Creating a Vocoder Using Machine Learning Techniques

Ayush Saun
MT24024
ayush24024@iiitd.ac.in

Abstract

This report presents an investigation into the development of a vocoder utilizing machine learning techniques for audio synthesis. The study employs spectrograms and Mel spectrograms as intermediate representations and evaluates the AudioProcessingPipeline implementation using Griffin-Lim algorithm for reconstructing high-quality audio. The implementation demonstrates effective audio reconstruction capabilities with comprehensive evaluation metrics including Euclidean distance and KL divergence.

Dataset Description

The dataset utilized in this study consists of 15 short audio recordings with the following characteristics:

- **Audio clip duration:** 4-61 seconds
- **File format:** WAV
- **Sampling rate:** 16 kHz (resampled to 22.05 kHz for processing)
- **Preprocessing:** Short-Time Fourier Transform (STFT) applied to create spectrograms and Mel spectrograms
- **Mel bins:** 500 frequency bins for detailed spectral representation

Technical Implementation

Core Architecture: AudioProcessingPipeline Class

The vocoder implementation is built around the `AudioProcessingPipeline` class, which encapsulates the entire audio processing workflow:

- **Initialization:** Sets up directory structure for outputs (npy, wav, png folders)
- **Feature Extraction:** Converts raw audio to Mel-spectrograms
- **Reconstruction:** Uses Griffin-Lim algorithm for audio synthesis
- **Evaluation:** Computes similarity metrics for quality assessment
- **Visualization:** Generates spectrogram plots for analysis

Libraries and Technologies

The implementation leverages several key Python libraries:

- **Librosa:** Core audio processing library for loading, feature extraction, and Griffin-Lim reconstruction
- **NumPy:** Numerical operations and array manipulations
- **SciPy:** Distance calculations (Euclidean distance) and signal processing
- **SoundFile:** High-quality audio file I/O operations
- **Matplotlib:** Spectrogram visualization and plotting
- **Pandas:** Data management and CSV export for results

Methodology

1. Feature Extraction

The raw audio data conversion process involves:

Mel Spectrogram Computation:

Listing 1: Mel Spectrogram Extraction

```
def compute_mel_spectrogram(self, file_path, sr=22050):
    y, sr = librosa.load(file_path, sr=sr)
    mel_spec = librosa.feature.melspectrogram(y=y, sr=sr, n_mels=self.
        n_mels)
    mel_db = librosa.power_to_db(mel_spec, ref=np.max)
    return mel_db, sr
```

- **Mel Spectrograms:** Audio converted to 500-bin Mel spectrograms emphasizing perceptually important frequencies
- **Power to dB conversion:** Logarithmic scaling for better dynamic range representation
- **Standardized sampling rate:** 22.05 kHz for consistent processing

2. Audio Reconstruction

Griffin-Lim Algorithm Implementation:

Listing 2: Griffin-Lim Reconstruction

```
def griffin_lim_reconstruction(self, mel_spec, sr, n_iter=32):
    mel_power = librosa.db_to_power(mel_spec)
    linear_spec = librosa.feature.inverse.mel_to_stft(mel_power, sr=sr)
    reconstructed_audio = librosa.griffinlim(linear_spec, n_iter=n_iter
        )
    return reconstructed_audio
```

The reconstruction process:

- Converts dB scale back to linear power spectrogram
- Transforms Mel-spectrogram to linear STFT representation
- Applies Griffin-Lim algorithm with 32 iterations for phase estimation
- Synthesizes time-domain audio signal

Models & Algorithms

Griffin-Lim Algorithm

This iterative algorithm estimates missing phase information from magnitude spectrograms:

- **Phase Estimation:** Iteratively refines phase information
- **Convergence:** 32 iterations for optimal reconstruction quality
- **STFT Consistency:** Maintains consistency between time and frequency domains

Evaluation Metrics

Distance Calculations:

Listing 3: Evaluation Metrics Implementation

```
def normalize(self, data):
    data = np.abs(data)
    return data / np.sum(data)

def kl_divergence(self, p, q):
    p = np.clip(p, 1e-10, None)
    q = np.clip(q, 1e-10, None)
    return np.sum(p * np.log(p / q))
```

- **Euclidean Distance:** Measures point-wise differences in waveforms and spectrograms
- **KL Divergence:** Quantifies distributional differences after normalization
- **Dual Evaluation:** Metrics computed for both audio waveforms and Mel-spectrograms

Analysis

The vocoder performance is evaluated using comprehensive metrics:

Audio File Analysis

- Waveform-level comparison using Euclidean distance
- Probabilistic distribution analysis via KL divergence
- Temporal alignment through length normalization

Mel Spectrogram Analysis

- Frequency-domain reconstruction fidelity
- Spectral content preservation assessment
- Visual comparison through PNG outputs

Pipeline Efficiency

The implementation processes all 15 audio files automatically:

- Batch processing with organized output structure
- Automated metric calculation and CSV export
- Visualization generation for qualitative assessment

Results

The vocoder demonstrates effective reconstruction capabilities:

Reconstruction Quality

- High perceptual clarity in reconstructed audio
- Minimal artifacts from Griffin-Lim phase estimation
- Preserved spectral characteristics across frequency bands

Quantitative Performance

- Low Euclidean distances indicate close waveform matching
- Controlled KL divergences show distributional similarity
- Consistent performance across varied audio durations (4-61 seconds)

Output Organization

The pipeline generates structured outputs:

- **NPY files:** Numerical spectrogram data for further analysis
- **WAV files:** Reconstructed audio for listening tests
- **PNG files:** Visual spectrogram comparisons
- **CSV files:** Quantitative metrics for statistical analysis

Spectrogram Visualization

The implementation generates side-by-side visual comparisons demonstrating reconstruction fidelity:

- **Original Mel Spectrograms:** Ground truth spectral representations
- **Reconstructed Spectrograms:** Post-Griffin-Lim spectral content
- **Visual Fidelity:** High correlation between original and reconstructed patterns
- **Frequency Preservation:** Maintained spectral envelope and harmonic structure

Conclusion

The AudioProcessingPipeline successfully demonstrates machine learning-based vocoder capabilities through:

- Robust Mel-spectrogram feature extraction using Librosa
- Effective Griffin-Lim reconstruction maintaining audio quality
- Comprehensive evaluation framework with multiple metrics
- Automated processing pipeline with organized output structure
- Scalable architecture suitable for larger datasets

Future enhancements could include neural vocoder integration for improved phase estimation and reduced reconstruction artifacts.