Real-Time Text-to-Speech (TTS) System

Overview

This project implements a **real-time Text-to-Speech (TTS) system** that accepts text input via streaming and produces **low-latency, natural-sounding** audio output. The system is built without third-party TTS APIs and includes a custom or fine-tuned TTS model.

Features

Real-Time Input and Output: Accepts

- text via WebSocket and streams generated speech.
- **Custom TTS Model:** Built or fine-tuned using Tacotron2, FastSpeech, or Transformer-TTS.
- **Natural Sounding Speech:** Optimized for realistic intonation and prosody.
- **Multi-Language Support (Optional):** English and other supported languages like Hindi.
- **Low Latency:** Generates and streams audio with minimal delay.
- **Deployable with Flask & WebSockets.**

System Architecture

The system consists of the following components:

1. **Frontend (HTML/JavaScript):**

   - Sends text input to the backend via WebSocket.
   - Receives real-time audio and plays it.
2. **Backend (Flask with WebSocket):**
   - Accepts text input and processes it in real-time.
   - Uses a fine-tuned TTS model to generate speech.
   - Streams audio back to the frontend.
3. **TTS Model:**
   - A pre-trained or fine-tuned model for speech synthesis.
   - Optimized for low latency and natural speech.

Model Training & Fine-Tuning

1. Dataset Preparation

- Use **LJSpeech** (for English) and **IndicTTS** (for Hindi/other languages).
- Preprocess text and alignments (phoneme conversion, normalization).

2. Training the TTS Model

- Use a deep learning model like Tacotron2 or FastSpeech2.
- Train using datasets with **Mel-spectrogram generation and WaveNet-based vocoder.**
- Optimize for **naturalness, clarity, and low-latency inference.**

3. Fine-Tuning

- Fine-tune using domain-specific datasets.
- Apply Transfer Learning if extending to multiple languages.
- Optimize **inference speed** for real-time generation.

Installation Guide

Prerequisites

Ensure you have the following installed:

- Python 3.8+
- PyTorch
- Flask & Flask-SocketIO
- WebSockets
- NumPy, SoundFile, and other dependencies

Setup

1. Clone the repository:
2. git clone https://github.com/yourusername/real-time-tts.git

cd real-time-tts

3. Create a virtual environment and install dependencies:
4. python -m venv venv
5. source venv/bin/activate # On Windows use: venv\Scripts\activate

pip install -r requirements.txt

6. Download or train the TTS model:

python train_tts.py # If training from scratch

7. Start the backend server:

python app.py

8. Open the frontend in a browser and start interacting.

Deployment

Local Deployment

Run the Flask server and access via localhost:

python app.py

Deploying on Cloud (Optional)

- Use **AWS EC2 / GCP / Azure** for cloud deployment.
- Use **Docker** for containerized deployment.
- Use **Nginx or Gunicorn** to serve efficiently.

Performance Benchmarks (Optional)

- **Latency Analysis:** Measure text-to-audio time.
- **Speech Naturalness Evaluation:** Subjective ratings or MCD scores.

Repository Link

GitHub Repository ---
https://github.com/GaurangMhatre31/real-time-tts

Contributors

- **Gaurang Mhatre** (AI/ML Developer)