P90, and BarkTTS records 14.75 s median with an extreme 36.00 s at the 90th percentile.

These values place both models firmly at the bottom of the ranking across all categories, reflecting their computationally heavy autoregressive architectures and lack of optimization for low-latency inference. Their high tail latencies also translate to poor throughput, rendering them impractical for any real-time or interactive voice assistant deployment.

These results indicate that while these models prioritize high-fidelity synthesis and expressive speech generation, their computational demands render them impractical for latency-sensitive applications.

These unusual performance patterns underscore the complexities inherent in text-to-speech benchmarking. Future research will need to focus on stabilizing tail latency, especially in hybrid or multi-stage decoder architectures like Tacotron 2 DDC. Potential approaches include adopting blockwise or speculative decoding methods to limit worst-case sequential steps [50], introducing monotonic alignment constraints or probabilistic duration models as used in Neural HMMs [36], and

 doi:10.20944/preprints202508.0654.v1

redesigning decoder pipelines to separate coarse and fine-grained generation stages with explicit convergence checks. Furthermore, the unexpected speed-ups observed for longer sequences suggest opportunities to exploit batching or hierarchical context windows, where multiple phonemes or words are processed jointly to amortize computation. These findings may also guide training strategies such as curriculum learning (progressively increasing input length) or targeted fine-tuning for structured versus unstructured texts, ensuring models maintain both stability and efficiency across diverse input types. More broadly, this highlights the importance of detailed benchmarking to uncover counterintuitive trends that can inform architectural refinements and future design priorities.

6.3. Cloud-Based TTS Services: GTTS vs. EdgeTTS

Cloud-based TTS services provide an alternative to locally deployed models, offering scalable synthesis capabilities with varying constraints on performance and accessibility. Two cloud-based models evaluated in this study are GTTS (Google Text-to-Speech) and EdgeTTS. These models differ significantly in availability, with GTTS imposing usage limits while EdgeTTS allows unrestricted access. This distinction is crucial for real-time applications, where consistent availability is essential.

When examining responsiveness, GTTS demonstrates moderate yet stable latency behavior across input types. As shown in Table 3, its median latency ranges from 0.43 s on single-word inputs to 0.93 s for 18-word sentences, with 90th-percentile latencies remaining between 0.65 s and 1.39 s. While this performance does not match the ultra-low latency of models such as FastPitch or Microsoft Speech, it remains predictable and comfortably within real-time thresholds across all tested scenarios. EdgeTTS, by contrast, performs notably worse in responsiveness: its median latency spans 1.11 s to 1.78 s, and 90th-percentile latencies rise as high as 2.41 s for longer inputs. The disparity is most evident in shorter phrases, where GTTS sustains sub-0.5 s median latency, whereas EdgeTTS more than doubles that figure.

Audio quality results for 12-word and 18-word sentence synthesis, summarized in Table 4, further highlight differences between the two cloud-based systems. GTTS achieves overall word error rates (WER) of 0.40 for 12-word inputs and 0.39 for 18-word inputs, with median WERs of 16.67 and 10.82, respectively. EdgeTTS records nearly identical overall WER values (0.41 and 0.38) and slightly higher median WERs (16.67 and 11.11). Both models therefore provide comparable transcription fidelity, and both substantially outperform models like NeuralHMM (overall WER 8.48–8.50) or BarkTTS (5.28–4.20), which suffer from high misalignment and token omission rates.

Incorrect transcription rates reinforce this finding: GTTS produces 2.84% incorrect files for 12-word inputs and 4.25% for 18-word inputs, while EdgeTTS reports 3.02% and 4.02%, respectively. These errors primarily arise from phoneme-duration misalignments rather than network instability, indicating that both cloud systems are robust to input length. Taken together, GTTS offers a slight advantage in responsiveness, while EdgeTTS delivers similar audio quality but suffers higher latency — a trade-off that may constrain its use in latency-critical deployments while remaining viable for general-purpose synthesis scenarios.

The primary trade-off between gTTS and EdgeTTS is an availability–versus–quality balance. Looking ahead, major TTS providers already expose configurable performance profiles—e.g., OpenAI's Realtime API for low-latency speech interactions[4] (and its separate, higher-latency Chat Completions audio path)[5], Microsoft Azure's guidance on chunked/streamed synthesis to cut first-byte and finish latency[6], and ElevenLabs' streaming endpoints for real-time audio generation[7].

Commercial tiers and usage limits are likewise segmented: Google Cloud TTS prices Standard/WaveNet/Neural2 voices separately[8], while Amazon Polly differentiates Standard, Neural,

[4] https://platform.openai.com/docs/guides/realtime
[5] https://openai.com/index/introducing-the-realtime-api/
[6] https://learn.microsoft.com/en-us/azure/ai-services/speech-service/how-to-lower-speech-synthesis-latency
[7] https://elevenlabs.io/docs/api-reference/text-to-speech/stream
[8] https://cloud.google.com/text-to-speech/pricing

 doi:10.20944/preprints202508.0654.v1

Long-Form, and Generative voices with distinct quotas and rates[9]. Meanwhile, industry forecasts project the voice user interface market to grow at ∼20% CAGR through 2032, reflecting rising demand for scalable yet stable voice tech—pressure that will drive convergence toward offerings blending reliability with accessibility.[10]

In the end, this benchmark provides a dedicated systematic evaluation of open-source TTS models with a focus on responsiveness—a metric critical to real-time voice assistant performance yet largely absent from prior studies. By standardizing latency and tail-latency measurement across multiple architectures, the framework establishes a reproducible baseline for future research and enables fair comparison with commercial systems. Our findings highlight the strengths of flow-based, parallel as well as in-built CPU based models for low-latency deployment, while exposing persistent bottlenecks in autoregressive approaches. Beyond benchmarking, this work offers practical insights for selecting TTS models in latency-sensitive applications and sets the stage for integrating similar evaluations into end-to-end voice assistant pipelines, ultimately supporting the development of more natural, real-time human–AI interactions.

Author Contributions: Conceptualization, H.P.T.D., A.C., and R.A.P.; methodology, H.P.T.D., A.C. and R.A.P.; software, H.P.T.D.; validation, H.P.T.D.; formal analysis, H.P.T.D.; investigation, H.P.T.D., resources, H.P.T.D.; writing—original draft preparation, H.P.T.D.; writing—review and editing, A.C., R.A.P. and M.L.; visualization, H.P.T.D.; supervision, A.C. and M.L.; project administration, A.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: The dataset utilized for the experiments in this study is publicly accessible on Hugging Face at the following URL: https://huggingface.co/datasets/sebdinh61/benchmarked_text_inputs (accessed on 8 August 2025). It is hosted under the Apache License 2.0 (Apache-2.0), permitting broad reuse with attribution

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hoy, M. Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants. *Medical Reference Services Quarterly* **2018**, *37*, 81–88.
2. Sainath, T.N.; Parada, C. Convolutional Neural Networks for Small-Footprint Keyword Spotting. In Proceedings of the Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH), 2015, pp. 1478–1482.
3. Singh, D.; Kaur, G.; Bansal, A. HOW VOICE ASSISTANTS ARE TAKING OVER OUR LIVES -A REVIEW 1 **2019**.
4. Défossez, A.; Mazaré, L.; Orsini, M.; Royer, A.; Pérez, P.; Jégou, H.; Grave, E.; Zeghidour, N. Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037* **2024**.
5. Srivastava, A.; Rastogi, A.; Rao, A.; Shoeb, A.A.M.; Abid, A.; Fisch, A.; Brown, A.R.; Santoro, A.; Gupta, A.; Garriga-Alonso, A.; et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615* **2022**.
6. Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300* **2020**.
7. Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H.P.D.O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* **2021**.
8. Reddi, V.J.; Cheng, C.; Kanter, D.; Mattson, P.; Schmuelling, G.; Wu, C.J.; Anderson, B.; Breughe, M.; Charlebois, M.; Chou, W.; et al. Mlperf inference benchmark. In Proceedings of the 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2020, pp. 446–459.

[9] https://aws.amazon.com/polly/pricing/
[10] https://www.gminsights.com/industry-analysis/voice-user-interface-market

 doi:10.20944/preprints202508.0654.v1

9. Spangher, L.; Li, T.; Arnold, W.F.; Masiewicki, N.; Dotiwalla, X.; Parusmathi, R.; Grabowski, P.; Ie, E.; Gruhl, D. Project MPG: towards a generalized performance benchmark for LLM capabilities. *arXiv preprint arXiv:2410.22368* **2024**.
10. Banerjee, D.; Singh, P.; Avadhanam, A.; Srivastava, S. Benchmarking LLM powered chatbots: methods and metrics. *arXiv preprint arXiv:2308.04624* **2023**.
11. Bai, G.; Liu, J.; Bu, X.; He, Y.; Liu, J.; Zhou, Z.; Lin, Z.; Su, W.; Ge, T.; Zheng, B.; et al. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. *arXiv preprint arXiv:2402.14762* **2024**.
12. Malode, V.M. Benchmarking public large language model. PhD thesis, Technische Hochschule Ingolstadt, 2024.
13. Jacovi, A.; Wang, A.; Alberti, C.; Tao, C.; Lipovetz, J.; Olszewska, K.; Haas, L.; Liu, M.; Keating, N.; Bloniarz, A.; et al. The FACTS Grounding Leaderboard: Benchmarking LLMs' Ability to Ground Responses to Long-Form Input. *arXiv preprint arXiv:2501.03200* **2025**.
14. Wang, B.; Zou, X.; Lin, G.; Sun, S.; Liu, Z.; Zhang, W.; Liu, Z.; Aw, A.; Chen, N.F. Audiobench: A universal benchmark for audio large language models. *arXiv preprint arXiv:2406.16020* **2024**.
15. Gandhi, S.; Von Platen, P.; Rush, A.M. Esb: A benchmark for multi-domain end-to-end speech recognition. *arXiv preprint arXiv:2210.13352* **2022**.
16. Fang, Y.; Sun, H.; Liu, J.; Zhang, T.; Zhou, Z.; Chen, W.; Xing, X.; Xu, X. S2SBench: A Benchmark for Quantifying Intelligence Degradation in Speech-to-Speech Large Language Models. *arXiv preprint arXiv:2505.14438* **2025**.
17. Chen, Y.; Yue, X.; Zhang, C.; Gao, X.; Tan, R.T.; Li, H. Voicebench: Benchmarking llm-based voice assistants. *arXiv preprint arXiv:2410.17196* **2024**.
18. Alberts, L.; Ellis, B.; Lupu, A.; Foerster, J. CURATe: Benchmarking Personalised Alignment of Conversational AI Assistants. *arXiv preprint arXiv:2410.21159* **2024**.
19. Viswanathan, M.; Viswanathan, M. Measuring speech quality for text-to-speech systems: development and assessment of a modified mean opinion score (MOS) scale. *Computer speech & language* **2005**, *19*, 55–83.
20. mrfakename.; Srivastav, V.; Fourrier, C.; Pouget, L.; Lacombe, Y.; main.; Gandhi, S. Text to Speech Arena. https://huggingface.co/spaces/TTS-AGI/TTS-Arena, 2024.
21. mrfakename.; Srivastav, V.; Fourrier, C.; Pouget, L.; Lacombe, Y.; main.; Gandhi, S.; Passos, A.; Cuenca, P. TTS Arena 2.0: Benchmarking Text-to-Speech Models in the Wild. https://huggingface.co/spaces/TTS-AGI/TTS-Arena-V2, 2025.
22. Picovoice. Picovoice TTS Latency Benchmark. https://github.com/Picovoice/tts-latency-benchmark, 2024. Accessed: 2025-03-09.
23. Artificial Analysis. Text-to-Speech Benchmarking Methodology. https://artificialanalysis.ai/text-to-speech, 2024. Accessed: 2025-03-09.
24. Labelbox. Evaluating Leading Text-to-Speech Models. https://labelbox.com/guides/evaluating-leading-text-to-speech-models/, 2024. Accessed: 2025-03-09.
25. Minixhofer, C.; Klejch, O.; Bell, P. TTSDS-Text-to-Speech Distribution Score. In Proceedings of the 2024 IEEE Spoken Language Technology Workshop (SLT). IEEE, 2024, pp. 766–773.
26. Radford, A.; Kim, J.W.; Xu, T.; Brockman, G.; McLeavey, C.; Sutskever, I. Robust speech recognition via large-scale weak supervision. In Proceedings of the International conference on machine learning. PMLR, 2023, pp. 28492–28518.
27. Griffies, S.M.; Perrie, W.A.; Hull, G. Elements of style for writing scientific journal articles. *Publishing Connect, Elsevier* **2013**, pp. 20–50.
28. Wallwork, A. *English for writing research papers*; Springer, 2016.
29. Carnegie Mellon University Speech Group. The Carnegie Mellon Pronouncing Dictionary (CMUdict). http://www.speech.cs.cmu.edu/cgi-bin/cmudict, 1993–2014. Accessed: 2025-07-23.
30. Day, D.L. CMU Pronouncing Dictionary Python Package. https://pypi.org/project/cmudict/, 2014. Accessed: 2025-07-23.
31. Davies, M. Word frequency data from The Corpus of Contemporary American English (COCA). https://www.wordfrequency.info, 2008. Accessed: 2025-07-23.
32. Davies, M. The iWeb Corpus: 14 billion words of English from the web. https://www.english-corpora.org/iweb/, 2018. Accessed: 2025-07-23.
33. AI, S. Bark: A Transformer-Based Text-to-Audio Model, 2023. Accessed: 2025-03-10.

 doi:10.20944/preprints202508.0654.v1

34. Défossez, A.; Copet, J.; Synnaeve, G.; Adi, Y. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438* **2022**.
35. Zhang, Z.; Zhou, L.; Wang, C.; Chen, S.; Wu, Y.; Liu, S.; Chen, Z.; Liu, Y.; Wang, H.; Li, J.; et al. Speak foreign languages with your own voice: Cross-lingual neural codec language modeling. *arXiv preprint arXiv:2303.03926* **2023**.
36. Mehta, S.; Székely, É.; Beskow, J.; Henter, G.E. Neural HMMs are all you need (for high-quality attention-free TTS). In Proceedings of the ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2022, pp. 7457–7461.
37. Su, J.; Jin, Z.; Finkelstein, A. HiFi-GAN-2: Studio-quality speech enhancement via generative adversarial networks conditioned on acoustic features. In Proceedings of the 2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA). IEEE, 2021, pp. 166–170.
38. Ito, K.; Johnson, L. The LJ Speech Dataset. https://keithito.com/LJ-Speech-Dataset/, 2017.
39. Mehta, S.; Kirkland, A.; Lameris, H.; Beskow, J.; Székely, É.; Henter, G.E. OverFlow: Putting flows on top of neural transducers for better TTS. *arXiv preprint arXiv:2211.06892* **2022**.
40. rany2. edge-tts: Microsoft Edge Text-to-Speech Library. https://pypi.org/project/edge-tts/, 2021. Accessed: 6 August 2025.
41. Łańcucki, A. Fastpitch: Parallel text-to-speech with pitch prediction. In Proceedings of the ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021, pp. 6588–6592.
42. Kim, J.; Kong, J.; Son, J. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In Proceedings of the International Conference on Machine Learning. PMLR, 2021, pp. 5530–5540.
43. Kim, J.; Kim, S.; Kong, J.; Yoon, S. Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems* **2020**, *33*, 8067–8077.
44. Yang, G.; Yang, S.; Liu, K.; Fang, P.; Chen, W.; Xie, L. Multi-band melgan: Faster waveform generation for high-quality text-to-speech. In Proceedings of the 2021 IEEE Spoken Language Technology Workshop (SLT). IEEE, 2021, pp. 492–498.
45. Battenberg, E.; Mariooryad, S.; Stanton, D.; Skerry-Ryan, R.; Shannon, M.; Kao, D.; Bagby, T. Effective use of variational embedding capacity in expressive end-to-end speech synthesis. *arXiv preprint arXiv:1906.03402* **2019**.
46. King, S.; Karaiskos, V. The Blizzard Challenge 2013. In Proceedings of the Proc. Blizzard Challenge Workshop, 2013.
47. Shen, J.; Pang, R.; Weiss, R.J.; Schuster, M.; Jaitly, N.; Yang, Z.; Chen, Z.; Zhang, Y.; Wang, Y.; Skerrv-Ryan, R.; et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In Proceedings of the 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2018, pp. 4779–4783.
48. Durette, P.N. gTTS: Documentation, 2014. Accessed: March 10, 2025.
49. Corporation, M. Speech API 5.4 Documentation, 2009. Accessed: 2025-03-10.
50. Stern, M.; Shazeer, N.; Uszkoreit, J. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems* **2018**, *31*.