Learning Music Similarity Embeddings for MECOMP

Anthony Rubick, Beyza Ispir, Joseph Berglund - Rice University

COMP/ELEC 576 - Introduction to Deep Learning - 2025-2026

Abstract

We wanted to build a deep learning model that learns vector embeddings for songs. These embeddings should match how humans think two songs sound similar. The goal is to replace or improve the handcrafted features inside MECOMP, a music player and recommendation tool written in Rust. We will use short audio clips
from Creative Commons songs and human similarity triplets for training. The final model should be small, fast, and able to process a local music library quickly on a normal laptop.

Background and Motivation

Traditional:
Audio → Classifier
Fixed-size
(not for similarity)

→

Our Approach:
Audio → Embedding
Variable-length
(optimized for similarity)

Problem Statement

Music recommendation systems need good embeddings to find similar songs, but current methods have some big problems:
- Most embeddings (like wav2vec) are built for classifying individual songs, not comparing them to each other
- They often need fixed-size inputs, which doesn't work well for songs of different lengths
- Some models are trained on just one genre, so they don't work across different types of music
- Previous work (the Bliss project) trains a distance metric on handcrafted features, we train a model to generate those embeddings instead.

Why This Matters

Recommendation is super important for music apps, but right now the methods miss what makes songs feel similar to humans. We're training embeddings specifically to capture how songs compare to each other, so recommendations can be based on how music actually "feels" rather than just metadata tags.

What Makes This New

This is a fresh take on music similarity with deep learning:
1. We train embeddings directly for similarity (not classification)
2. We use human judgments with triplet loss to match how people actually hear music
3. Our model handles variable-length audio, which is what you get in real music libraries
4. It's designed to work with MECOMP, an existing Rust-based music player

Dataset

We used the Free Music Archive (FMA) Small Dataset:
- 8,000 songs covering evenly distributed across 8 genres
- 8,000 synthetic triplets we generated using genre info
- 2,000 triplets for human annotation (60 fully labeled)
- Audio format: 22.05 kHz mono, clips are 30 seconds long (though to save memory during stage 1 training, only the first 10-20 seconds are used)

Model

Raw Audio
(Var. Length)

Variable-length audio input

Frame Extract
(2048/frame)

Split into 2048-sample frames

CNN
(1→32→64)

Extract features with depthwise CNN

GRU
(2×128)

Capture temporal patterns

Attention

Focus on key segments

32-D Embed

Output 32-dimensional vector

AudioEmbeddingTiny: Lightweight CNN-GRU with attention pooling.
Processes variable-length audio through depthwise separable convolutions (1→32→64),
2-layer GRU (128 dim), attention mechanism, and projects to 32-D embedding.
Total: 213K parameters - compact for deployment.

Experimentation

Stage 1: Synthetic
8K triplets
LR: 1e-3

→

Stage 2: Fine-tune
60 triplets
LR: 1e-4
Freeze CNN

Training Strategy:

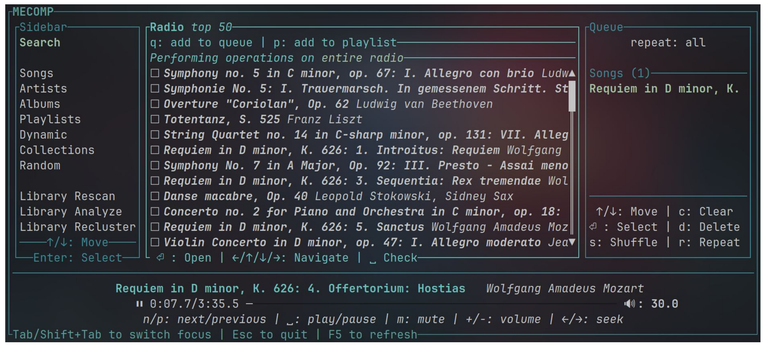
• Two-stage training: First trained on 8,000 synthetic triplets, then fine-tuned on 60 human-annotated triplets
• Triplet margin loss (margin=0.2) to learn similarity relationships
• Learning rate scheduling: ReduceLROnPlateau (1e-3 for synthetic, 1e-4 for fine-tuning)
• Freeze the CNN during fine-tuning
• Early stopping with patience=10 to prevent overfitting
  • Only the best model from each stage makes it to the next.

What We Tried:

✓ AudioEmbeddingTiny: Depthwise CNN + GRU + Attention Pooling (213K params) - FINAL
✓ Two-stage training approach: lets us get the most out of our limited human triplets
✓ Different hyperparameters (frame size, embedding dim, batch size)
✓ Smaller hidden state (256 -> 128): improved performance and cut parameters from ~800k to ~200k
✗ AudioEmbeddingGRU: Standard CNN + GRU (larger, less efficient)
✗ Single-stage training (poor generalization)
✗ Training only on human data (overfitting)

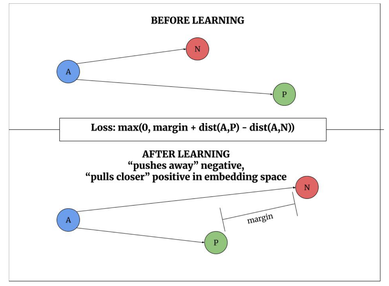
Hyperparameters: Frame=2048 | Embed=32-D | Batch=12→4 | GRU=2×128 | Loss=margin 0.2

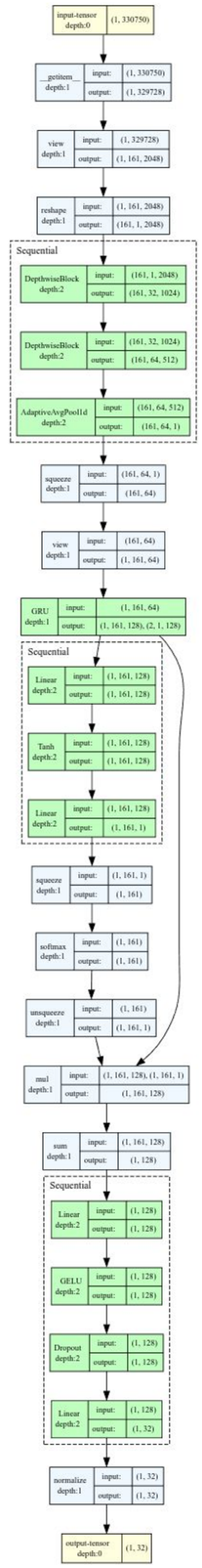
Results



The model achieved 75% test accuracy after fine-tuning with human-annotated triplets, showing that even when you don't have enough quality data to train a model directly, you can still get good results with well crafted synthetic data and a bit of fine-tuning.

This model will allow for significant improvements to MECOMPs recommendation system, realizing the dream of high-quality offline song recommendations.





Conclusion and Future Work

We showed that a CNN-GRU model with attention pooling can learn good music embeddings for similarity. The two-stage training (first on synthetic data, then fine-tuning on human labels) worked well - it connected metadata-based similarity with how people actually hear music.
What we Learned:
1. Synthetic data worked great as a starting point: The 8,000 genre-based triplets gave the model a solid foundation for learning similarity
2. Human fine-tuning helped: Even with just 60 human-labeled triplets, we saw better alignment with human judgments (75% accuracy on human triplets)
3. Small embeddings are enough: 32 dimensions captured what we needed for similarity tasks
4. Ready to deploy: We exported to ONNX, so it can actually be used in production
In the future, we plan to complete the human annotated dataset for better performance.

Works Cited

1. Defferrard, M., Benzi, K., Vandergheynst, P., & Bresson, X. (2017). FMA: A dataset for music analysis. *18th International Society for Music Information Retrieval Conference*.
2. Bello, J. P., et al. (2019). *Bliss: A Music Similarity Metric Based on Human Perceptual Judgments*. Available at: https://lelele.io/thesis.pdf
3. Schneider, S., et al. (2019). wav2vec: Unsupervised Pre-training for Speech Recognition. *Interspeech 2019*.
4. Hermans, A., Beyer, L., & Leibe, B. (2017). In Defense of the Triplet Loss for Person Re-Identification. *arXiv preprint arXiv:1703.07737*.
5. Free Music Archive Dataset. (2017). *GitHub repository: mdeff/fma*. https://github.com/mdeff/fma
6. MECOMP Project. *GitHub repository: AnthonyMichaelTDM/mecomp*. https://github.com/AnthonyMichaelTDM/mecomp