Title: Haptica

Motivation:

Music is experienced in many ways: through speakers, headphones, or tactile vibrations, and Deaf and hard-of-hearing (DHH) individuals have diverse preferences and hearing levels. Some have residual hearing and enjoy music through speakers or headphones; some prefer feeling loud music as tactile vibrations; some prefer quiet; some enjoy signed music, while others do not. DHH people are not a monolith; they are individuals with their own preferences and hearing levels. Haptica explores translating musical elements into vibration patterns so transitions and energy changes can be felt, not just heard. This prototype does not aim to "fix" anything or offer a universal solution; it is an experiment to learn and move closer to making music accessible in more forms. By running a tiny depthwise-separable CNN on an Arduino Nano, we investigate whether sound felt through the skin can convey the dynamic structure of music, opening new pathways for inclusive musical experiences.

Objective:

- Design and deploy a lightweight depthwise-separable CNN that runs on an Arduino Nano to process audio in real time.
- Convert audio signals into vibration patterns by extracting musical features (transitions, energy changes, rhythmic elements) using mel-spectrogram analysis.
- Implement a system that translates audio features into distinct haptic patterns across multiple motors, enabling tactile representation of musical dynamics.
- Optimize the model and inference pipeline to meet the memory and computational constraints of embedded hardware.

Platform: Google Colab, Arduino IDE, Local IDE
Dataset:

- 5 different songs that resonate well with deaf and hard of hearing people
    - Appangal Eambathu
    - Without Me
    - Am I dreaming
    - Seven Nation Army
    - VibrationBass Music

Code: github
Demo Video: Demo

Components:

- Arduino Nano BLE
- 1× 470 µF electrolytic
- 3× 0.1 µF ceramics
- 3× N-MOSFETs
- 3× SS14 diodes
- 3× 100 Ω gate resistors
- 3× 100 kΩ pulldowns
- 3x coin vibrators
- Heat-shrink
- Power Bank
- USB A to screw terminal adapter
- 5 V → 3.3 V buck converter, ≥2 A (LM2596)
- Breadboard
- Jumperwires

Algorithm

Audio feature extraction

- Mel-spectrogram computation: Uses librosa to convert audio to mel-scale spectrograms (32 mel bins, 20–8000 Hz). Parameters: window length 25 ms, hop 10 ms, Hann window, power=2.0, Slaney normalization.
- Beat detection: librosa.beat.beat_track() estimates tempo (BPM) and beat locations using dynamic programming on onset strength.
- Onset detection: librosa.onset.onset_detect() finds musical events (hits, kicks) using spectral flux with delta thresholding.
- RMS energy extraction: Computes root-mean-square energy per frame for energy envelope analysis.

Frequency band separation

- Splits mel-spectrogram into three bands:
- Low: ≤ 250 Hz (bass)
- Mid: 250–2000 Hz (midrange)
- High: > 2000 Hz (treble)
- Extracts per-band energy via mean pooling and log scaling.

Deep learning model architecture

- Depthwise-separable CNN: Two blocks of:
- Causal zero-padding (left-pad along time)
- Depthwise convolution (k×1, k=3) for temporal patterns
- Pointwise convolution (1×1) for channel mixing
- Global Average Pooling: Aggregates spatial-temporal features.
- Dense output layer: Sigmoid activation outputs three motor intensities [0, 1].

- Optimization: Adam (lr=1e-3), MSE loss, early stopping, learning rate reduction on plateau.

Model inference pipeline

- Sliding window extraction: Extracts 20-frame windows from the mel-spectrogram.
- TensorFlow Lite inference: Quantized model (INT8) for Arduino deployment.
- Real-time processing: Processes windows at ~100 Hz (10 ms hop).
- This algorithm fuses scaling, causal masking, and numerically stable softmax into one GPU kernel.

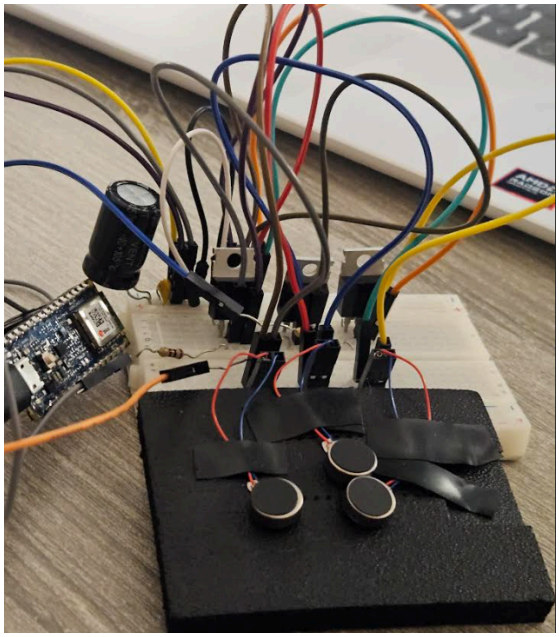
Pattern generation algorithms

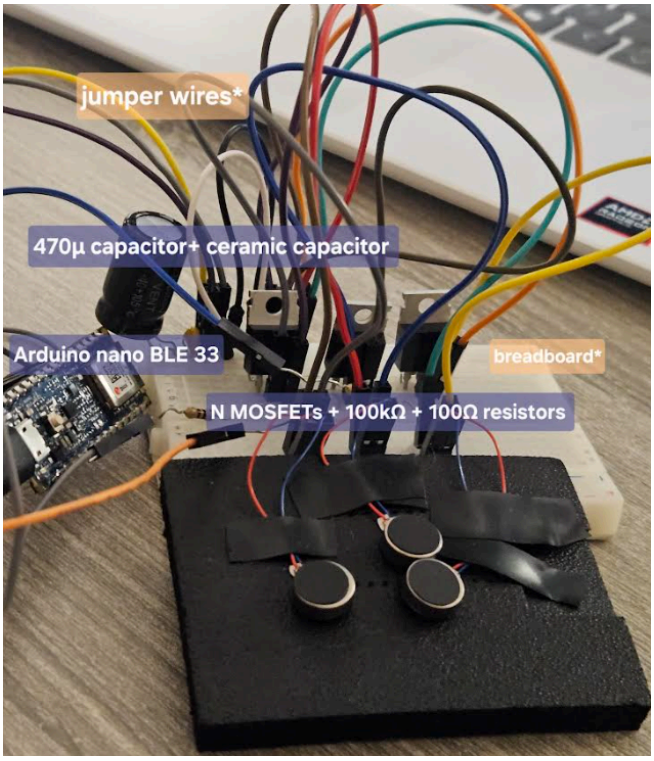
- Heartbeat pattern (Low motor): Thump-thump-pause with configurable pulse duration (150 ms) and gap (300 ms).
- Zigzag pattern (Mid motor): Sawtooth oscillation with 200 ms period.
- Beat pulse pattern (High motor): Short pulses (80 ms) synchronized with detected beats.
- Inference modulation: Model outputs gate pattern activation, combining learned features with rhythmic patterns.

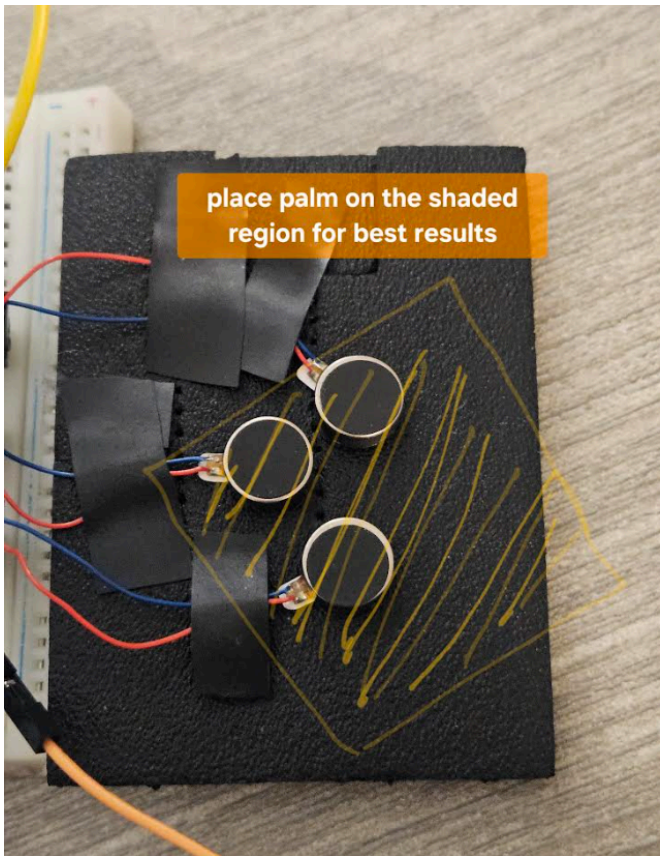
Embedded control algorithms

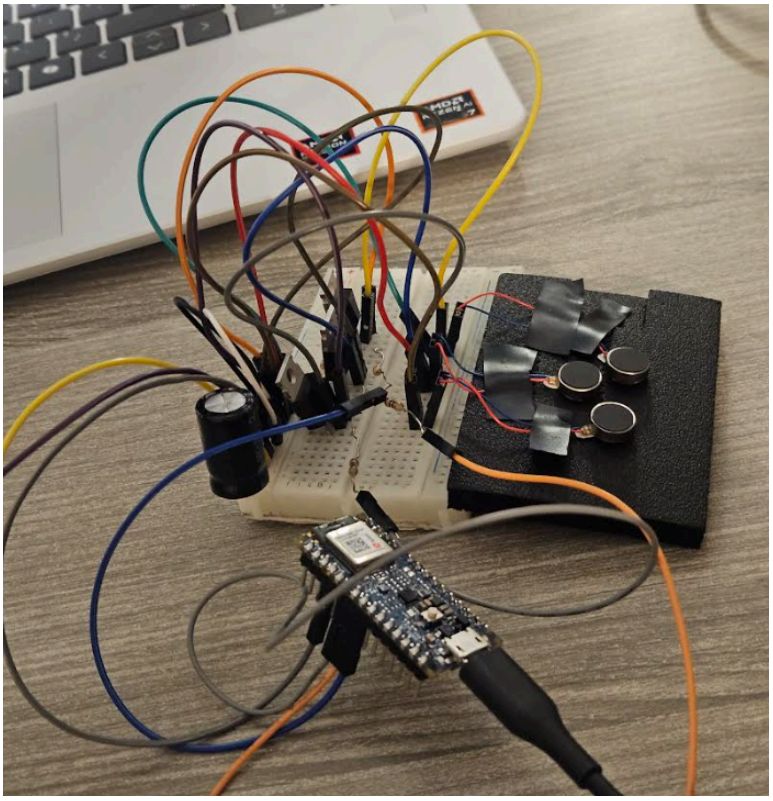
- PWM generation: Maps model outputs [0, 1] to PWM values [0, 255] for motor control.
- Serial communication protocol: ASCII-based protocol with commands (READY, COUNTDOWN, START, END) and motor value transmission (L M H format).
- Real-time synchronization: Frame-based timing to align vibrations with audio playback.
- Parallel mapping:

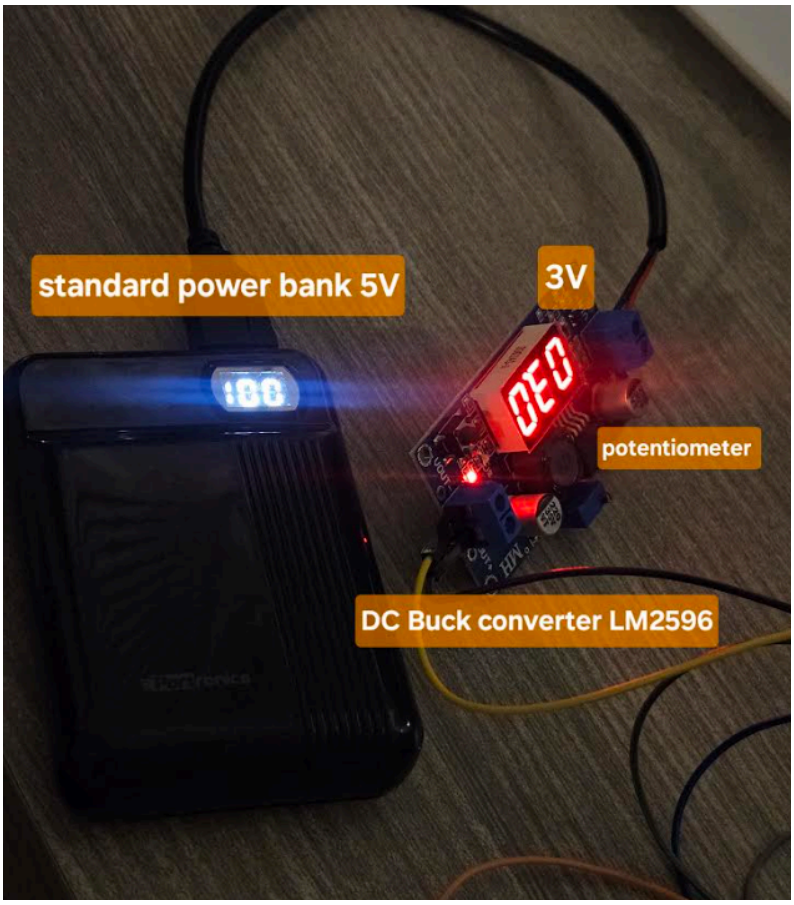
Hardware Setup:




jumper wires*
470µ capacitor+ ceramic capacitor
Arduino nano BLE 33
breadboard*
N MOSFETs + 100kΩ + 100Ω resistors


place palm on the shaded region for best results




standard power bank 5V
3V
potentiometer
DC Buck converter LM2596

Code Explanation:

1. ML Model (hapticTranslator.ipynb)

Purpose: Train a CNN to predict motor intensities from audio.

Key Steps:
- Data Preparation: Load audio files, compute mel-spectrograms (32 mel bins, 20-8000 Hz), extract 20-frame windows
- Target Generation: Create 3-channel targets (low, mid, high) from frequency band energy, beat detection, and onset accents
- Model Architecture:
  - Input: (20 frames × 32 mel bins × 1 channel)
  - Two blocks of depthwise-separable convolution (k=3, channels=16)
  - Global Average Pooling → Dense(3) with sigmoid
- Training: Adam optimizer, MSE loss, early stopping
- Export: Convert to TensorFlow Lite (INT8) for Arduino

2. application.py

Purpose: Real-time inference and motor control.

Main Flow:
1. Setup : Load TFLite model, configure serial port
2. Audio Processing :
   - Load MP3 file → compute full mel-spectrogram
3. Beat Detection : Detect beats and tempo using librosa
4. Inference Loop :
   - Extract 20-frame windows from mel-spectrogram
   - Run model inference → get (y_low, y_mid, y_high)
   - Apply vibration patterns (heartbeat, zigzag, beat pulses)
   - Convert to PWM values [0-255]
   - Send to Arduino via serial: "L M H\n"
5. Pattern Generation: Generate distinct patterns for each motor based on inference output and beat locations

Key Functions:
- compute_full_melspectrogram(): Audio → mel-spectrogram
- extract_mel_window(): Extract 20-frame window
- run_model_on_window(): Run TFLite inference
- apply_patterns_with_inference(): Combine model output with patterns
- to_pwm(): Convert [0,1] → [0,255] PWM

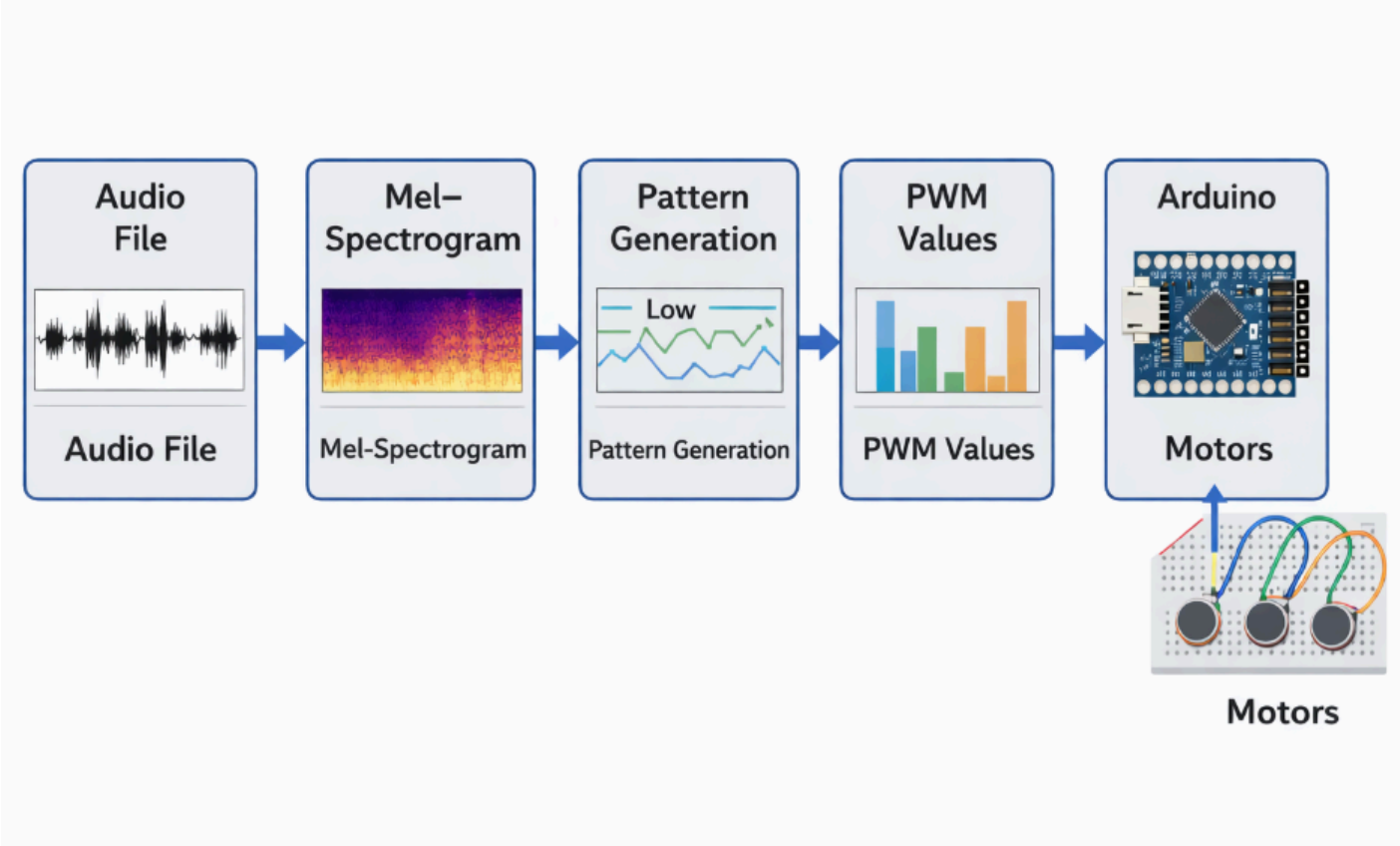
3. Arduino Code (brain.ino)

Purpose: Receive PWM values and drive motors.

- Initialize serial (115200 baud)
- Configure motor pins: M1 (pin 3) = low, M2 (pin 5) = mid
- Wait for serial connection

Main Loop:
1. Read Serial: Wait for incoming data
2. Parse Commands:
   - "READY" → respond "READY"
   - "COUNTDOWN N" → respond "COUNTING..."
   - "START" → respond "STARTED"
   - "END" → stop motors
3. Parse Motor Values: Read "L M H\n" format
4. Drive Motors: Use analogWrite() to set PWM on pins 3 and 5

Workflow:



Pseudocode:

Step 1: Pre-Processing

```
None
Audio File (MP3)
  ↓
librosa.load() → Waveform (16kHz, mono)
  ↓
compute_full_melspectrogram()
  ↓
Mel-Spectrogram: (32 mels × T frames)
  ↓
detect_beats() → Beat locations, tempo
```

Step 2: Real-Time Inference Loop

```
None
For each 20-frame window (sliding):
  ↓
1. Extract Window:
   extract_mel_window   → (20 frames × 32 mels × 1 channel)
  ↓
2. Model Inference:
   run_model_on_window(window)
   → (y_low_base, y_mid_base, y_high_base) in [0,1]
  ↓
3. Apply Patterns:
   apply_patterns_with_inference()
   - Low motor: Heartbeat pattern (thump-thump-pause)
   - Mid motor: Zigzag pattern (sawtooth)
   - High motor: Beat pulses (synchronized with beats)
   → Pattern shapes × Inference values
  ↓
4. Convert to PWM:
   to_pwm() → [0,1] → [0,255]
   → (L_pwm, M_pwm, H_pwm)
  ↓
5. Send to Arduino:
   Serial: "L M H\n"
   → Arduino receives and drives motors
  ↓
6. Timing Control:
```

```
   Sleep to maintain sync with audio playback
   (10ms per frame = 100 Hz update rate)
```

Step 3: Arduino Processing

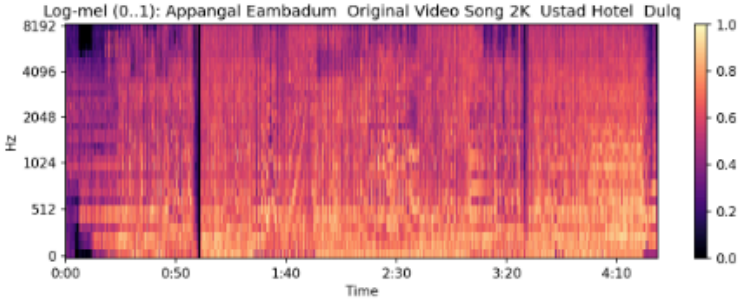
```
None
Arduino receives: "L M H\n"
  ↓
Parse values (L, M, H)
  ↓
Clamp to [0, 255]
  ↓
analogWrite(M1_PIN, L)  → Low motor (heartbeat)
analogWrite(M2_PIN, M)  → Mid motor (zigzag)
analogWrite(M3_PIN, H)  → High motor
  ↓
Motors vibrate according to patterns
```

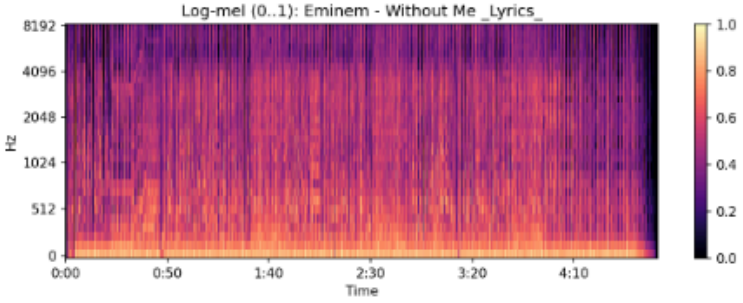
KEY TIMING DETAILS
Window Size: 20 frames = 200ms of audio (20 × 10ms hop)
Update Rate: 100 Hz (every 10ms)
Processing: Each window processed in <10ms to maintain real-time
Synchronization: Frame-based timing ensures vibrations sync with audio playback

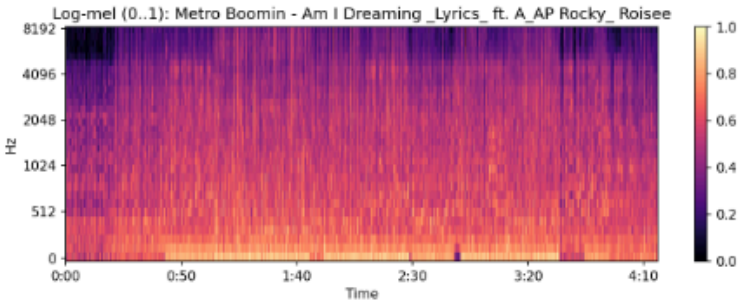
PATTERN GENERATION LOGIC
Low Motor (Heartbeat):
Pattern: Thump (150ms) → Gap (50ms) → Thump (150ms) → Long pause (1-1.5s)
Controlled by: y_low_base from model
Mid Motor (Zigzag):
Pattern: Sawtooth wave (200ms period)
Controlled by: y_mid_base from model
High Motor (Beat Pulses):
Pattern: Short pulses (80ms) on detected beats
Controlled by: y_high_base from model
Pattern Activation: Model output gates pattern intensity
If model output = 0.8, pattern runs at 80% intensity
If model output = 0.2, pattern runs at 20% intensity
This workflow runs in real time, processing audio and generating synchronized haptic feedback.

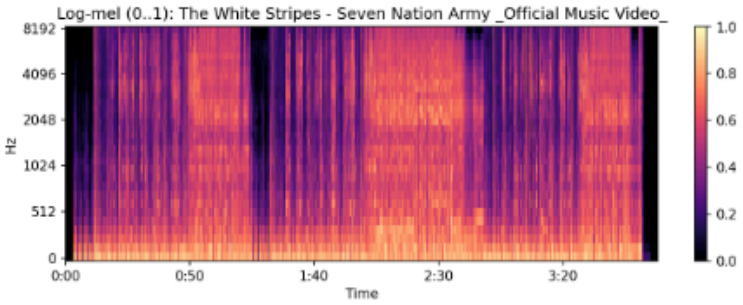
Implementation:

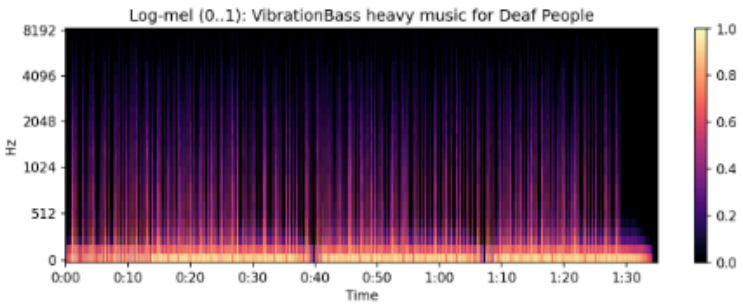
Log mel plots:



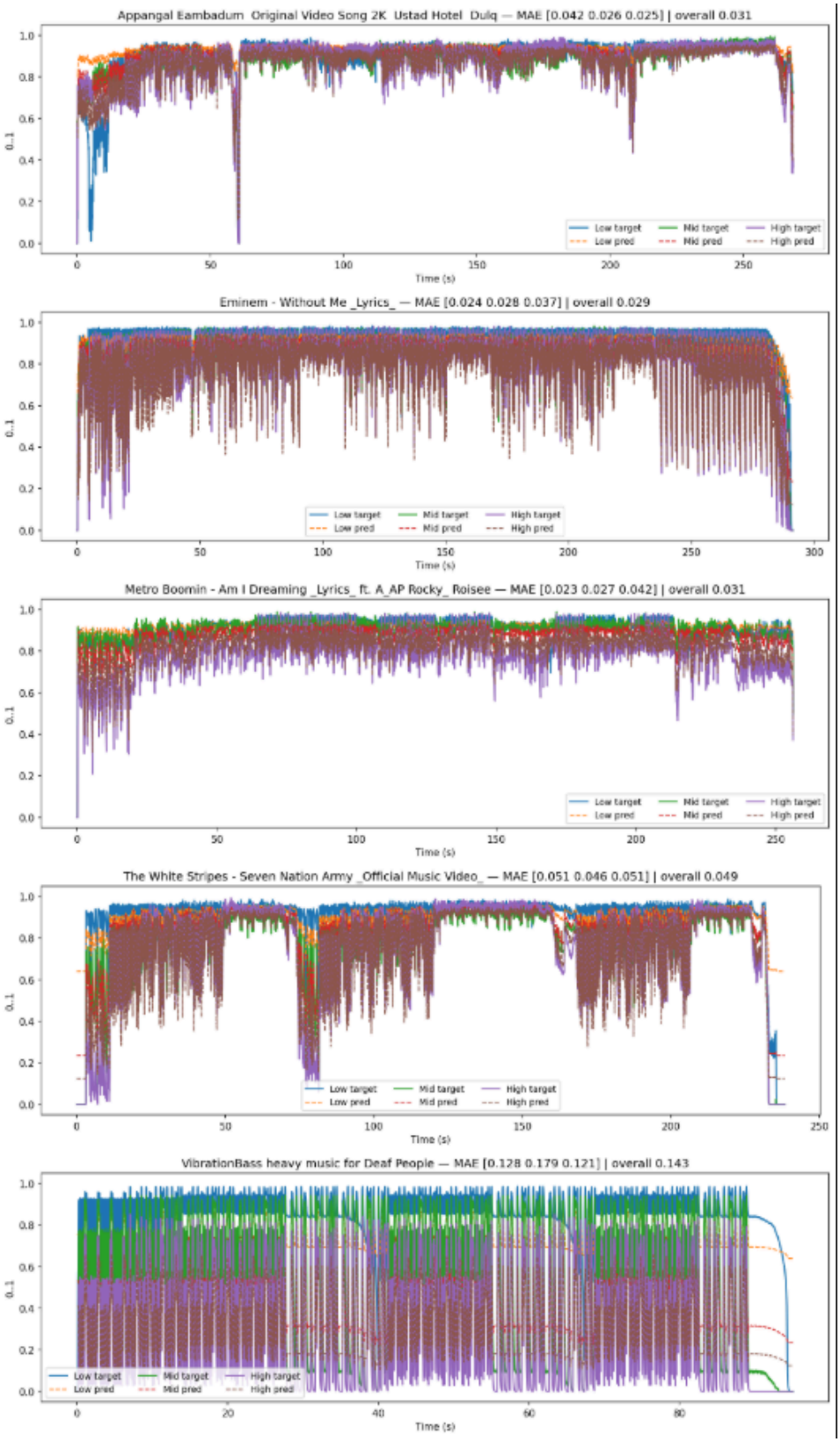




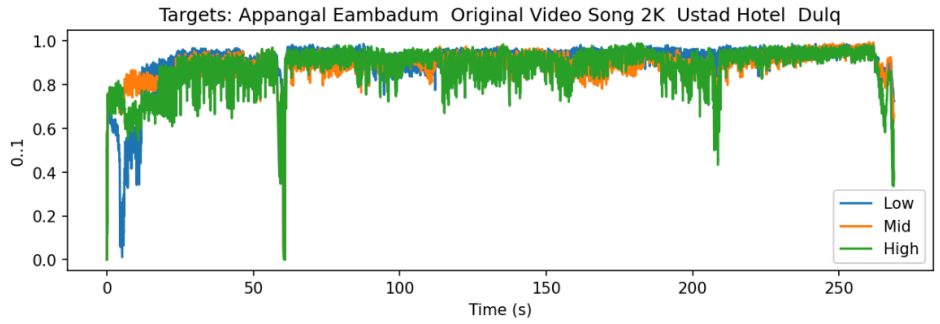


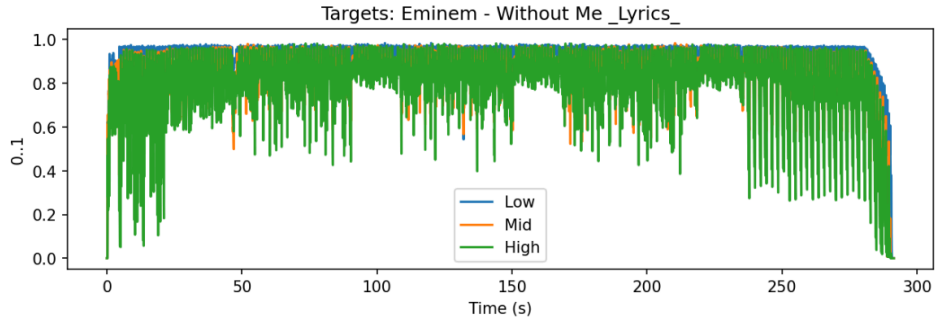


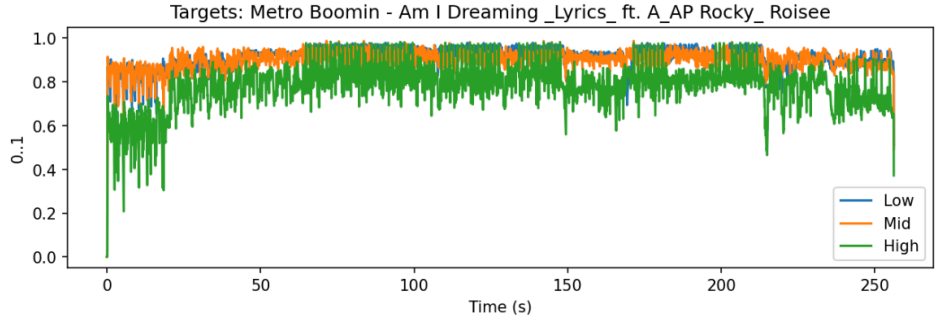
Predictions Plots:

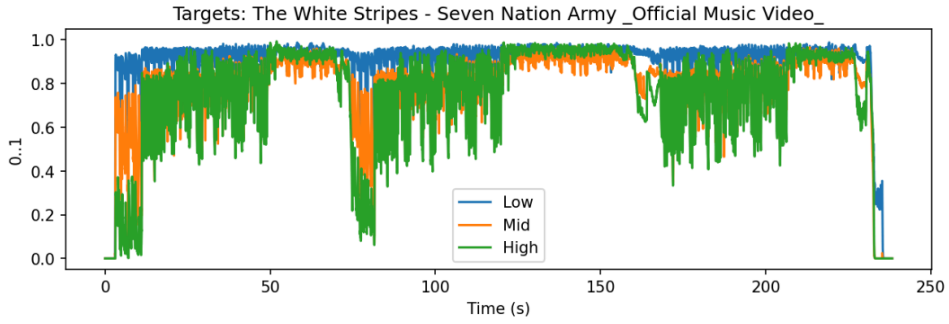


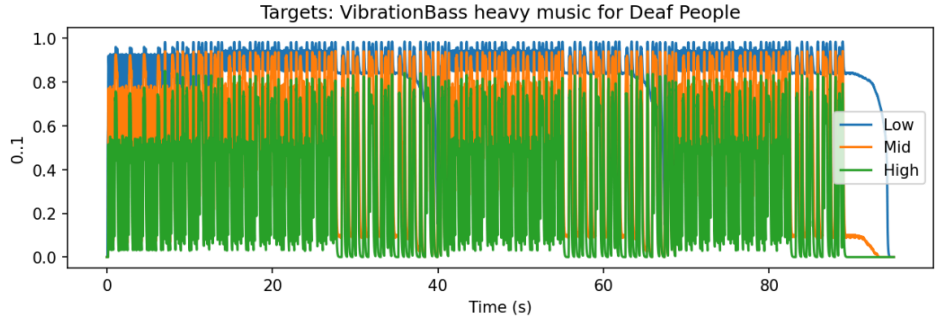
Target plot:


Targets: Appangal Eambadum Original Video Song 2K Ustad Hotel Dulq
0..1
Time (s)
Low
Mid
High

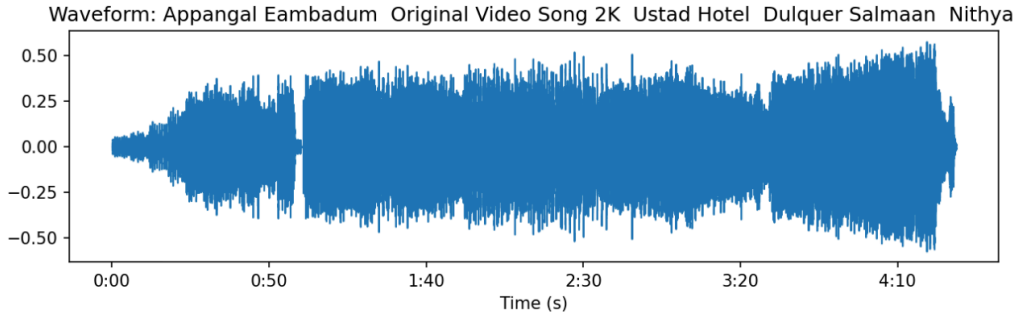

Targets: Eminem - Without Me _Lyrics_
0..1
Time (s)
Low
Mid
High

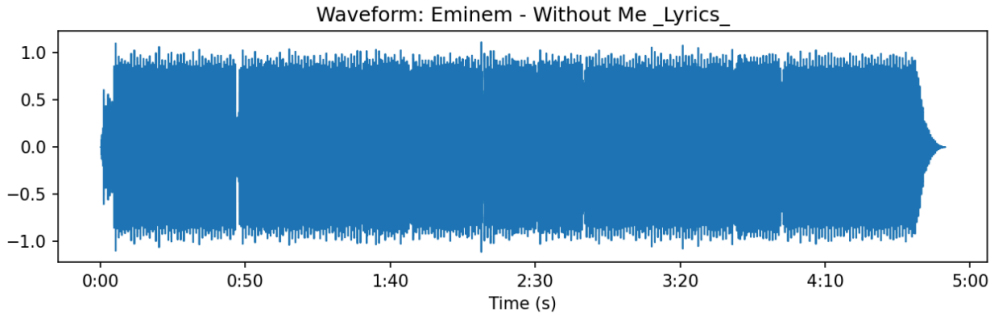

Targets: Metro Boomin - Am I Dreaming _Lyrics_ ft. A_AP Rocky_ Roisee
0..1
Time (s)
Low
Mid
High

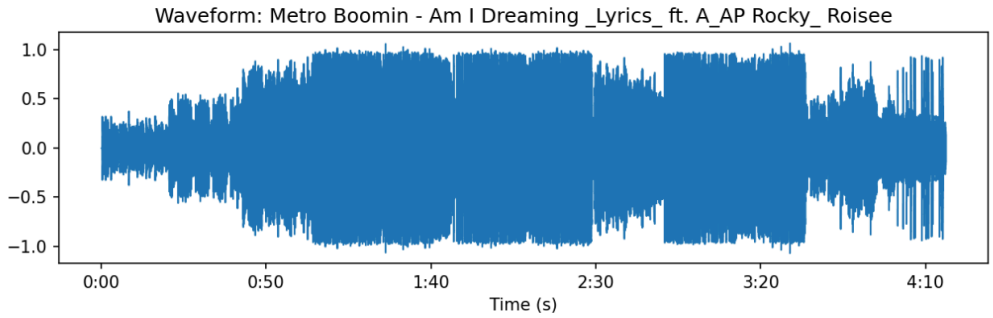

Targets: The White Stripes - Seven Nation Army _Official Music Video_
0..1
Time (s)
Low
Mid
High

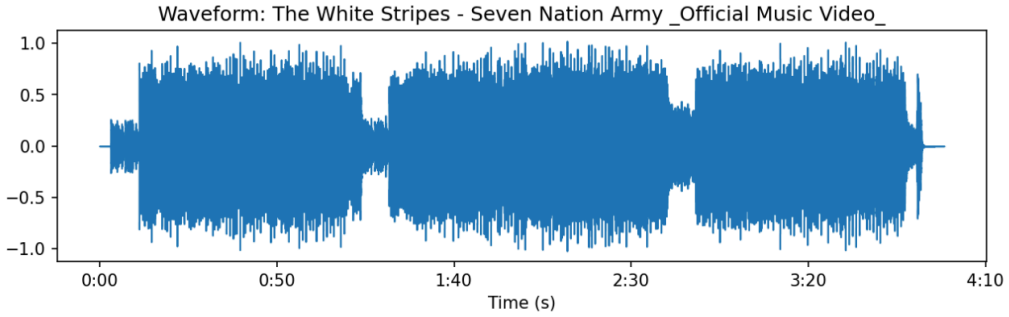

Targets: VibrationBass heavy music for Deaf People
0..1
Time (s)
Low
Mid
High

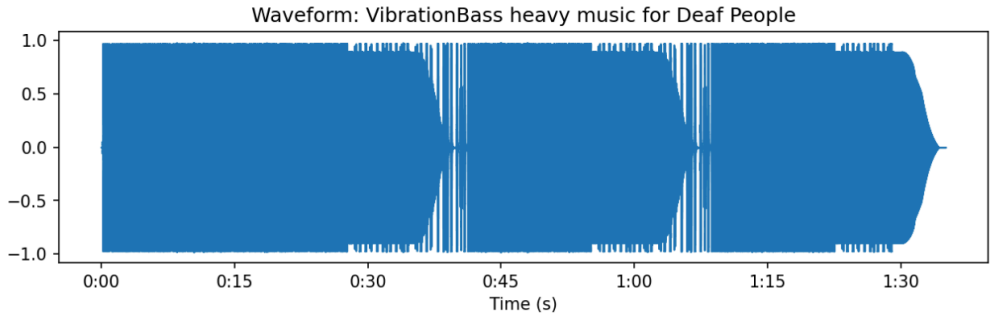
Waveforms:


Waveform: Appangal Eambadum Original Video Song 2K Ustad Hotel Dulquer Salmaan Nithya
0.50
0.25
0.00
−0.25
−0.50
0:00
0:50
1:40
2:30
3:20
4:10
Time (s)


Waveform: Eminem - Without Me _Lyrics_
1.0
0.5
0.0
−0.5
−1.0
0:00
0:50
1:40
2:30
3:20
4:10
5:00
Time (s)


Waveform: Metro Boomin - Am I Dreaming _Lyrics_ ft. A_AP Rocky_ Roisee
1.0
0.5
0.0
−0.5
−1.0
0:00
0:50
1:40
2:30
3:20
4:10
Time (s)


Waveform: The White Stripes - Seven Nation Army _Official Music Video_
1.0
0.5
0.0
−0.5
−1.0
0:00
0:50
1:40
2:30
3:20
4:10
Time (s)


Waveform: VibrationBass heavy music for Deaf People
1.0
0.5
0.0
−0.5
−1.0
0:00
0:15
0:30
0:45
1:00
1:15
1:30
Time (s)

Application UI:

The song is loaded and testing is done between all motors for a quick sanity check.

Haptica visualization pipeline: waveform → mel spectrogram → log-mel windows → predicted pattern labels over time for a song.

These plots show how the model tracks transitions and energy in the music before mapping them to motor vibration patterns.

```
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
============================================================
HAPTIC TRANSLATOR - INFERENCE
============================================================
Loaded TFLite model: tflite/inference.tflite
Input shape: [ 1 20 32  1], dtype: <class 'numpy.int8'>
Output shape: [1 3], dtype: <class 'numpy.int8'>

============================================================
HAPTIC TRANSLATOR - INFERENCE STARTED
============================================================
Keyboard control: SPACE=pause/resume, S=stop
✓ Loaded song: song1.mp3
  Duration: 256.16 s, Sample rate: 16000 Hz
Loaded song: song1.mp3 (256.16s, 16000Hz)
Computing mel spectrogram...
✓ Mel spectrogram: 32 mels × 25617 frames
Mel spectrogram computed: 32 mels × 25617 frames
Detecting beats and tempo...
```

```
  tempo = float(tempo)
✓ Detected 385 beats at 89.6 BPM
Distinct patterns created: Heartbeat (Low), Zigzag (Mid), Beat Pulses (High)

============================================================
GENERATING VISUALIZATIONS
============================================================
Plotting audio waveform...
✓ Saved: plots/inference/song1_20251205_205430_waveform.png
Plotting mel spectrogram...
✓ Saved: plots/inference/song1_20251205_205430_melspectrogram.png
Plotting sample windows...
✓ Saved: plots/inference/song1_20251205_205430_sample_windows.png
Plotting distinct patterns...
✓ Saved: plots/inference/song1_20251205_205430_distinct_patterns.png

============================================================
CONNECTING TO ARDUINO
============================================================
✓ Serial connection opened on COM5
Serial connected: COM5

============================================================
MOTOR CHECK SEQUENCE
============================================================
Testing
Testing
test complete
test complete
Motor check sequence complete!
Motor check sequence complete!
============================================================
```

```
============================================================
SYNCHRONIZATION COUNTDOWN
============================================================
Sending READY to Arduino...
  Arduino: READY

Starting 5-second countdown...
  5...   4...   3...   2...   1... GO!

Sending START to Arduino...
  Arduino: STARTED
============================================================

Countdown synchronization complete
✓ Audio playback started: song1.mp3
Audio playback started

============================================================
RUNNING INFERENCE
============================================================
Total windows: 25598
Output rate: 100.0 Hz
Estimated duration: 255.98 s

🎼 Serial communication: ACTIVE
🎵 Song playing
🎹 Vibrations active

Controls: SPACE = pause/resume, S = stop
```

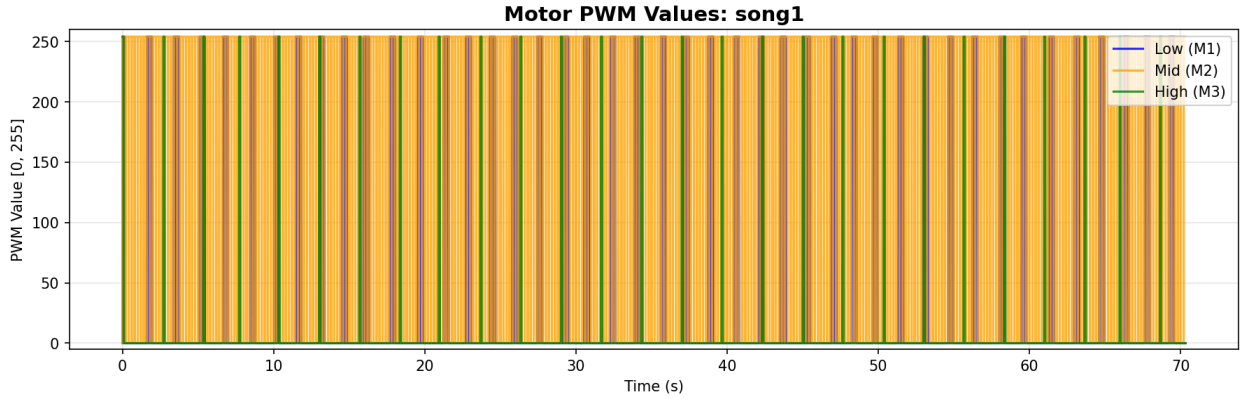
```
PAUSED - Press SPACE to resume, S to stop
Stop requested by user

Stopping motors...
Serial connection closed

============================================================
GENERATING RUNTIME PLOTS
============================================================
Plotting predictions timeline...
✓ Saved: plots/inference/song1_20251205_205430_predictions.png
Plotting motor PWM values...
✓ Saved: plots/inference/song1_20251205_205430_motor_pwm.png

============================================================
INFERENCE COMPLETE
============================================================
Total windows processed: 7034
All plots saved to: plots/inference/
  - song1_20251205_205430_waveform.png
  - song1_20251205_205430_melspectrogram.png
  - song1_20251205_205430_sample_windows.png
  - song1_20251205_205430_predictions.png
  - song1_20251205_205430_motor_pwm.png
Log file saved to: logs/inference_20251205_205430.log


C:\Users\shiva\Documents\VScode projects\haptica>
```

The song is broken down into multiple small window chunks

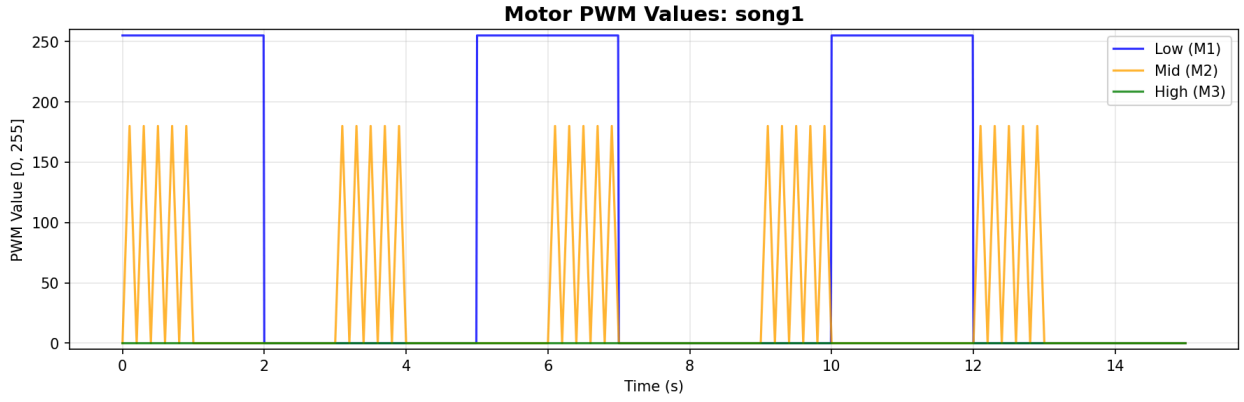
```
Window 0: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.00(heartbeat) M:0.03(zigzag) H:0.30(beat)] → Final=[0.000, 1.000, 1.000] → PWM=[  0, 254, 254]
Progress: 0.0% (0/25598 windows, 0.0s) - L=  0 M=254 H=254
Window 1: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.00(zigzag) H:0.30(beat)] → Final=[1.000, 0.000, 1.000] → PWM=[254,   0, 254]
Window 2: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.03(zigzag) H:0.30(beat)] → Final=[1.000, 1.000, 1.000] → PWM=[254, 254, 254]
Window 3: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.06(zigzag) H:0.30(beat)] → Final=[1.000, 1.000, 1.000] → PWM=[254, 254, 254]
Window 4: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.09(zigzag) H:0.30(beat)] → Final=[1.000, 1.000, 1.000] → PWM=[254, 254, 254]
Window 5: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.12(zigzag) H:0.30(beat)] → Final=[1.000, 1.000, 1.000] → PWM=[254, 254, 254]
Window 6: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.15(zigzag) H:0.30(beat)] → Final=[1.000, 1.000, 1.000] → PWM=[254, 254, 254]
Window 7: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.18(zigzag) H:0.30(beat)] → Final=[1.000, 1.000, 1.000] → PWM=[254, 254, 254]
Window 8: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.21(zigzag) H:0.00(beat)] → Final=[1.000, 1.000, 0.000] → PWM=[254, 254,   0]
Window 9: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.24(zigzag) H:0.00(beat)] → Final=[1.000, 1.000, 0.000] → PWM=[254, 254,   0]
Window 10: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.27(zigzag) H:0.00(beat)] → Final=[1.000, 1.000, 0.000] → PWM=[254, 254,   0]
Window 11: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.30(zigzag) H:0.00(beat)] → Final=[1.000, 1.000, 0.000] → PWM=[254, 254,   0]
Window 12: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.27(zigzag) H:0.00(beat)] → Final=[1.000, 1.000, 0.000] → PWM=[254, 254,   0]
Window 13: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.24(zigzag) H:0.00(beat)] → Final=[1.000, 1.000, 0.000] → PWM=[254, 254,   0]
Window 14: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.21(zigzag) H:0.00(beat)] → Final=[1.000, 1.000, 0.000] → PWM=[254, 254,   0]
Window 15: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.30(heartbeat) M:0.18(zigzag) H:0.00(beat)] → Final=[1.000, 1.000, 0.000] → PWM=[254, 254,   0]
Window 16: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.00(heartbeat) M:0.15(zigzag) H:0.00(beat)] → Final=[0.000, 1.000, 0.000] → PWM=[  0, 254,   0]
Window 17: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.00(heartbeat) M:0.12(zigzag) H:0.00(beat)] → Final=[0.000, 1.000, 0.000] → PWM=[  0, 254,   0]
Window 18: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.00(heartbeat) M:0.09(zigzag) H:0.00(beat)] → Final=[0.000, 1.000, 0.000] → PWM=[  0, 254,   0]
Window 19: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.00(heartbeat) M:0.06(zigzag) H:0.00(beat)] → Final=[0.000, 1.000, 0.000] → PWM=[  0, 254,   0]
Window 50: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.00(heartbeat) M:0.27(zigzag) H:0.00(beat)] → Final=[0.000, 1.000, 0.000] → PWM=[  0, 254,   0]
Window 100: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.00(heartbeat) M:0.03(zigzag) H:0.00(beat)] → Final=[0.000, 1.000, 0.000] → PWM=[  0, 254,   0
]
Progress: 0.4% (100/25598 windows, 1.0s) - L=  0 M=254 H=  0
Window 150: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.00(heartbeat) M:0.27(zigzag) H:0.00(beat)] → Final=[0.000, 1.000, 0.000] → PWM=[  0, 254,   0
]
Window 200: Inference=[114.000, 101.000, 85.000] Patterns=[L:0.00(heartbeat) M:0.03(zigzag) H:0.00(beat)] → Final=[0.000, 1.000, 0.000] → PWM=[  0, 254,   0
```

Result:

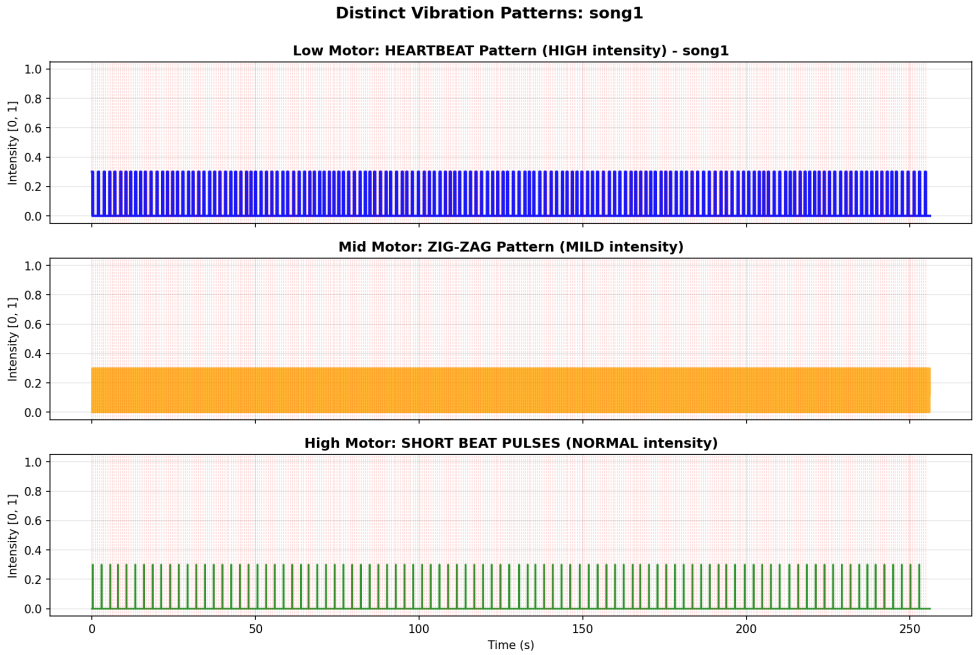
The whole song:



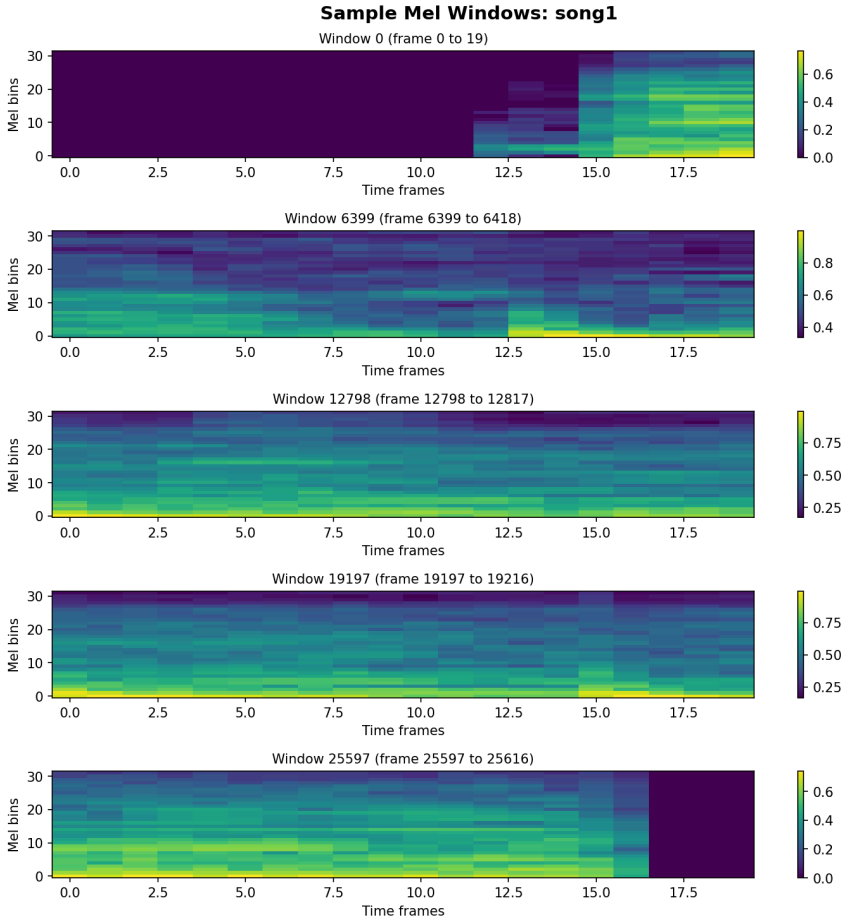
A small sample window:



Distinct Patterns:



Mel Windows of the selected song:



Conclusion:

Haptica successfully implements a depthwise-separable CNN on an Arduino Nano that translates music into haptic patterns in real time. The system processes audio through mel-spectrogram analysis, extracts musical features, and generates three distinct vibration patterns (heartbeat, zigzag, and beat pulses) synchronized with the music. The optimized TensorFlow Lite model runs efficiently on embedded hardware, processing audio at approximately 100 Hz. The prototype demonstrates that real-time audio-to-haptic translation is feasible on resource-constrained devices, enabling tactile perception of musical dynamics, transitions, and rhythmic elements.