Mel-Spectrogram Conversion: Mathematical Framework & Technical Implementation

Abstract

This document provides a comprehensive technical overview of the mel-spectrogram conversion process, including mathematical foundations, psychoacoustic motivations, and implementation considerations for audio processing in machine learning applications.

1. Mathematical Pipeline Overview

1.1 Discrete Audio Signal Representation

The input audio signal is represented as a discrete-time sequence:

```
x[n], n = 0, 1, ..., N-1
```

Parameters: - N = 32,000 samples - fs = 32,000 Hz (sampling frequency) - Duration = 1.0 second - Amplitude range: [-1.0, +1.0]

1.2 Short-Time Fourier Transform (STFT)

The STFT decomposes the audio signal into time-frequency components:

```
X[m,k] = Σ(n=0 to L-1) x[n+mH] · w[n] · e^(-j2 kn/L)
```

Parameters: - **Window function**: w[n] = Hann window = 0.5(1 - cos(2 n/(L-1))) - **Window length**: L = 1024 samples - **Hop length**: H = 320 samples - **Time frames**: m = 0, 1, …, M-1 where M = 101 - **Frequency bins**: k = 0, 1, …, K-1 where K = 513 - **Frequency resolution**: Δf = fs/L = 31.25 Hz - **Time resolution**: Δt = H/fs = 10 ms - **Overlap factor**: (L-H)/L = 68.75%

1.3 Power Spectral Density

Convert complex STFT coefficients to real power values:

```
P[m,k] = |X[m,k]|²
```

Properties: - Shape: (101, 513) - Represents energy distribution across time and frequency - Linear frequency scale: 0 to fs/2 Hz (Nyquist frequency)

1.4 Mel Filter Bank Application

Apply mel-scale filtering to the power spectrogram:

```
S[m,j] = Σ(k=0 to K-1) P[m,k] · H[j,k]
```

Filter Bank Design: - J = 64 triangular filters - Mel-spaced center frequencies - Coverage: fmin = 50 Hz to fmax = 14,000 Hz

Mel-Scale Frequency Mapping:

```
f_mel = 2595 × log (1 + f_hz/700)
```

Triangular Filter Response:

```
H[j,k] = {
    (f[k] - f[j-1])/(f[j] - f[j-1])  if f[j-1]   f[k]   f[j]
    (f[j+1] - f[k])/(f[j+1] - f[j])  if f[j]   f[k]   f[j+1]
    0                                otherwise
}
```

1.5 Logarithmic Compression

Apply logarithmic compression to the mel-filtered spectrogram:

```
M[m,j] = log (S[m,j] +  )
```

Parameters: -   = 1e-10 (numerical stability constant) - Converts to decibel-like scale - Final shape: (101, 64)

1.6 Final Mel-Spectrogram

The resulting mel-spectrogram M ^(101×64) contains: - **Rows**: 101 time frames (10ms temporal resolution) - **Columns**: 64 mel-frequency bins - **Values**: Log-mel energy coefficients - **Format**: Ready for CNN processing

2. Psychoacoustic Foundations

2.1 Human Auditory System Characteristics

The mel-scale is based on empirical observations of human auditory perception:

1. **Logarithmic Frequency Perception**: Humans perceive frequency differences logarithmically
2. **Critical Bands**: Auditory system processes sound in frequency bands
3. **Masking Effects**: Loud sounds can mask nearby frequencies
4. **Non-linear Loudness Perception**: Perceived loudness follows Weber-Fechner law

2.2 Mel-Scale Derivation

Historical Background: - Based on equal-pitch interval experiments (Stevens & Volkmann, 1940) - 1000 Hz = 1000 mel (reference point) - Derived from psychoacoustic experiments on pitch perception

Mathematical Justification: The mel-scale approximates the critical band structure of human hearing, providing: - Higher resolution at low frequencies (speech fundamentals) - Lower resolution at high frequencies (less perceptually important) - Optimal balance between perceptual relevance and computational efficiency

2.3 Weber-Fechner Law Application

Logarithmic Compression Rationale:

```
Perceived Intensity   log(Physical Intensity)
```

This justifies the logarithmic compression step, which: - Mimics auditory nerve response - Reduces dynamic range for neural network stability - Improves gradient flow during training

3. Implementation Details

3.1 Computational Complexity

STFT Computation: - Time Complexity: O(M · L · log L) using FFT - M = 101 frames, L = 1024 - Total Operations: ~1.04M per audio segment

Mel Filtering: - Time Complexity: O(M · K · J) - M = 101, K = 513, J = 64 - Total Operations: ~3.31M per audio segment

Memory Requirements: - Mel-spectrogram: 101 × 64 × 4 bytes = 25.9 KB per segment - Intermediate STFT: 101 × 513 × 8 bytes = 415 KB per segment - GPU-friendly: Supports efficient batch processing

3.2 Design Trade-offs

Window Length Selection: - **Larger L**: Better frequency resolution, worse time resolution - **Smaller L**: Better time resolution, worse frequency resolution - **Chosen L = 1024**: Optimal for speech/music (32ms @ 32kHz)

Hop Length Selection: - **H = L/4**: Common choice providing 75% overlap - **Balances**: Temporal smoothness vs. computational efficiency - **Result**: 10ms frame rate suitable for dynamic audio analysis

Mel Bins Selection: - **64 bins**: Empirically optimal balance - **Coverage**: Perceptually relevant frequency range - **Efficiency**: Suitable for CNN architectures

3.3 Numerical Stability Considerations

Epsilon Addition:

```
S[m,j] +   where   = 1e-10
```

- Prevents log(0) = -∞
- Maintains gradient flow in neural networks
- Typical range: 1e-8 to 1e-10

Floating Point Precision: - **Data Type**: float32 (sufficient precision, memory efficient) - **Dynamic Range**: ~10^-38 to 10^38 - **Precision**: ~7 decimal digits (adequate for audio processing)

Normalization Strategies: 1. **Per-sample normalization**: Each audio file independently 2. **Global statistics**: Dataset-wide mean/std normalization 3. **Batch normalization**: Applied within neural network layers

3.4 Alternative Approaches

Variants and Extensions: 1. **MFCC**: Discrete Cosine Transform of log-mel coefficients 2. **Gammatone Filters**: More biologically plausible filter bank 3. **Constant-Q Transform**: Logarithmic frequency resolution 4. **Spectral Features**: Centroid, rolloff, spectral flux

Modern Alternatives: 1. **Raw Waveform Processing**: End-to-end learning (WaveNet, SampleRNN) 2. **Learnable Filter Banks**: Trainable frequency decomposition 3. **Attention Mechanisms**: Time-frequency attention modeling 4. **Self-supervised Learning**: Contrastive audio representations

4. Implementation Code Structure

4.1 Core Implementation (Python/PyTorch)

```python
import torch
import torchaudio
from torchlibrosa.stft import Spectrogram, LogmelFilterBank

# Initialize extractors
spectrogram_extractor = Spectrogram(
    n_fft=1024,
    hop_length=320,
    win_length=1024,
    window='hann',
    center=True,
    pad_mode='reflect',
    freeze_parameters=True
)

logmel_extractor = LogmelFilterBank(
    sr=32000,
    n_fft=1024,
    n_mels=64,
    fmin=50,
    fmax=14000,
    ref=1.0,
    amin=1e-10,
    top_db=None,
    freeze_parameters=True
)
```

```python
# Process audio
def extract_mel_spectrogram(audio_tensor):
    # STFT
    spectrogram = spectrogram_extractor(audio_tensor)

    # Mel filtering
    mel_spectrogram = logmel_extractor(spectrogram)

    return mel_spectrogram
```

4.2 Batch Processing Considerations

GPU Optimization: - Process multiple audio segments simultaneously - Utilize tensor operations for efficiency - Memory-efficient for large datasets

Preprocessing Pipeline:

```python
# Segment audio into 1-second chunks
def segment_audio(audio, segment_length=32000):
    segments = []
    for i in range(0, len(audio), segment_length):
        segment = audio[i:i+segment_length]
        if len(segment) < segment_length:
            # Pad last segment
            segment = torch.nn.functional.pad(
                segment, (0, segment_length - len(segment))
            )
        segments.append(segment)
    return torch.stack(segments)
```

5. Performance Considerations

5.1 Real-time Processing

Latency Analysis: - Frame-based processing: 10ms latency per frame - Suitable for real-time applications - GPU acceleration enables batch real-time processing

Throughput Optimization: - Vectorized operations using SIMD instructions - GPU tensor operations for parallel processing - Memory layout optimization for cache efficiency

5.2 Quality Metrics

Reconstruction Fidelity: - Signal-to-noise ratio after mel-scale conversion - Perceptual evaluation of audio quality (PESQ, STOI) - Preservation of relevant acoustic features

Machine Learning Performance: - Classification accuracy on downstream tasks - Convergence speed during neural network training - Generalization across different audio domains

6. Applications and Extensions

6.1 Emotion Recognition

Feature Relevance: - Low-frequency content: Fundamental frequency patterns - Mid-frequency content: Formant structure - High-frequency content: Spectral texture

Temporal Dynamics: - Frame-level analysis: Instantaneous emotional content - Sequence modeling: Temporal evolution of emotions - Attention mechanisms: Focus on emotionally relevant segments

6.2 Speech Processing

Phonetic Information: - Formant frequencies: Vowel identification - Spectral transitions: Consonant characteristics - Prosodic features: Stress, intonation patterns

6.3 Music Analysis

Harmonic Content: - Chord progression analysis - Key detection and modulation - Timbre characterization

7. Conclusion

The mel-spectrogram representation provides an optimal balance between: - **Perceptual Relevance**: Matches human auditory processing - **Computational Efficiency**: Suitable for deep learning applications - **Information Preservation**: Retains essential acoustic characteristics - **Practical Implementation**: Robust and numerically stable

This framework has become the standard for audio preprocessing in machine learning, enabling significant advances in speech recognition, music analysis, and audio emotion recognition systems.

References

1. Stevens, S. S., & Volkmann, J. (1940). The relation of pitch to frequency: A revised scale. American Journal of Psychology, 53(3), 329-353.
2. Davis, S., & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Transactions on Acoustics, Speech, and Signal Processing, 28(4), 357-366.

3. Rabiner, L., & Juang, B. H. (1993). Fundamentals of speech recognition. Prentice Hall.
4. Müller, M. (2015). Fundamentals of music processing: Audio, analysis, algorithms, applications. Springer.
5. Gemmeke, J. F., et al. (2017). Audio Set: An ontology and human-labeled dataset for audio events. IEEE ICASSP.

Document prepared for technical implementation of mel-spectrogram conversion in audio processing systems.