**Task: Train a Non-Hindi Language Model using the AI4Bharat Data / Svarah Corpus with VITS**

**Project Overview:**

The goal is to train a robust speech synthesis model using the VITS architecture, specifically focusing on a non-Hindi language from the AI4Bharat data corpus. This project challenges candidates to leverage advanced machine learning techniques to create a model capable of generating high-quality audio from text.

**Objective:**

Train a speech synthesis model in a language other than Hindi by utilizing the AI4Bharat data corpus, avoiding the use of news datasets. The model should be capable of streaming WAV audio to FLAC audio using WebSockets.

**Overview**

This project sets up a WebSocket server that allows users to convert WAV audio files to FLAC format. The server listens for incoming WAV audio data, processes it, and sends back the converted FLAC data. It is built using Python’s websockets library for real-time communication and soundfile for audio file handling.

**Requirements**

Before running the server, ensure you have the following libraries installed:

• websockets

• soundfile

• asyncio

• nest\_asyncio (for compatibility in environments like Google Colab)

You can install these dependencies using the following command:



**Setup Instructions**

1. Clone the Repository (if applicable):

If this code is part of a repository, clone it to your local machine or open it in Google Colab.

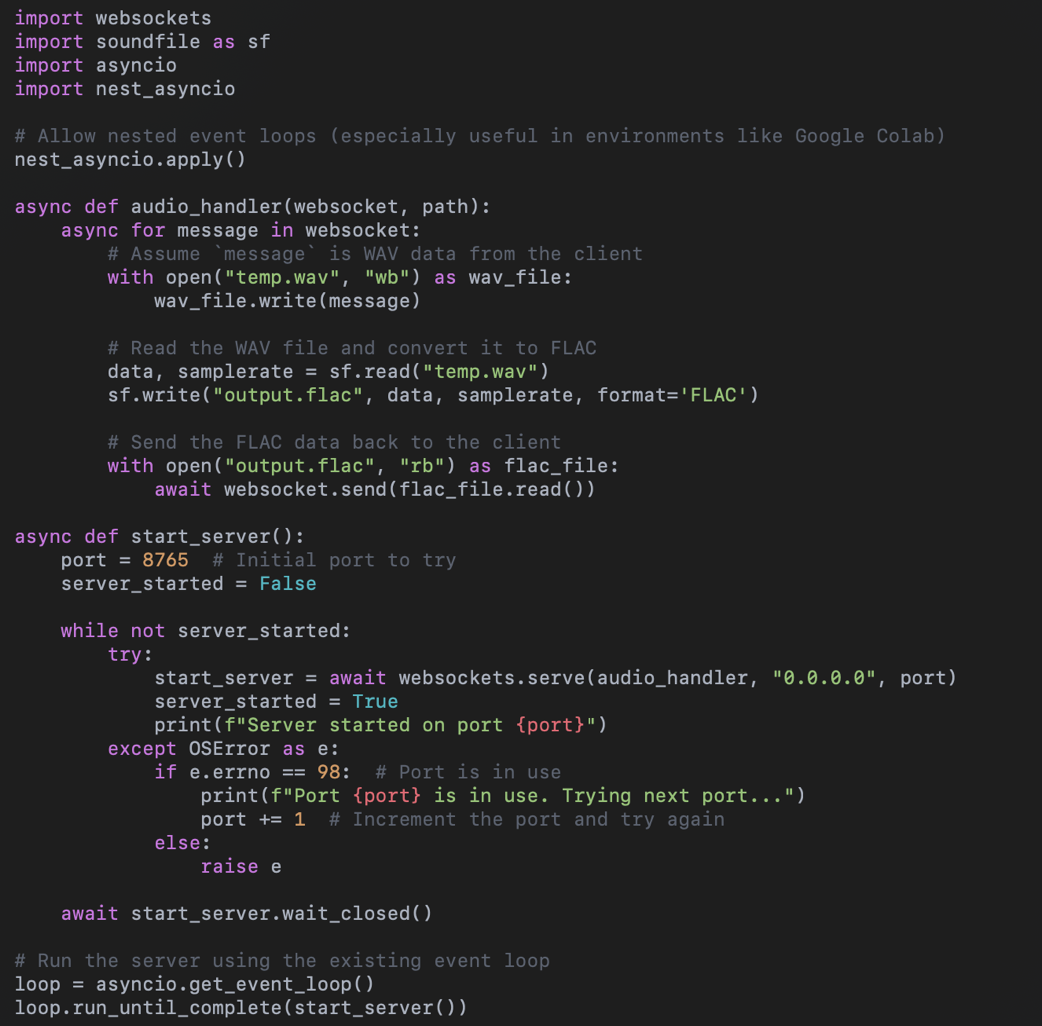
1. Install Requirements:

Run the following command to install the required libraries:



1. Create the Server Script:

Create a new Python file (e.g., audio\_server.py) and copy the following code into it:



**Configuration**

* Port Configuration: The server will attempt to bind to port 8765 by default. If that port is in use, it will increment the port number until an available one is found.

**Usage Instructions**

1. Run the Server:

Execute the script using Python:

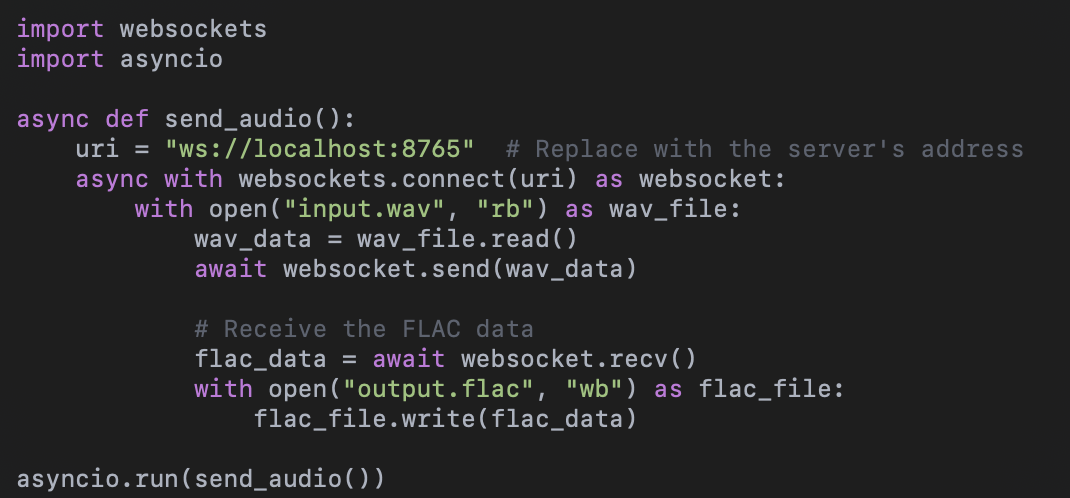


1. Connect to the Server:

Use a WebSocket client to connect to the server. The server will accept WAV audio data sent through the WebSocket connection and return the converted FLAC audio data.

1. Example Client Code:

Here is a simple example of a client that connects to the WebSocket server and sends WAV audio data:



Conversion Process Using the Model

**Text Analysis:** First, the model analyses the input text to understand its structure, including phonetics and semantics. It breaks down the words into smaller units called phonemes, which represent the distinct sounds of the language.

**Prosody Prediction:** Next, the model predicts prosodic features such as pitch, duration, and intensity for each phoneme. This step is crucial for ensuring the generated speech sounds natural and expressive.

**Waveform Generation:** Finally, the model synthesizes the audio waveform corresponding to the phonemes and prosody using advanced techniques, such as neural vocoders. This results in high-quality audio that mimics human speech patterns.

Model and Scripts Sharing

If permissible, you can share this server code and any additional scripts for community use and further research. Consider using platforms like GitHub to host the code, making it accessible for others to clone, contribute to, or adapt for their own projects.

Troubleshooting

•Port Already in Use: If you encounter an error about the port being in use, check for other instances of the server or applications using that port. You may manually specify a different port.

•Dependency Issues: Ensure all required packages are installed. Use the command pip install -r requirements.txt to install dependencies.