main

February 15, 2024

[ ]:
```
import time                             # For Measuring Latency
import pyaudio                          # For Real-Time Digital Audio Input
import numpy as np                      # For Arrays
import tensorflow as tf                 #
from tensorflow import keras            #
import matplotlib.pyplot as plt         # For Plots
```

[ ]:
```
# Define audio parameters
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 44100
CHUNK = 1024
RECORD_SECONDS = 5
```

[ ]:
```
# Initialize PyAudio
audio = pyaudio.PyAudio()
```

[ ]:
```
# Open stream
stream = audio.open(format=FORMAT, channels=CHANNELS,
                    rate=RATE, input=True,
                    frames_per_buffer=CHUNK)
```

[ ]:
```
# Load pre-trained TensorFlow model
model = tf.keras.models.load_model('your_pretrained_model.h5')
```

[ ]:
```
# Function to preprocess audio data
def preprocess_audio(data):
    # Convert to numpy array and normalize
    data = np.frombuffer(data, dtype=np.int16) / 32768.0
    # Perform additional preprocessing if necessary (e.g., spectrogram␣
 ↪conversion)
    # Example: spectrogram = np.abs(np.fft.rfft(data))[:512]
    return data
```

[ ]:
```
try:
    print("Recording...")
    for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
        # Read audio data from stream
```

```
        audio_data = stream.read(CHUNK)
        # Preprocess audio data
        processed_data = preprocess_audio(audio_data)
        # Perform inference using the pre-trained model
        predictions = model.predict(np.expand_dims(processed_data, axis=0))
        # Process predictions as needed
        # Example: print(predictions)
except KeyboardInterrupt:
    print("Recording stopped by user.")
finally:
    # Stop stream and close PyAudio
    stream.stop_stream()
    stream.close()
    audio.terminate()
```

[ ]:
```
input_data = 0
output_data = 0

# Record audio input
input_data = input_stream.read(CHUNK)

# Function to measure latency
def measure_latency():
    global input_data
    global output_data

    # Record start time
    start_time = time.time()

    # DSP System
    output_data = rnn_digital_filter(input_data)

    # Calculate elapsed time
    elapsed_time = time.time() - start_time

    # Calculate latency in milliseconds
    latency_ms = elapsed_time * 1000
    return latency_ms

print("Latency of the IIR filter:", latency, "seconds")

# Play back recorded audio
output_stream.write(output_data)
```

[ ]:
```
if __name__ == "__main__":
    pass
```