MUSE: Mobile User-Sensitive musical Expression

E6692.2022Spring.MUSE.report.lm3963
Liam McHugh, lm3963
Columbia University

Abstract

MUSE transforms everyday activity into continuous, context-appropriate music by coupling an on-device activity‑understanding encoder with a cloud-hosted MusicGen [4] decoder. An mobile-class edge device ingests wrist-worn inertial and heart-rate signals, encodes the user's current activity into interpretable semantics, and injects encodings into prompts sent to remote MusicGen-small model workers. A high-speed bi-directional WebSocket pipeline streams autoregressively generated audio fragments back to the client as they're generated , where an asynchronous mixer stitches overlapping chunks for gap-free playback. The original semester goal of edge-situated ML is met with the Jetson Nano utilized as a stand-in for modern wearable mobile computers(eg Apple Watch[7]); quantitative signal encoder accuracy and end-to-end streaming fall within range of project goals, enabling flexible activity-sensitive musical expression as a step towards a new paradigm of human-computer interaction. Remaining challenges involve prompt-to-prompt timbral smoothing and streaming latency improvements.

Keywords: Entrainment, MusicGen, SigLIP, Pulse Code Modulation (PCM), Mel Spectrogram, Websocket

1. Introduction

Music consumption is largely passive: playlists, radio-style recommendation and skip-based control. Yet cognitive-affective research shows rhythmic entrainment and semantic congruence between music and physical state enhance performance, mood and immersion.

Meanwhile, parallel computing capabilities have massively expanded in the age of AI (AlexNet in 2012). Figure 1a displays a histogram of GPU capability improvements; speedups and storage both continue to massively increase. Similarly, mobile device parallel processing storage has followed the same trend, enabling advanced local machine learning tasks at the human-machine interface. [12]

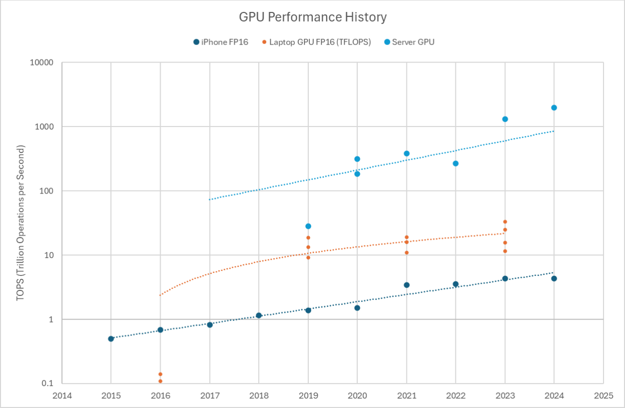


Figure 1.1a: Performance History (half-precision TOPS) of Graphics Processing Units used in Deep Learning

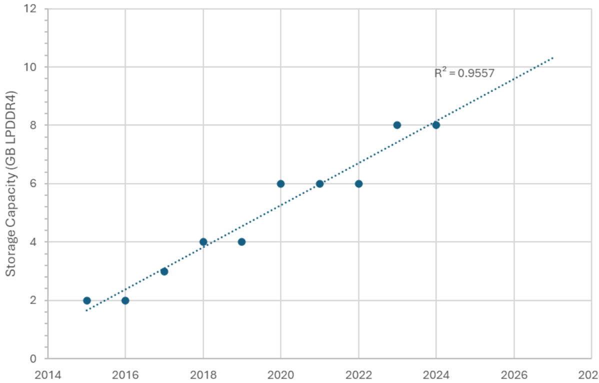


Figure 1.1b: iPhone GPU Storage History

MUSE forms a **closed loop** between **physiological activity** and **live generative audio**, enabling preconditioned "be your own soundtrack" experiences for fitness, productivity and rehabilitation applications.

Technical hurdles include (i) low-latency on-device activity understanding, (ii) high-quality audio generation under compute & bandwidth limits, and (iii) seamless streaming with inaudible transitions. We address these by splitting computation: a lightweight CNN/SigLIP2 encoder runs on the edge device (modeled with Jetson Nano), while a GPU VM hosts MusicGen. A proxy-relay architecture decouples unreliable wireless links from the real-time audio stream.

2. Summary of Prior Development

Although no single paper proves out a complete paradigm shift at this capacity, three seminal works underpin MUSE generative streaming technology:

2.1 Fundamental Technology Developments

MusicGen (Meta/Facebook, 2023) - an autoregressive transformer that conditions 32 kHz waveform generation on text prompts via an EnCodec tokenizer and a T5-style text encoder. MusicGen-small consists of 256M parameters & scores well on opinion-based quality benchmarks, with larger models also available. The system is an autoregressive generator, so is a prime option for fragment streaming. [4]

Musicgen-Streaming [5] (Gandhi & HuggingFace, 2024) - Introduced Musicgen-streaming, a lightweight iterator that yields playback-ready PCM every *play_steps* decoding iterations, achieving sub-second first-token latency without modifying model weights or internal structure. We adopt the same fragment assembly logic (token cache → delay-pattern mask → PCM) and modify it for asynchronous WebSocket delivery and dynamic *play_steps* tuning on our GPU VM.

SigLIP2 (Google Research, 2024) - a vision-language model that replaces softmax contrastive loss with a Sigmoid contrastive objective, improving embedding-space training at wide batch size varieties (useful for low-resource training systems). The work's open-source checkpoints and model size flexibility(90M-1B params) make it ideal for edge-cloud network prototyping. [6]

2.2 Key Results of the Original Papers

MusicGen-small achieves MOS ≈ 4.0 on the MAESTRO test-set while running in <8 GB VRAM. SigLIP2-base attains 78 % zero-shot top-1 on ImageNet-1K using 90 M params and fp16 weights. MusicGen-streaming reaches consistent (no lags) audio streaming from an H200-enabled VM with a wide variety of *play_steps*.

SigLIP (and simpler CNN-based systems) can effectively encode multi-channel signal data into semantic representations.

These results demonstrate that small models suffice for high-quality audio generation and multimodal semantic alignment under real-time application constraints.

3. Methodology

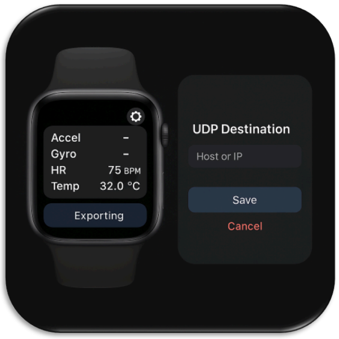
3.1. Objectives and Technical Challenges

Objective	Target	Challenge & Mitigation
Real-time activity encoding on Edge Device (Jetson Nano)	Semantic Output with variability between planned activities	CNN w/0.5M params SigLIP w/MusicGen embedding latents
Minimize stream lags	Set up workers to build up queue until new-prompt generation complete	Local vs VM GPU; Cloud offloading, "play-steps" modif. in MusicGen streamer
Seamless Prompt Transitions	Opinion-Eval: fine-grained, subtle shifts	Overlapping, Embedding Encoder, Context provision

3.2 High-Level Development Methodology

MUSE is developed as a user-oriented product. Simple utility and smooth experience is prioritized in long-term implementation & architecture/network scaling.

Testing Data for this study is obtained from UC Irvine's PPG+Dalia wearable sensor dataset, described further in implementation. Long-term test data should be acquired through utility-oriented wearable & mobile devices.



Accel –
Gyro –
HR 75 BPM
Temp 32.0 °C

Figure 3.2.1: WatchOS Data Acquisition GUI (simulated)

4. Implementation

In accordance with the e6692 course project requirements and timeline, this initial MUSE study uses an NVIDIA Jetson Nano to mimic a modern wearable computing device. Semantic data encoding, tested primarily with a custom lightweight CNN, is performed on the Jetson. Encoded information is injected into a custom prompt (with engineered base features for transition smoothing) & offloaded to a high-capacity Virtual Machine for music generation. With generation & streaming rates higher than real-time consumption, a prompt-specific music fragment queue is built up & drawn upon for music on the local player device. Sorted fragments are played in real-time & saved for qualitative analysis.

4.1 Utility Data

Dynamic multi-modal datastream in MUSE utility is to be collected from the wearable mobile device. Dynamic data in this project includes three-dimensional accelerometer data and heart-rate. Scripting is established to pull data from Apple WatchOS (GUI in Fig. 3.1) to stream to local machine, but personal data collection & usage is out of scope of this study. Static Data is limited to age, sex, height & weight in this study.

Train & Test Dataset: PPG-Dalia wrist-worn multimodal dataset — 128 k s across 15 subjects, 9 annotated activities (walking, cycling outdoors, climbing stairs, working at desk, lunch break, driving a car, sitting and reading, transition).

Pre-processing: 8-second samples of 3-dim accelerometer data are resampled to 64 Hz and translated to 64×64 log-mel spectrograms (Fig. 4.1a/b) via AccelToRGBMel utility, then augmented with sliding windows (stride = 2 s) for a total of 65k training frames. Heart Rate samples are taken per-frame in this study, but heart-rate context may also be provided.

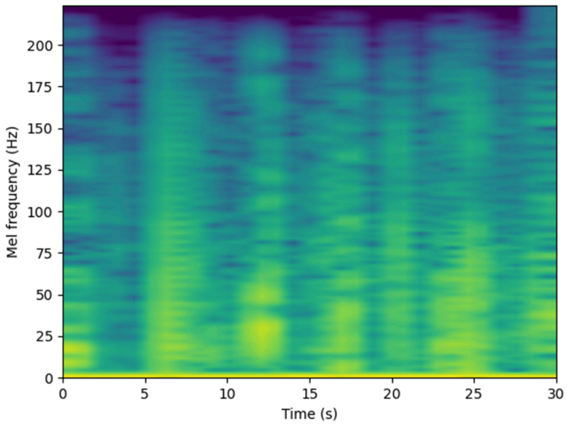


Figure 4.1a: Single-Channel Mel Spectrogram (Colorized) Encompassing 30s of Frequency - Mel Power Data.

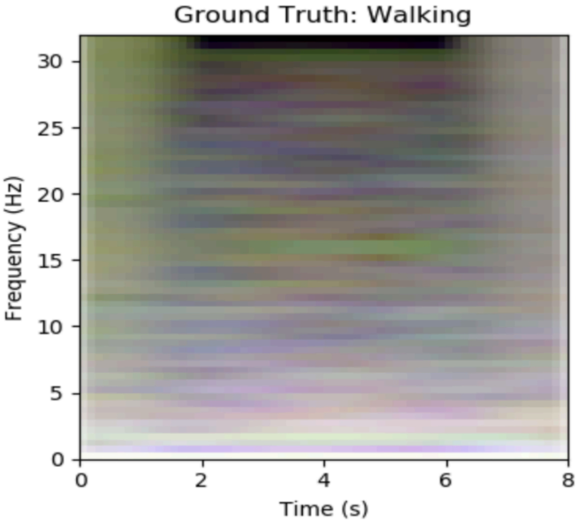


Figure 4.1b: RGB-Channelled Mel Spectrogram of Accel. 8-sec frames used as input to multimodal data encoders.

4.2 Deep Learning Architectures

This study employs a signal-semantic encoder(used primarily as classification) to prompt the pretrained MusicGen system. Encoder training is performed on a local machine w/RTX 4070 gpu (8GB gpu memory, 15.6 fp16 TOPS). Train/val split is 80 / 20, stratified by subject.

The primary data encoder used in this study is a lightweight Convolutional Neural Network (Fig. 4.2.1) which incorporates multichannel accelerometer spectrogram maps and pseudo-static data (per-frame heart-rate + indiv. statics). Three standard convolutional blocks process frame spectrograms and feed to a series of two shared linear layers, totalling only 500k parameters. Outputs are trained with conventional Cross-Entropy loss (Results Fig. 5.1.1).

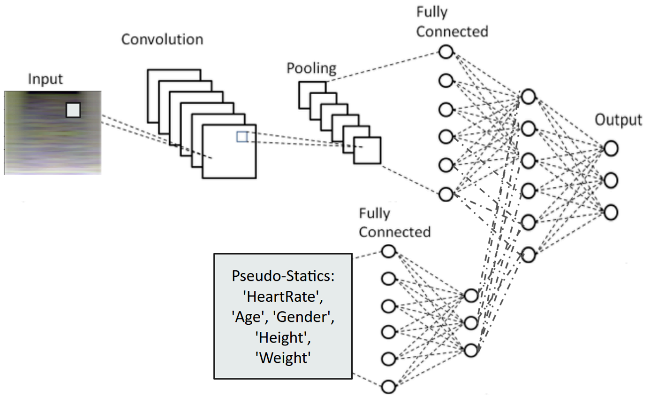

Input
Convolution
Pooling
Fully Connected
Output
Fully Connected
Pseudo-Statics: 'HeartRate', 'Age', 'Gender', 'Height', 'Weight'

Figure 4.2.1: Convolutional Deep-Learning Classification Encoder. Pseudo-static data fed into intermediate FC layers. Output classes correspond to activity semantics.

SigLIP is also proposed as a longer-term multimodal encoder(Fig. 4.2.2). MusicGen embeds semantic prompts into 768-dimensional Flan T5 text embeddings; as a step towards a signal-MusicGen integrated encoder with signal encoder embeddings as direct MusicGen inputs, we train SigLIP on annotation class T5 embeddings. This system likely does not zero-shot generalize beyond the class tokens due to the narrowness of training spread, but future systems will utilize such implementations with more varied training sets. See Results Fig.   5.1.2 for initial training.

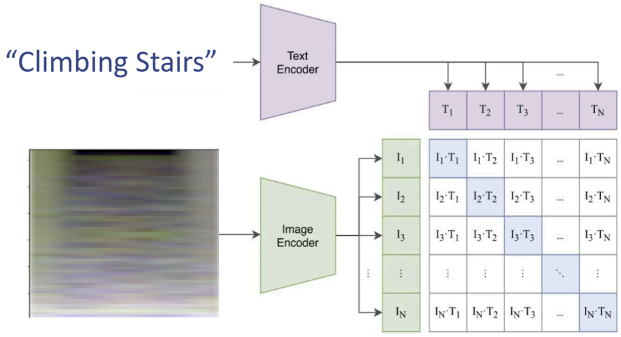


Figure 3.3.2: SigLIP Comparative Paradigm. Sigmoid losses provide classification-style rewards, helping converge training over a variety of batch sizes [6].

4.3 Compute Network & Streaming Design

On the edge device, data is encoded into semantics and sent to the local "player/client" device (laptop in testing), where it's immediately injected into user-customized stylistic semantic prompts and sent into the cloud(GCP). Customization details in Appendix Fig. A.2.1.

Example Prompt: "*Chill techno music. Current activity: Working at Desk. The listener is 34 years old, 182 cm tall, and weighs 78.0 kg. Heart rate is 64 bpm. Compose music that reflects the listeners style and physiological state. The track should begin with a seamless lead-in, evolving naturally from previous activity and ending naturally for the next activity.*"

New prompts are accepted on the Virtual Machine (Figure 4.3.2), triggering a MusicGen worker deployment to generate successive music fragments until the following prompt's worker is deployed. These fragments are placed in a streaming queue and sent back to a local "player/client" machine, where the queue is parsed & chronological music is played. As discussed in Results, music may have lags due to unsatisfactory MusicGen inference throughput; stitched chronological music is also saved to a .wav file.

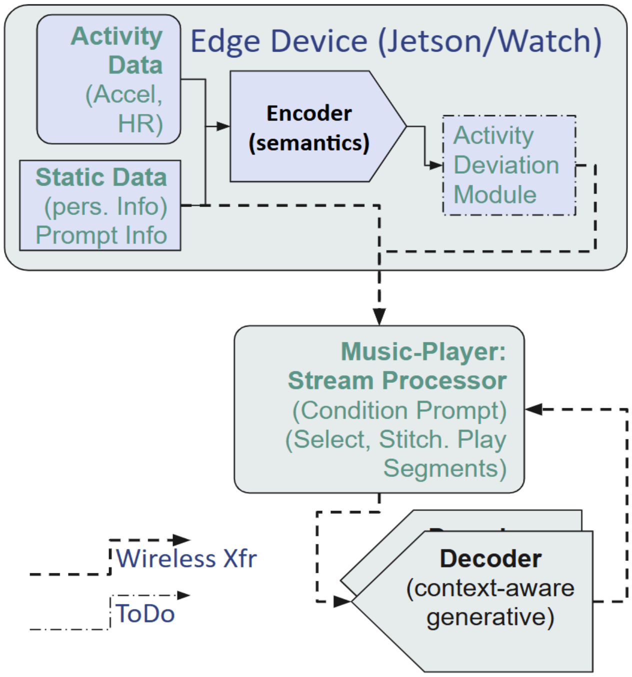

Edge Device (Jetson/Watch)
Activity Data (Accel, HR)
Static Data (pers. Info) Prompt Info
Encoder (semantics)
Activity Deviation Module
Music-Player: Stream Processor (Condition Prompt) (Select, Stitch. Play Segments)
Decoder (context-aware generative)
Wireless Xfr
ToDo

Figure 4.3.1: High-Level MUSE Network Diagram

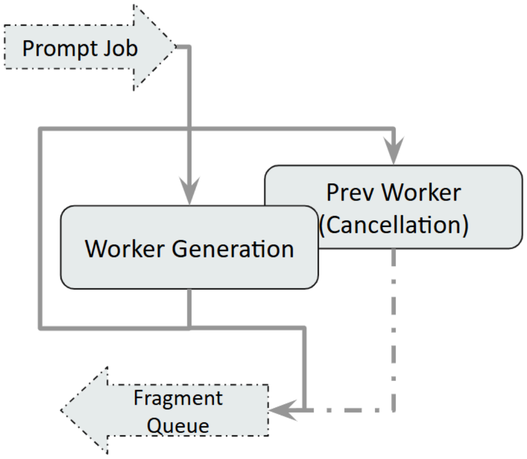


Figure 4.3.2: Virtual Machine Workflow

5. Development Results

5.1 Encoder Training & Deployment

Custom CNN classifier training (Figure 5.1.1) is very successful, reaching ~85% accuracy within 25 training epochs. SigLIP training (Figure 5.1.2) quickly converges to a steady loss, but shows no marked improvement beyond the first few training batches.

Both the custom CNN and SigLIP are trained as proof of concept, but only the CNN is deployed on the Jetson for this study: SigLIP is useful as a proven system for generating class embeddings; in a high-frequency prompting pipeline (utilizing base prompt embeddings as opposed to text semantics), this system would likely provide smoother transitions between MusicGen deployment segments. Without modifying the forward pass of MusicGen's base-code, however, our scope is limited such that the lightweight, highly-performant CNN classifier is used.

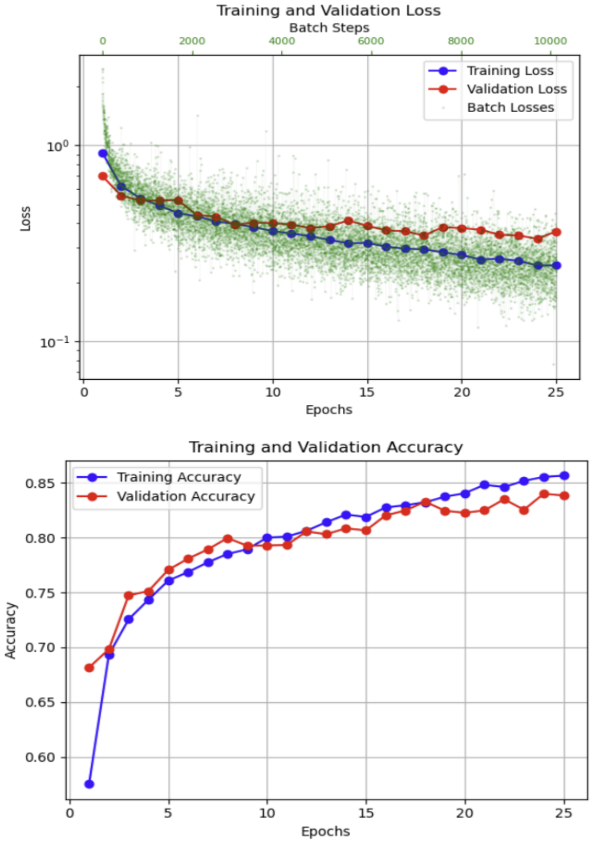


Figure 5.1.1 a/b: CNN Training Plots
Upper: Cross-Entropy Loss. Lower: Classfn. Accuracy

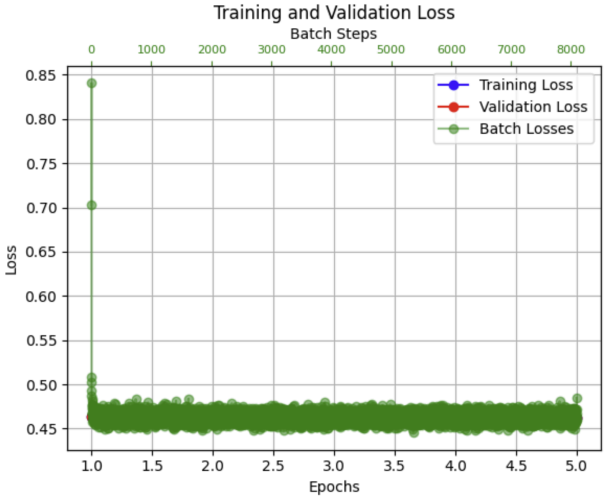


Figure 5.1.2: SigLIP Initial Training (Sigmoid Loss). System quickly converges to consistent loss over 9 classes; unused for this study due to added evaluation complexity w/out embedding-input justification.

5.2 MUSE Realtime Deployment

Qualitatively, the current MUSE implementation demonstrates promising results. Individual generation passes are largely(>90%) good-quality, but transitions provide a challenge amidst the standard MusicGen forward-pass. Even with very similar prompts, including fadein/fadeout commands, pronounced timbral jumps occur between consecutive segments, exposing the need for further work on embedded prompt smoothing & context provision. In short, the prototype proves feasibility on legacy GPUs yet highlights a trade-off between hardware spend and perceptual polish that must be weighed against the addressable market.

Running MusicGen-small in fp16 on an NVIDIA V100 with 0.5-2.5-second fragments yields workable but slightly laggy performance (~10% fragment lengths). Below 0.35sec fragment lengths, communication lags begin to dominate. Newly-prompted audio arrives about 1.5 fragment-lengths after the edge sends the prompt, a delay that can be compensatorily reduced with fragment length and covered by built-in queue buildup/runout on A100/H100-class VM hardware. Lags are acceptable for testing, where stitched data can be saved for annotation, but is unacceptable for casual listening. Aside from VM compute power-ups, samples may also be globally stretched (~0.8x slomo) to cover lags and prepare a short queue for prompt switches.

5.3 Development Challenges

- MusicGen Location: Runs on local machine, but slow, low scalability. Forced onto VM.
- MusicGen input: requires text, doesn't take context without extensive under-the-hood work. Limited scope to semantic classification prompt injection.
- Classifier Inference: Jetson Nano locally runs Python 3.6 and comparatively outdated versions of Pytorch/Cuda, so pipeline adaptations must be carefully made to avoid offline training to Jetson inference congruence.
- Prompt Engineering: Transition disparity effects can be partially smoothed with careful lead-in/out commands & transition timing alignment. See appendix A.2.1 for detailed prompt info.
- Network Streaming: Three-node streaming proves challenging for queue / worker management. Without effective queue & worker dynamics, asynchronous threads produce disordered queue on transitions. Streaming system centrally facilitated by local streaming client (music player).

5.4 Discussion / Insights Gained

MUSE generative streaming smoothness and latency minimization are complex tasks, but where gaps exist, so do clear paths forward. Through CNN & initial SigLIP training results, smooth activity-sensitive semantic prompting has proven to be a manageable goal, but even with careful base prompt engineering, affecting smooth transitions will likely require forward-pass modification to MusicGen's decoder context tokens to accept previous-segment context tokens.

6. Future Work **(Ordered by Priority)**

- **Smooth Activity Transitions**: previous-context provision to token-audio decoder (MusicGen fork)
- **Latency Minimization**: Increase VM compute power (ex streaming app: H200 w/800 fp16 TOPS! [5]). Investigate communication costs & network variants.
- **Product Design**: Value vs Cost Study, Addressable Markets & Technology Orientation. Monetization.
- Build out **Data Acquisition** w/target device (Fig. 6.1)
- Develop SigLIP variational inference: train w/acquired data, introduce synthetic chaos in annotations.
- **Fine-grained sensitivity**: Encoder+prompt embedding input as opposed to text prompt (large-scale SigLIP training, MusicGen fork): Smooth encoder embedding changes (as opposed to stark transitions in activity classifications) will produce more subtle differences in MusicGen passes while maintaining interpretability.
- **Personalized fine-tuning**: engagement/feedback RL
- Activity deviance detection & query initiation (Stat. / DL frameworks) for large-scale streaming

7. Conclusion

Development progress on MUSE validates the feasibility of real-time, activity-aware music generation on a split edge–cloud stack. This study effectively meets the primary academic study goals: A custom 0.5M-parameter CNN running on a Jetson Nano reliably classifies nine wrist-sensor activities at ~85 % macro-F1, prompting MusicGen-small to stream activity-sensitive music into a local device, limiting new-activity to new-music latency to roughly 1.5s on a V100 (likely below 0.75s on newer A100/H100 GPUs). The system is structured to deliver uninterrupted playback through queue management & overlap-additive mixing, promoting nimble edge-situated ML-driven activity entrainment. Although limitations to transition smoothness and stream lags currently exist, paths forward on both fronts have been identified to guarantee that MUSE will usher in a new paradigm of user-sensitive musical expression.

6. Acknowledgement

Special thanks to Professor Zoran Kostic, Devika Gumaste, & Ian Li for their guidance & material support throughout study development.

Thanks to the Meta Research group and Hugginface team (especially Sanchit Gandhi, now at Mistral) for publicizing their research and promoting AI accessibility.

7. References

[1] L. McHugh, “Project report,” Google Docs, 13 May 2025. [Online]. Available: https://docs.google.com/document/d/1zrKnF0ciJ9SkFMEfH6NLzc4ACPADLBZMTX92izYoTfI . Accessed: 13 May 2025.

[2] L. McHugh, “Project presentation,” Google Slides, 13 May 2025. [Online]. Available: https://docs.google.com/presentation/d/1A4nk4Ospce7u2O_nVVZyAI0OlnZDigZH9pl_1U1-LgY . Accessed: 13 May 2025.

[3] L. McHugh, “Project proposal document,” Google Docs, 2025. [Online]. Available: https://docs.google.com/document/d/1ysuf-gNWOS9CF6A7tQjX72crqGVqCAQJkAoN8FW9xHg . Accessed: 13 May 2025.

[4] Facebook Research, “MusicGen model card,” GitHub repository, 2023. [Online]. Available: https://github.com/facebookresearch/audiocraft/blob/main/model_cards/MUSICGEN_MODEL_CARD.md . Accessed: 13 May 2025.

[5] S. Gandhi, “MusicGen streaming app,” Hugging Face Spaces, 2024. [Online]. Available: https://huggingface.co/spaces/sanchit-gandhi/musicgen-streaming . Accessed: 13 May 2025.

[6] Google Research, “SigLIP2 zero-shot image classification,” Hugging Face Collections, 2024. [Online]. Available: https://huggingface.co/collections/google/siglip2-67b5dcef38c175486e240107 . Accessed: 13 May 2025.

[7] Apple Inc., “Core ML.” [Online]. Available: https://developer.apple.com/documentation/coreml . Accessed: January 2025.

[8] u/spotify community, “Spotify’s AI playlist generator,” Reddit, forum post, 2025. [Online]. Available: https://www.reddit.com/r/spotify/comments/1fxxboq/lets_talk_about_ai_playlists/ . Accessed: 13 May 2025.

[9] Data Science Dojo, “State of generative music and audio-generative architectures,” blog article, 2024. [Online]. Available: https://datasciencedojo.com/blog/5-ai-music-generation-models/ . Accessed: February 2025.

[10] J. DiCarlo et al., “Work on human visual-perceptive control,” OpenReview, preprint, 2023. [Online]. Available: https://openreview.net/pdf?id=5GmTI4LNqX . Accessed: January 2025.

[11] MIT Media Lab, “Semantic synth programming,” research publication, 2023. [Online]. Available: https://www.media.mit.edu/publications/ctag-neurips/ . Accessed: January 2025.

[12] A. Jafari, M. Khalighinejad, and N. Mesgarani, “BCI-augmented attentional audio control,” *Adv. Sci.*, vol. 11, no. 9, p. 2401379, 2024. [Online]. Available: https://advanced.onlinelibrary.wiley.com/doi/full/10.1002/advs.202401379 . Accessed: January 2025.

8. Appendix

8.1 Individual Student Contribution

Liam McHugh (lm3963) completed all work individually.

8.2 Support Material

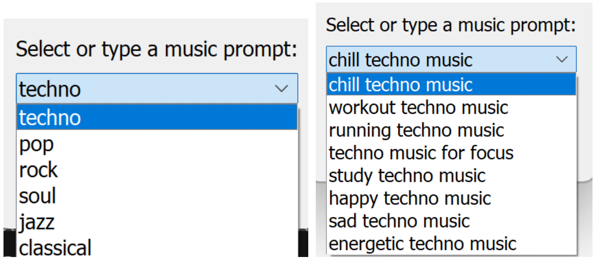
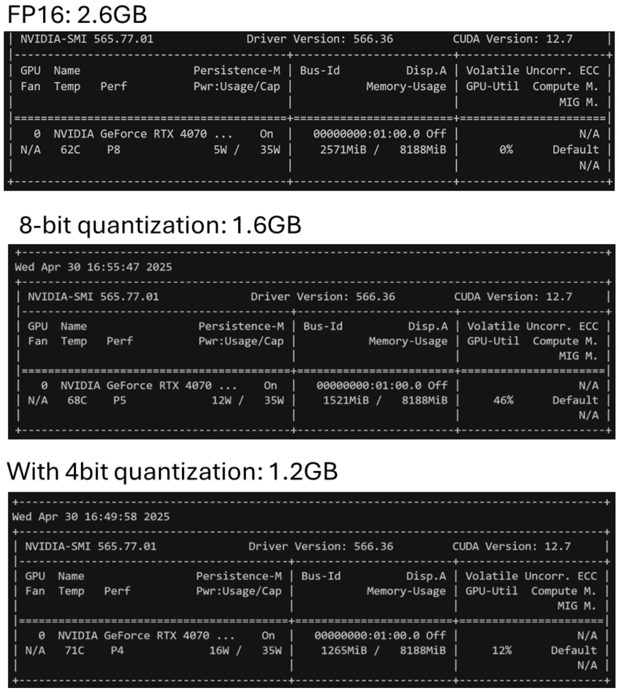


Figure A.2.1 Prompt Style Customizations.

A.2.2 Complete Example Prompt:
"Chill techno music. Current activity: Working at Desk. The listener is 34 years old, 182 cm tall, and weighs 78.0 kg. Heart rate is 64 bpm. Compose music that reflects the listeners style and physiological state. The track should begin with a seamless lead-in, evolving naturally from previous activity and ending naturally for the next activity." See Appendix Fig. A.2.1 for customization details



FP16: 2.6GB

```
| NVIDIA-SMI 565.77.01              Driver Version: 566.36         CUDA Version: 12.7     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 4070 ...    On  |   00000000:01:00.0 Off |                  N/A |
| N/A   62C    P8              5W /   35W |    2571MiB /   8188MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

8-bit quantization: 1.6GB

```
+-----------------------------------------------------------------------------------------+
Wed Apr 30 16:55:47 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 565.77.01              Driver Version: 566.36         CUDA Version: 12.7     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 4070 ...    On  |   00000000:01:00.0 Off |                  N/A |
| N/A   68C    P5             12W /   35W |    1521MiB /   8188MiB |     46%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

With 4bit quantization: 1.2GB

```
+-----------------------------------------------------------------------------------------+
Wed Apr 30 16:49:58 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 565.77.01              Driver Version: 566.36         CUDA Version: 12.7     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 4070 ...    On  |   00000000:01:00.0 Off |                  N/A |
| N/A   71C    P4             16W /   35W |    1265MiB /   8188MiB |     12%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

Figure A.2.3 MusicGen Quantization & Storage Reqs