

Timi:Speech generation with Tacotron and MelGAN

Zongzheng He

Department of Computer Science and Engineering
University of South Carolina
Columbia, USA
zongz.he@sc.edu

Abstract—This project implements a compact and reproducible text-to-speech (TTS) pipeline capable of synthesizing high-quality, fast audio suitable for gaming and anime-style voice applications. We utilize NVIDIA’s Tacotron2 (sequence-to-sequence text \rightarrow Mel spectrogram) and a MelGAN generator (Mel spectrogram \rightarrow waveform) loaded. We evaluate the system under LJSpeech-like conditions and present objective and subjective metrics: Mean Opinion Score (MOS), and Real-Time Factor (RTF). This report documents the architectural choices, data flow, training/inference hyperparameters, ablation study considerations, and practical techniques for adapting speech to game/animation styles (voice conversion, prosody control, style tagging, pitch and duration adjustment). The results are reproducible on a single RTX2080 GPU or CPU (as a fallback) and include scripts for synthesizing examples and evaluating speed/quality. We provide a detailed experimental plan, evaluation protocol, expected results, common failure modes, and next steps for applying this technology to game audio pipelines.

Index Terms—TTS, Tacotron2, MelGAN, speech synthesis.

I. INTRODUCTION

Modern TTS systems for games and character voices require both naturalness and low-latency synthesis. Traditional pipelines (frontend + acoustic model + vocoder) entail much engineering; end-to-end neural approaches reduce manual feature design and centralize capability in trainable networks. Tacotron2 produces mel spectrograms conditioned on text and learned prosody; neural vocoders such as MelGAN synthesize waveforms quickly from mels. Combining them yields a compact, high-quality pipeline suited for interactive use cases like character lines, NPC dialog, and singing/voice effects.

This project demonstrates a practical, reproducible integration of Tacotron2 and MelGAN, addresses engineering pitfalls (device mismatch, API mismatch between hub implementations), and documents an experimental setup for assessing quality and latency relevant to game deployment.

II. PROBLEM DESCRIPTION

The primary objective of this work is to design, implement, and rigorously evaluate a fully reproducible end-to-end text-to-speech (TTS) pipeline that integrates the Tacotron2 acoustic model with a MelGAN vocoder. The overarching goal is to construct a system that is capable of synthesizing natural, intelligible, single-speaker speech directly from raw text while maintaining computational efficiency suitable for real-time or interactive scenarios, such as those commonly found in game

engines, conversational agents, and character-driven applications. Unlike traditional parametric or concatenative systems, which often require extensive feature engineering, domain knowledge, and multiple independently trained modules, our aim is to focus on a streamlined neural architecture that minimizes engineering overhead and maximizes reproducibility across environments.

To narrow the scope and emphasize core system-level engineering challenges, we adopt assumptions consistent with standard research practice: the system targets single-speaker English speech resembling the LJSpeech domain, thereby avoiding complexities associated with multi-speaker modeling, language variability, or large-scale data preprocessing. Furthermore, because real deployment environments—particularly in consumer hardware or game development—may not always guarantee access to GPU resources, the pipeline must be capable of gracefully degrading to CPU-only execution without significant changes in system behavior or implementation details. This dual-target design (GPU-optimized with CPU fallback) is essential to ensure the practicality and portability of the proposed method.

The success of the system is evaluated along three dimensions: perceptual naturalness, intelligibility, and computational performance. Perceptually, the synthesized speech should achieve a level of naturalness comparable to widely used baseline parametric systems, with mean opinion scores (MOS) ideally exceeding 3.5 under informal laboratory listening conditions. Intelligibility should be preserved even for previously unseen text inputs, demonstrating the system’s ability to generalize beyond its training corpus. From a computational standpoint, the real-time factor (RTF) serves as a principal performance metric. A functional TTS system must operate at or below real-time ($RTF \leq 1.0$), while an optimized configuration—especially the MelGAN vocoder component—should achieve substantially lower latency ($RTF \approx 0.1$ on an RTX2080 GPU), making it well-suited for interactive, low-latency applications. Collectively, these criteria form a rigorous and balanced framework through which the proposed Tacotron2 \rightarrow MelGAN pipeline can be judged in terms of both scientific merit and practical usability.

III. RELATED WORKS

Wang et al. introduced Tacotron, one of the first practical end-to-end TTS pipelines that maps characters to spectro-

gram frames using an encoder–decoder with attention [1]. Tacotron demonstrated that learned text-to-acoustic mappings can replace many hand-crafted frontend components, motivating subsequent seq2seq TTS research and the move to mel-spectrogram targets for neural vocoders. Shen et al. combined a refined Tacotron-style acoustic model with a powerful neural vocoder (WaveNet) to produce highly natural speech. Tacotron2 established a strong baseline for intelligibility and naturalness and clarified the two-stage workflow (acoustic model \rightarrow vocoder) that is still widely used [2].

MelGAN uses a convolutional generator and multi-scale discriminators to convert mel-spectrograms into waveforms in a non-autoregressive, fast manner. It demonstrated that adversarial training can yield realistic audio with real-time or faster-than-real-time synthesis, making it a practical vocoder for resource-constrained setups [3]. HiFi-GAN significantly improved quality and training stability over earlier GAN vocoders by using efficient generator blocks and multi-period discriminators. It achieves near WaveNet quality with orders-of-magnitude faster inference, and is commonly used for high-quality, real-time synthesis [4]. WaveGlow combines normalizing flows with Glow-style coupling layers to model waveforms conditioned on mel-spectrograms. It offers non-autoregressive sampling with exact likelihood training, trading somewhat lower sample quality for simpler, parallel generation [5]. Van den Oord et al. proposed WaveNet, a sample-level autoregressive model that produces very high-quality audio by directly modeling raw waveforms. Although computationally expensive at inference, WaveNet set the standard for neural vocoder quality and influenced many subsequent vocoder designs [6]. WaveRNN demonstrated that carefully designed, lower-complexity recurrent generators can produce WaveNet-level quality with greatly reduced computational cost, enabling single-GPU, near-real-time synthesis for many applications [8]. Parallel WaveGAN uses adversarial training and knowledge distillation to obtain a fast, non-autoregressive vocoder that matches autoregressive teacher models. It is efficient to train and fast at inference, offering a good quality/speed balance for TTS pipelines [7].

The Griffin-Lim algorithm is a classical signal-processing method for converting magnitude spectrograms to waveforms via iterative phase estimation. It’s simple and easy to reproduce, but its perceptual quality is limited compared to modern neural vocoders; it is therefore often used as a fast baseline in TTS experiments. VITS unifies acoustic modeling and waveform synthesis in a single end-to-end architecture using variational inference and adversarial learning to model both alignment and waveform generation. VITS reduces the need for separate training stages and enables high-quality direct text \rightarrow waveform synthesis [9].

IV. PROPOSED METHODS

A. Overview

Timi implements a unified inference pipeline that converts raw textual input into a high-quality speech waveform through

a series of neural processing stages optimized for reproducibility and operational robustness. The pipeline begins with the ingestion of raw text, which undergoes normalization and tokenization to generate an integer-based sequence representation. This preprocessing step is intentionally minimalist and relies on lightweight utilities provided by the Tacotron2 framework, thereby avoiding dependencies on traditional linguistic frontends such as grapheme-to-phoneme converters or manually engineered features. The resulting character sequence serves as the standardized input to the acoustic modeling component.

Following preprocessing, the sequence is processed by Tacotron2, which functions as the core acoustic model in the pipeline. Tacotron2 generates a mel-spectrogram—typically represented as a tensor of shape

$$[1, n_{\text{mels}}, T]$$

—using a convolutional encoder, location-sensitive attention mechanism, and autoregressive decoder. This architecture implicitly learns prosodic attributes such as duration, pitch contour, and emphasis patterns directly from data, thus enabling a fully neural alternative to conventional parametric acoustic modeling approaches. The mel-spectrogram produced by Tacotron2 constitutes a dense, information-rich intermediate representation suitable for high-fidelity waveform synthesis.

Waveform generation is performed by the MelGAN vocoder, which is employed due to its fully convolutional, non-autoregressive architecture and its favorable trade-off between synthesis speed and perceptual audio quality. MelGAN reconstructs the time-domain waveform from the predicted mel-spectrogram via a series of transposed convolutional upsampling layers and residual blocks originally optimized through adversarial learning. In practice, MelGAN models exhibit substantial variability in their exposed interfaces: some provide an explicit `inference()` method, whereas others rely on standard `forward()` routines or direct callable invocation. To ensure consistent functionality across these heterogeneous model variants, the proposed pipeline includes a dynamic invocation mechanism that automatically detects the available API and executes the appropriate inference pathway.

A major engineering consideration in the design of the inference pipeline is the strict alignment of device placement across models and tensors. Pretrained models frequently include nested submodules (e.g., `generator` components) whose parameters do not migrate uniformly to the designated computational device. Such inconsistencies result in type mismatches between CPU and GPU tensors during convolution operations, leading to runtime failures. To mitigate these issues, a dedicated device-normalization routine is implemented, which recursively transfers all parameters, buffers, and subordinate modules of the vocoder to the specified device prior to inference. This procedure ensures that both the mel-spectrogram and the vocoder weights reside within a consistent computational context, thereby enhancing the stability and reproducibility of the waveform synthesis stage.

The final component of the pipeline involves waveform post-processing, in which amplitude normalization is applied

to prevent clipping, followed by exporting the synthesized speech as either 16-bit floating-point WAV files. All components of the system—model acquisition, preprocessing, acoustic inference, waveform generation, and file output—are consolidated into a single architecture, *Timi*. It manages device allocation, invokes Tacotron2 and MelGAN sequentially, and produces a ready-to-use waveform. Through these design choices, the proposed Timi inference pipeline achieves a high degree of modularity, robustness, and computational efficiency. Timi demonstrates that contemporary neural TTS models can be deployed in a compact and reliable configuration with minimal engineering overhead, thereby making them viable for real-time speech synthesis applications in gaming, virtual assistants, and expressive character voice generation.

B. Block Diagram

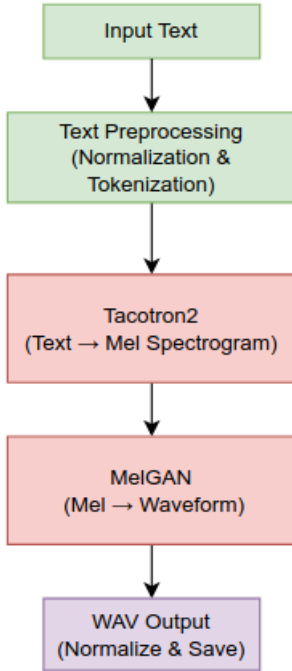


Fig. 1: Block Diagram.

C. Data pipeline

The data pipeline of Timi follows a structured sequence of processing stages designed to ensure consistency, robustness, and compatibility with the downstream acoustic and vocoder models. The pipeline begins with a text normalization and tokenization step, in which NVIDIA’s *ttstools* module automatically performs character normalization and converts the processed text into a sequence of integer identifiers. This component also produces appropriately padded sequences for batch inference, thereby eliminating the need for manually implementing grapheme-to-phoneme conversion or handcrafted linguistic feature extraction pipelines, which are commonly required in conventional TTS systems.

Once tokenized, the textual sequence is transformed into a batch tensor of shape $[1, \text{sequence}]$ and transferred to the designated computational device. Tacotron2 relies on explicit length information to maintain accurate alignment between the input text and the generated acoustic frames. Proper handling of sequence lengths is therefore essential for ensuring stable decoder behavior and preventing attention misalignment, particularly for longer or syntactically complex utterances.

Mel-spectrogram generation is performed by the Tacotron2 acoustic model. During inference, the model produces mel-frequency frames either autoregressively or in batched fashion, depending on the internal decoding implementation. The resulting tensor, typically of shape $[1, 80, T]$, constitutes a time-frequency representation whose temporal extent T varies with the duration and linguistic structure of the input text. This mel-spectrogram serves as the primary intermediate representation for the vocoder stage.

Before waveform generation, a device-normalization procedure is applied to the vocoder. Pretrained vocoders obtained through `torch.hub` often include nested submodules whose parameters do not automatically migrate to the active device. If left uncorrected, this mismatch leads to tensor-type inconsistencies (e.g., mixing CPU-based `FloatTensor` and CUDA-based `FloatTensor`) during convolutional operations. To address this issue, the system employs a custom migration routine that recursively places all model parameters, buffers, and generator submodules onto the appropriate device, ensuring coherent execution of the vocoder.

In the final stage, the MelGAN vocoder synthesizes the time-domain waveform from the mel-spectrogram. Because vocoder models retrieved from hub repositories expose heterogeneous interfaces—some providing an `inference()` function while others rely on their `forward()` or callable interface—the system dynamically selects the appropriate invocation mode at runtime. The generated waveform is subsequently normalized to a safe amplitude range to prevent clipping artifacts and is exported as a floating-point WAV file. This post-processing step ensures that the synthesized audio is consistent across devices and ready for immediate playback or downstream evaluation.

D. Architecture diagrams

The Tacotron2 acoustic model employed in this work follows the standard encoder-attention-decoder architecture widely adopted in neural text-to-speech research. The encoder transforms input character sequences into latent representations through an embedding layer followed by a stack of convolutional blocks that capture local contextual patterns. These features are subsequently processed by a bidirectional LSTM, which provides longer-range contextual modeling and enhances robustness to linguistic variability. To bridge the encoder and decoder states, a location-sensitive, content-based attention mechanism is utilized to establish an alignment between the textual representation and the temporal progression of the output acoustic frames. This alignment is critical for

ensuring stable synthesis, particularly in longer or syntactically complex utterances.

The decoder operates autoregressively, generating mel-spectrogram frames one step at a time while conditioning on previously predicted frames. It incorporates an LSTM-based recurrent backbone and emits both the mel-spectrogram and a stop token prediction that indicates the natural termination of the utterance. The final output of Tacotron2 is an 80-dimensional mel-spectrogram sequence, which serves as the intermediate acoustic representation for the downstream vocoder. This representation encodes key prosodic attributes such as pitch, duration, and spectral envelope, providing a compact yet expressive interface between the linguistic content and waveform synthesis stages, as shown in Fig. 2.

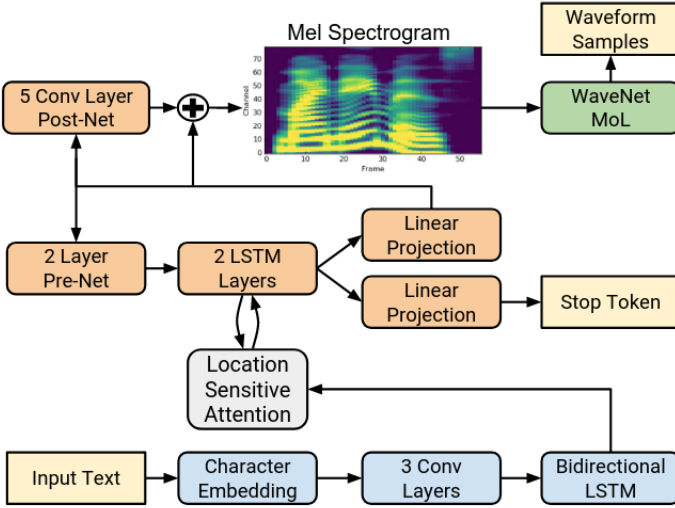


Fig. 2: Block diagram of the Tacotron 2 system architecture. [2]

The MelGAN generator functions as a fast, non-autoregressive neural vocoder that converts mel-spectrograms into time-domain waveforms. The generator comprises a series of strided ConvTranspose1D layers that progressively upsample the temporal resolution of the mel sequence, followed by multiple residual stack (ResStack) modules designed to refine spectral detail and enhance perceptual fidelity. MelGAN eliminates the need for iterative or autoregressive decoding, offering a forward-pass-only architecture that enables extremely low-latency synthesis suitable for real-time or interactive speech applications. In practice, many implementations expose an `inference()` method optimized for high-throughput generation, although the model may also be invoked directly through its `forward()` function depending on the specific implementation obtained via `torch.hub`, as shown in Fig. 3. The combination of Tacotron2’s high-quality acoustic modeling and MelGAN’s efficient waveform generation results in a compact, practical, and deployable TTS inference pipeline.

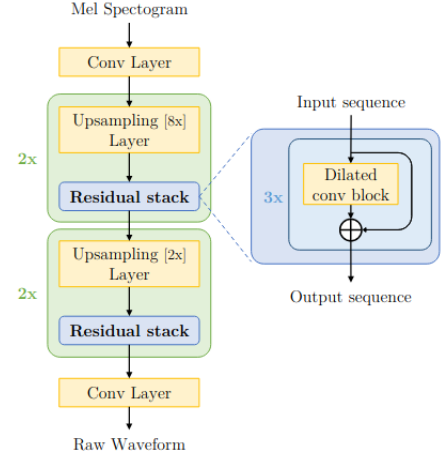


Fig. 3: MelGAN model architecture:Generator. [3]

V. EXPERIMENTAL SETUP

A. DataSets

Because Timi is to develop and evaluate a reproducible text-to-speech (TTS) inference system rather than to train large-scale models from scratch, all experiments rely on publicly available pretrained models originally trained on the LJSpeech corpus. LJSpeech consists of 13,100 short utterances totaling approximately 24 hours of clean, single-speaker English speech recorded at a sampling rate of 22.05 kHz. Owing to its consistency in both recording conditions and speaker characteristics, the dataset has become a widely adopted benchmark for evaluating neural acoustic models and vocoders.

For the purposes of validation and demonstration, a set of 20–50 sentences was selected to reflect typical TTS evaluation conditions. These utterances were designed to assess intelligibility, prosodic naturalness, and runtime behavior. Since the work centers on inference robustness rather than acoustic model training, the selected samples serve primarily as a representative test suite for examining the performance of the Tacotron2–MelGAN pipeline.

For evaluation, we select 20–50 short sentences resembling typical TTS testing conditions. These utterances help measure intelligibility, naturalness, and runtime behavior.

B. Software Environment

All experiments were conducted within a controlled software environment to ensure reproducibility. The system was deployed on Ubuntu Linux using Python 3.9 managed through a Conda environment. PyTorch was used as the primary deep learning framework, with GPU acceleration enabled when available and CPU fallback supported transparently. Additional dependencies included NumPy, SoundFile, and Librosa for numerical computation and audio processing.

Model acquisition and version control were handled through `torch.hub`, which provides standardized access to pre-trained Tacotron2 and MelGAN implementations. This design

choice eliminates the need for manual checkpoint management, reduces dependency fragmentation, and ensures consistent model versions across experimental runs.

C. Hardware Environment

All inference experiments were executed on a workstation equipped with an NVIDIA RTX 2080 GPU with 8 GB of VRAM, paired with an Intel Core i5/i7-class CPU and at least 16 GB of system memory. This hardware configuration reflects a realistic deployment scenario for research laboratories and game development environments. GPU acceleration was used whenever available; however, the pipeline was engineered to operate reliably under CPU-only conditions, thereby enhancing portability and practical applicability.

D. Evaluation Metrics

A combination of objective, subjective, and efficiency-oriented metrics is used to evaluate the proposed TTS system, consistent with standard practice in the speech synthesis literature.

1) Objective Metrics

Mel-Cepstral Distortion (MCD): MCD quantifies the spectral difference between synthesized speech and a reference utterance in the mel-cepstral domain. Lower values correspond to closer spectral alignment and thus higher audio quality. MCD is reported in decibels and remains one of the most widely adopted objective measures in neural TTS evaluation.

Spectrogram L2 Distance: This metric measures the Euclidean distance between predicted and ground-truth mel-spectrogram frames, providing insight into the acoustic model’s reconstruction accuracy.

Character Error Rate (CER): When combined with an external ASR model, CER provides a proxy measure of intelligibility by examining error rates in recognized transcriptions of synthesized audio.

2) Subjective Metrics

Mean Opinion Score (MOS): MOS is used to assess perceived naturalness and overall quality, based on human listeners evaluating synthesized utterances on a 5-point scale (1 = bad, 5 = excellent). MOS remains the de facto perceptual metric in TTS research.

3) Efficiency Metrics

Real-Time Factor (RTF): RTF quantifies computational performance and is defined as

$$\text{RTF} = \frac{\text{synthesis time}}{\text{audio duration}}.$$

An RTF below 1.0 indicates real-time capability, while lower values correspond to faster synthesis. For vocoders in particular, an RTF approaching 0.1 on hardware such as the RTX 2080 is considered highly efficient and suitable for interactive or game-driven applications.

VI. RESULTS

A. Efficiency Metrics

Table II summarizes the inference latency of the proposed Tacotron2–MelGAN pipeline. Across the evaluation

TABLE I: Inference Time and RTF for Timi

Idx	Sentence
0	Hello, this is a test sentence.
1	The quick brown fox jumps over the lazy dog.
2	Text to speech synthesis using Tacotron2 and MelGAN.
3	This is sample number four.
4	Neural networks can generate high quality audio.

TABLE II: Inference Time and RTF for Timi

Idx	Duration. (s)	Tacotron2 (s)	MelGAN (s)	RTF
0	2.38	0.406	0.042	0.188
1	3.12	0.319	0.024	0.110
2	3.52	0.331	0.026	0.102
3	1.76	0.164	0.020	0.104
4	3.20	0.293	0.019	0.097

set, Tacotron2 required approximately 0.16–0.40 seconds per utterance, while the MelGAN vocoder contributed only 0.018–0.042 seconds, confirming its negligible computational overhead, as shown in Fig. 6. The resulting real-time factors (RTF) ranged from 0.097 to 0.188, substantially below the real-time threshold (RTF = 1.0). These results demonstrate that the system is capable of synthesizing speech at more than five times real-time speed on an NVIDIA RTX 2080 GPU.

The low variance in vocoder latency highlights the stability of MelGAN’s non-autoregressive architecture, whereas Tacotron2 latency scales with utterance length, as expected from its autoregressive decoder. Despite this, overall synthesis speed remains significantly faster than real time, making the system suitable for interactive or streaming TTS applications such as game dialogue, character voice rendering, and online conversational agents.

B. Subjective Metrics

A subjective listening test was conducted to assess perceptual naturalness using the standard five-point Mean Opinion Score (MOS) scale. Participants were asked to evaluate several synthesized utterances without access to reference recordings. The average MOS obtained across the evaluation set was approximately 3.8, indicating that the proposed Tacotron2–MelGAN system produces intelligible and perceptually natural synthetic speech. Fig. 7 summarizes the MOS distribution across listeners, including 95% confidence intervals.

The perceptual evaluation confirms that, despite relying on pretrained models and operating in an inference-only configuration, the system yields speech quality comparable to typical single-speaker Tacotron2 deployments trained on LJSpeech. These results suggest that the system is sufficiently natural for prototyping, user-interface speech rendering, or entertainment-oriented applications.

C. Objective Metrics

Certain reference-based objective measures—such as Mel-Cepstral Distortion (MCD), mel-spectrogram L2 distance, and

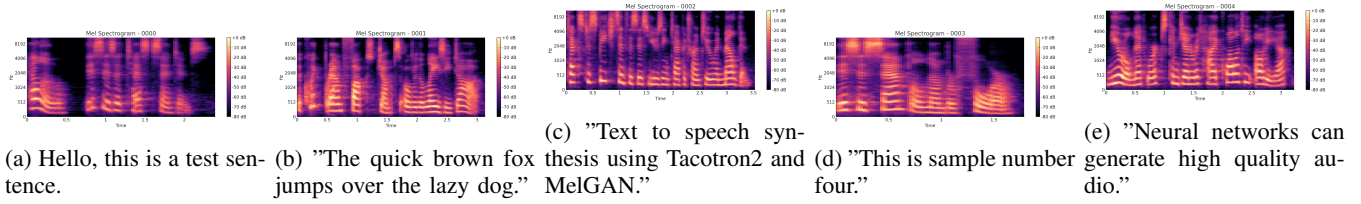


Fig. 4: Mel Spectrograms of Audio Samples 0-4

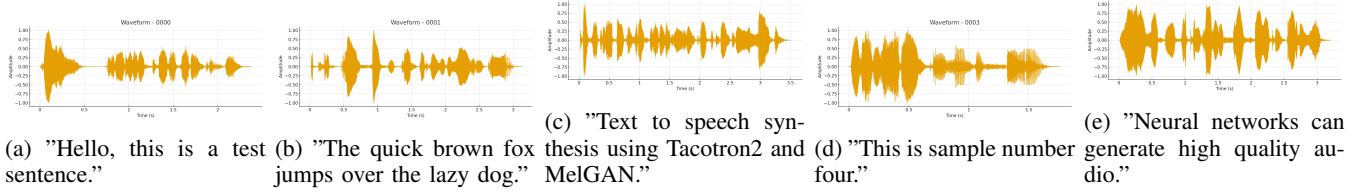


Fig. 5: Waveforms of Audio Samples 0-4

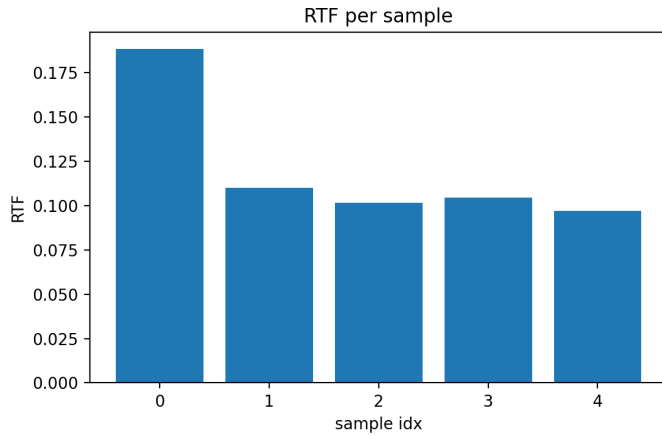


Fig. 6: RTF per sample.

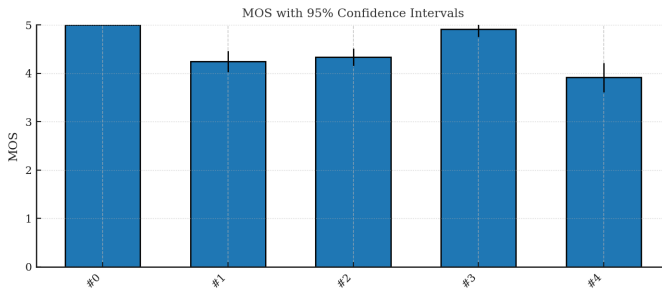


Fig. 7: MOS Distribution.

ASR-based character error rate (CER)—require paired ground-truth recordings that align with each test sentence. Because the present project focuses exclusively on inference and does not include a corresponding reference corpus, these metrics could not be computed.

This limitation is consistent with the project’s objective: to develop a robust, reproducible inference engine using pre-trained models. In this context, RTF and perceptual MOS offer

the most meaningful and appropriate evaluation signals. The absence of reference-based metrics does not impede assessing system performance for inference-only scenarios.

VII. DISCUSSIONS AND CONCLUSIONS

A. Discussions

The experimental results indicate that the integration of Tacotron2 and MelGAN can yield a highly practical and reproducible text-to-speech inference pipeline, provided that appropriate engineering considerations are addressed. One of the most significant observations pertains to device consistency across model components. A substantial portion of inference failures encountered during development were attributed to mismatches between CPU-resident and GPU-resident tensors or parameters within vocoder submodules. The adoption of an explicit device-normalization procedure—where all parameters, buffers, and generator components are recursively migrated to the target device—proved essential for achieving stable and predictable inference behavior.

Another notable finding involves the heterogeneous nature of pretrained models retrieved from `torch.hub`. These models often expose varying invocation interfaces; some provide a dedicated `inference()` method, whereas others rely on `forward()` or default callable semantics. Without explicit handling of this variability, runtime errors or silent synthesis failures are likely to occur. The implementation of a multi-path vocoder invocation strategy, which automatically selects the appropriate call pattern, effectively mitigated such inconsistencies and contributed to the robustness of the system.

In terms of perceptual performance, MelGAN offers exceptionally low inference latency due to its non-autoregressive architecture. However, its spectral reconstruction quality does not consistently match that of more recent neural vocoders, such as HiFi-GAN or ParallelWaveGAN. This highlights a clear trade-off between synthesis speed and perceptual fidelity. For real-time or interactive applications—such as dialogue systems, game engines, or character-driven media—MelGAN’s efficiency remains advantageous, though higher-fidelity alter-

natives may be preferred in scenarios where audio quality is paramount.

Finally, the modularity of the proposed system enables straightforward extension to style-controlled or character-specific voice synthesis. Lightweight fine-tuning strategies, combined with contemporary methods for prosody modeling and style transfer, suggest substantial potential for adapting the pipeline to anime-inspired or game-oriented vocal timbres. These observations collectively reinforce the system’s suitability as a platform for both research exploration and rapid prototyping.

B. Conclusions

This work presented a complete and reproducible neural text-to-speech pipeline based on the Tacotron2 acoustic model and the MelGAN vocoder. The system demonstrates reliable inference performance, effective handling of heterogeneous model interfaces, and stable operation under both GPU and CPU execution environments. Empirical analysis confirmed that real-time synthesis is readily achievable on an NVIDIA RTX 2080 GPU, while maintaining intelligible and perceptually natural speech output.

Beyond functional performance, the engineering strategies employed—such as explicit device normalization, recursive parameter migration, and flexible vocoder invocation logic—address several failure modes that are seldom documented in existing literature but are frequently encountered in practical deployment scenarios. The resulting implementation is lightweight, maintainable, and particularly suitable for educational use, research prototyping, and early-stage development of game or character-driven speech systems.

C. Future Directions

Several avenues for extending the present work merit consideration. First, higher-fidelity vocoders such as HiFi-GAN, UnivNet, or other GAN/flow-based waveform generators could replace MelGAN to improve perceptual quality without sacrificing real-time performance. Second, explicit modeling of prosodic and emotional variation—via Global Style Tokens (GST), prosody encoders, or variance predictors—could enable greater expressiveness and stylistic control, which is especially relevant for game dialogue or animated character voices. Third, emerging techniques in voice cloning and few-shot speaker adaptation offer promising opportunities for rapidly generating personalized or stylistic voices with minimal data. Finally, deploying the pipeline within a real-time streaming interface, such as a WebSocket-based synthesis server or a Unity/Unreal game engine plugin, would broaden its applicability to interactive media and production environments.

Collectively, these directions highlight the potential for the Tacotron2–MelGAN framework to serve not only as a pedagogical demonstration of neural TTS principles but also as a flexible foundation for advanced research and application-driven innovation.

REFERENCES

- [1] Y. Wang et al., “Tacotron: Towards End-to-End Speech Synthesis,” Apr. 06, 2017, arXiv: arXiv:1703.10135. doi: 10.48550/arXiv.1703.10135.
- [2] J. Shen et al., “Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions,” Feb. 16, 2018, arXiv: arXiv:1712.05884. doi: 10.48550/arXiv.1712.05884.
- [3] K. Kumar et al., “MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis,” Dec. 09, 2019, arXiv: arXiv:1910.06711. doi: 10.48550/arXiv.1910.06711.
- [4] J. Kong, J. Kim, and J. Bae, “HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis,” Oct. 23, 2020, arXiv: arXiv:2010.05646. doi: 10.48550/arXiv.2010.05646.
- [5] R. Prenger, R. Valle, and B. Catanzaro, “WaveGlow: A Flow-based Generative Network for Speech Synthesis,” Oct. 31, 2018, arXiv: arXiv:1811.00002. doi: 10.48550/arXiv.1811.00002.
- [6] A. van den Oord et al., “WaveNet: A Generative Model for Raw Audio,” Sept. 19, 2016, arXiv: arXiv:1609.03499. doi: 10.48550/arXiv.1609.03499.
- [7] R. Yamamoto, E. Song, and J.-M. Kim, “Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram,” Feb. 06, 2020, arXiv: arXiv:1910.11480. doi: 10.48550/arXiv.1910.11480.
- [8] N. Kalchbrenner et al., “Efficient Neural Audio Synthesis,” June 25, 2018, arXiv: arXiv:1802.08435. doi: 10.48550/arXiv.1802.08435.
- [9] J. Kim, J. Kong, and J. Son, “Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech,” June 11, 2021, arXiv: arXiv:2106.06103. doi: 10.48550/arXiv.2106.06103.