Learning Music Similarity Embeddings for MECOMP Using Triplet Loss

Beyza Ispir
beyza.ispir@rice.edu

Anthony Rubick
ar312@rice.edu

Joseph Berglund
jb255@rice.edu

November 2025

Abstract

We want 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

Music recommendation systems often rely on metadata, tags, or user behavior. These methods work well for large platforms but do not always capture the true "feel" of a song. Content-based music information retrieval (MIR) instead focuses on features extracted directly from the audio signal, such as MFCCs, spectral features, and chroma, which describe timbre, harmony, and rhythm [1]. These features allow comparisons between songs even when metadata is missing or inconsistent.

The Free Music Archive (FMA) dataset is a widely used open collection of Creative Commons music and is commonly used for research on music analysis and MIR systems [3]. FMA provides enough diversity to study general audio similarity beyond fixed genre labels.

However, traditional MIR pipelines depend heavily on handcrafted features and simple distance measures. These methods often fail to capture higher-level musical structure, subjective similarity, and long-time patterns in songs [1]. The thesis also shows that human perception of similarity is complex and that low-level acoustic descriptors alone are not enough.

To model human perception more directly, several studies have used human similarity judgments. One common approach is the triangle (triplet) discrimination test, where a listener chooses which two out of three clips sound the most similar [1, 6]. Modern deep learning work also uses triplet-based training for music similarity and ranking, showing that this method can learn more accurate embeddings than classical MIR features [2, 5].

Our project follows these ideas but extends them with deep learning. Instead of relying only on handcrafted MIR features, we aim to learn audio embeddings directly from spectrograms using a CNN combined with an RNN. These learned embeddings can capture both short-time acoustic patterns and long-time structure. By training with human similarity triplets, the model can better match subjective musical relationships than traditional feature engineering approaches. The final embeddings will be used inside MECOMP, a local music recommendation tool.

Goal

The main goal of this project is to train a deep learning model that:

- learns audio embeddings that match human similarity,

- runs inside MECOMP in Rust,
- fits within 1–5 MB,
- can embed around 1000 songs in a 1–2 minutes on consumer hardware.

Dataset

We will use two types of data:

Audio (FMA Dataset):

We will use Creative Commons audio clips from the Free Music Archive. We will start with the small and medium subsets to make training manageable. All audio will be resampled to 16 kHz mono to match MECOMP's preprocessing.

Human Similarity Triplets:

We will collect short clips (20–30 seconds) and ask listeners to choose which two sound the most similar. From this choice, we can form anchor–positive–negative triplets. These triplets will be used to train the embedding model with triplet loss. If we have limited time, we will first use weak triplets from metadata (e.g., genre similarities) to start training.

Method

Our model will use two parts together: a CNN encoder and an RNN head. The goal is to learn both short-time patterns in audio and long-time structure across the whole song.

1. CNN Encoder

The input audio will be converted into log-mel spectrograms. We will use one-second windows. A small CNN will extract local features from each window. These features describe short patterns such as timbre, texture, rhythm, or repeated shapes in the audio.

2. RNN Head (GRU or LSTM)

The CNN will output a sequence of feature vectors. We will then pass this sequence into a small RNN, either a GRU or an LSTM. The RNN is used because songs are sequences and the order of sounds matters. The RNN will learn how different windows connect over time and will output one final embedding for the full audio clip.

3. Training with Triplet Loss

We will train the model using triplet loss. For each triplet (A, P, N), we compute embeddings for all three audio clips and apply a standard margin-based triplet loss. This loss makes the embedding of the anchor clip (A) close to the positive clip (P) and far from the negative clip (N). After training, the embeddings of similar songs should be close in the vector space.

4. Model Size and Efficiency

We will keep the CNN and RNN small so that the full model stays within 1–5 MB. This size allows fast inference inside MECOMP and supports analyzing around 1000 songs in a few minutes on a normal laptop.

Implementation Plan

Stage 1: Baseline (Week 1)

Prepare FMA audio, compute mel spectrograms, and create simple metadata-based triplets. Build a simple baseline model using current MECOMP features for comparison.

Stage 2: Prototype Embedding Model

Implement the CNN + GRU/LSTM model in Python. Train on weak triplets to confirm that the model runs correctly. Measure training loss and basic retrieval accuracy.

Stage 3: Human Triplet Collection

Build a small survey for collecting human similarity annotations. Collect and clean several thousand triplets. Retrain or fine-tune the model with these labels.

Stage 4: Rust Integration

Export the trained model to ONNX and load it in Rust using `burn` or `candle`. Measure speed and memory usage. Try to meet the target of processing 1000 songs within 1–2 minutes.

Stage 5: Final Evaluation and Report

Compare embeddings with the baseline, evaluate triplet accuracy, run small user tests inside MECOMP, and prepare the final write-up and poster.

Team Roles

All team members will share the main responsibilities equally. Each person will work on:

- searching, preparing, and cleaning audio data,
- creating training samples and triplets,
- building and training the deep learning models in Python,
- helping with evaluation and result analysis,
- supporting the Rust integration step and testing the final model.

We will divide tasks week by week based on workload, but everyone will contribute to data work, model training, and implementation.

Feasibility and Limitations

The project is feasible because FMA is public, and we will use small models that can run on a normal GPU. Triplet loss and audio CNNs are well-studied. MECOMP also gives a clear use case for music embeddings.

However, collecting high-quality human triplets takes time. Additionally, Rusts deep learning ecosystem is not very mature yet, so converting and running the model may require extra engineering effort. Meeting strict runtime goals for 1000 songs may require pruning or distillation. Due to time limits during the semester, the endpoint goal is to produce a working Python model and a basic Rust inference setup. Full optimization may become a topic for future work.

Potential Impact

A good embedding model can make MECOMP [4] give better recommendations without using external APIs or private user data. This is useful for people who keep local music collections or use rare or independent music. The method can also support playlist generation, cover song detection, and other music information retrieval tasks. The survey and training pipeline can help future students explore new architectures or loss functions for music similarity.

Helpful Links

1. MECOMP Project, GitHub Repository.
2. Free Music Archive.
3. Initial Proposal Document
4. Bliss-rs, GitHub Repository.

References

[1] Pierre Arzelier. Music similarity tool for contemporary music. Master's thesis, IRCAM, 2018.

[2] Joseph Cleveland et al. Content-based music similarity with triplet networks. *arXiv preprint arXiv:2010.06448*, 2020.

[3] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. In *International Society for Music Information Retrieval Conference*, 2017.

[4] Anthony Michael. Mecomp project. `https://github.com/AnthonyMichaelTDM/mecomp`. GitHub Repository, accessed 2025.

[5] Louis Prétet et al. Learning to rank music tracks using triplet loss. *arXiv preprint arXiv:2002.04650*, 2020.

[6] Xiaoyu Qi, Deshun Yang, and Xiaoou Chen. Audio feature learning with triplet-based embedding network. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4979–4980. AAAI Press, 2017.