Technical Report: New Wave

Team

Vladislav Kalinichenko – v.kalinichenko@innopolis.university
Polina Korobeinikova – p.korobeinikova@innopolis.university
Janna Ivanova – j.ivanova@innopolis.university

Repository

https://github.com/poinka/NewWave.git

Project Topic

Natural Language Music Recommender System (Music-by-Description Retrieval)

Artifacts

- Repository: https://github.com/poinka/NewWave.git
- iOS Client Repository: https://github.com/iJeannne/NewaveApp
- Server: https://github.com/poinka/NewWave/blob/main/api.py
- Datasets: MusicCaps, Song Describer Dataset (SDD), enrich-music4all, Song-Interpretation-Dataset
- Models: CLAP (fine-tuned), all-MiniLM-L6-v2 (deprecated), bge-m3, Fusion model
- Training infrastructure: A100 GPU, InnoDataHub servers

Methods

Dataset

The project uses multiple datasets with specific roles:

1. **MusicCaps (Google HF)** – 10-second audio clips with captions; approximately 5,300 of ~5,600 targeted clips successfully cached. Used for CLAP training and initial validation.
2. **Jamendo CC catalog** – Large-scale corpus (~1 TB with many songs) used for extended training and some evaluation batches.
3. **Song Describer Dataset (SDD)** – Small, higher-quality dataset used exclusively for evaluation to assess generalization under description style shifts.

4. **enrich-music4all** – Metadata and text set with 108,363 rows containing track_id, tag_list, pseudo_caption, title, artist_name, release, m4a_genres, and m4a_tags. Used for sampling hard negatives and text descriptions.

5. **Song-Interpretation-Dataset** – Links lyrics and user interpretations to music metadata; combined descriptions include pseudo captions and user interpretations.

6. **FMA (Free Music Archive) Large Dataset** - A large-scale dataset containing full-length audio tracks across a wide variety of genres. It served as the primary source of audio for our combined dataset, where tracks were aligned with textual data from the Song-Interpretation-Dataset using a fuzzy matching algorithm based on track titles and artist names.

7. **Test data for demo** - For demonstration and final evaluation, we selected a diverse subset of 1,000 songs from the existing LyricCovers 2.0 dataset. Using stratified sampling across languages and release years, we identified representative tracks and processed them through our custom pipeline: downloading high-quality audio via yt-dlp and scraping clean lyrics from Genius to ensure high-fidelity pairing for our demo application.

Data Preprocessing:

1. **Parquet-based Pseudo-streaming** (Jamendo)
To handle the massive scale of the Jamendo dataset (~1 TB), we implemented a custom parquet-by-parquet pseudo-streaming pipeline. Instead of loading the entire dataset into memory, the system loads individual parquet shards, processes them into batches for training, and immediately deletes the local files to free up storage space.

2. **Audio Extraction & Truncation** (MusicCaps, LyricCovers, Song Describer)
We applied specific audio processing pipelines depending on the dataset source and purpose:

- MusicCaps: Entries were downloaded and cut into fixed 10-second audio clips to align with the original CLAP pre-training objectives.
- LyricCovers 2.0 (Custom Subset) & Song Describer Dataset (SDD): For these evaluation datasets, we processed the audio using librosa (converting to mono, 48kHz). We applied a 30-second truncation (for LyricCovers) and standard duration handling (for SDD) to ensure consistent input lengths for our validation metrics and demo application, balancing context quality with inference speed.

3. **Fuzzy Matching & Alignment** (FMA_large + Song-Interpretation-Dataset)
To create a multimodal dataset with both audio and rich textual descriptions, we merged the FMA_large (audio source) and Song-Interpretation-Dataset (lyrics/description source). The alignment was performed using a fuzzy matching algorithm based on track titles and artist names, filtering for high-confidence matches to ensure data integrity despite the potential for noise.

4. **Text Tokenization & Truncation** (Enrich-Music4All & Song-Interpretation-Dataset)
For the text retrieval models (bi-encoders), we preprocessed text data from enrich-music4all and Song-Interpretation-Dataset by truncating lyrics and user descriptions to 8192 tokens. We specifically selected the bge-m3 model for our final architecture because of its ability to handle such long context windows, whereas all-MiniLM was used only as a preliminary baseline to validate the retrieval pipeline with shorter sequences.

Modelling

CLAP (Contrastive Language-Audio Pre-training)

Objective: Extend CLAP beyond its pretraining limits (10 s audio and 77-token text) to handle full-length music tracks.

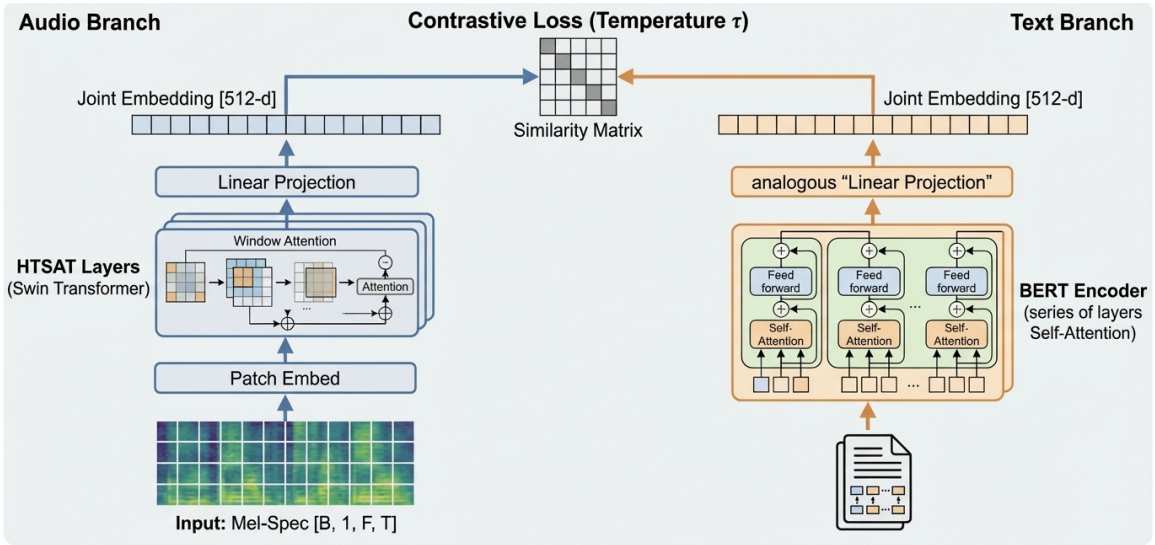
Approach:

- Applied linear schedule gradually increasing audio duration from 10 seconds toward 240 seconds.
- Expanded text descriptions beyond 77 tokens, approximately adding one second of audio per several batches.
- Trained on A100 for approximately one day, covering roughly 10% of the Jamendo dataset in a single pass.

Model Architecture:
[Architecture details – not provided in reports; skipping]

Training Details:

- Validation behavior varied by run: some runs showed stagnation while others produced sharp gains on Jamendo subsets and Sound Describer evaluation.
- Extended training ongoing to consolidate gains and improve generalization across description styles.



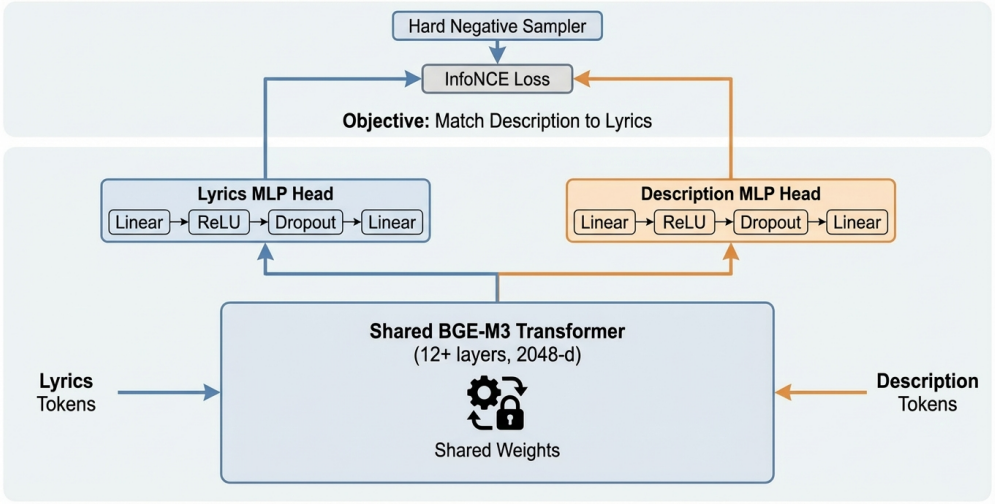
Lyrics/Description Retrieval Encoder

Model Iterations:

1. **Longformer (removed):**

- Failed to train; was very slow and consumed excessive memory.
- Lyric embeddings refreshed only every 2000 steps—inefficient.
- Literature review showed poor suitability for semantic retrieval.

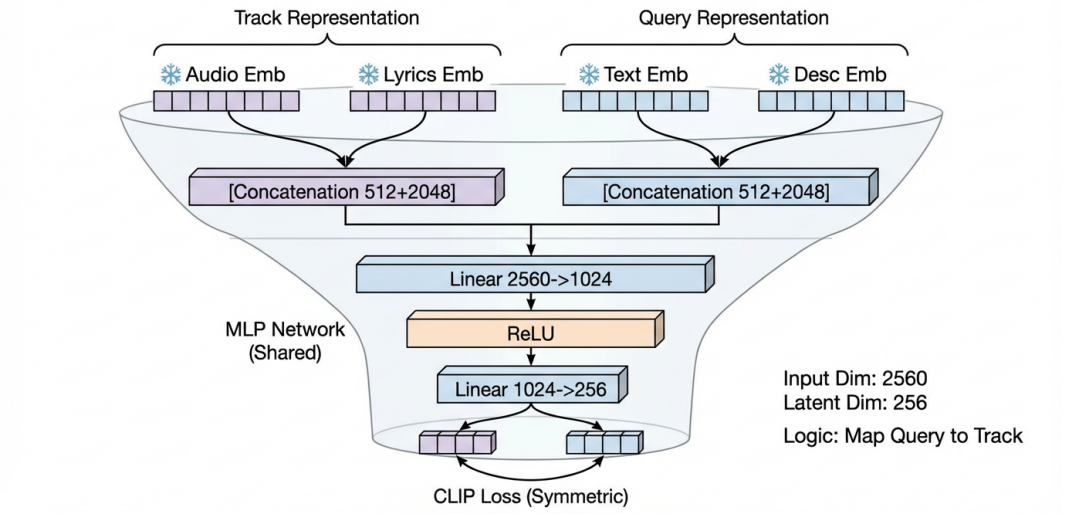
2. **all-MiniLM-L6-v2:**
   - Significantly faster training; enabled addition of hard negatives (by artist, then by tags).
   - 256-token truncation applied to both descriptions and lyrics.
   - **Results:** MRR = 0.0685; Recall@1 = 0.0288; Recall@5 = 0.0984; Recall@10 = 0.1479; Recall@20 = 0.2065
   - **Limitation:** Truncation discarded significant text content.
3. **Average Pooling over Full Text:**
   - Attempted to address truncation problem.
   - **Results:** Significantly worse: MRR = 0.0086; Recall@1 = 0.0016
   - Conclusion: did not pay off.
4. **bge-m3 (Current Best Model):**
   - Larger model capable of processing full description and lyrics.
   - Separate heads trained for lyrics and descriptions.
5. **Iteration a – Initial:**
   - MRR = 0.0871; Recall@1 = 0.0404; Recall@5 = 0.1258; Recall@10 = 0.1849; Recall@20 = 0.2499
6. **Iteration b – Increased Batch Size:**
   - MRR = 0.0932; Recall@1 = 0.0441; Recall@5 = 0.1325; Recall@10 = 0.1947; Recall@20 = 0.2641
7. **Iteration c – Extended to 25 Epochs (Best Results):**
   - MRR = 0.1169; Recall@1 = 0.0542; Recall@5 = 0.1734; Recall@10 = 0.2496; Recall@20 = 0.3335



Multimodal Fusion Model (Audio + Lyrics)

To leverage the complementary nature of audio acoustics and lyrical semantics, we developed a late-fusion neural network. While CLAP captures the "sound" (mood, instruments) and the Bi-encoder captures the "meaning" (lyrics, topics), this fusion model combines both signals into a unified representation.

- **Architecture:** The model consists of a Multi-Layer Perceptron (MLP). It takes two input vectors: the embedding from the fine-tuned CLAP model and the embedding from the Lyrics Bi-encoder. These vectors are concatenated and passed through the MLP layers to generate a single, joint embedding.
- **Training Data:** Training this fusion layer presented a unique challenge: no existing open-source dataset possessed both the raw audio signal and the high-level semantic interpretations we required. We had to construct our own. We started by isolating FMA_large, selected specifically for its massive repository of audio tracks, yet it lacked semantic depth. Conversely, we identified the Song-Interpretation-Dataset, which offered rich lyrical insights and user comments but contained no audio files. To bridge this gap, we engineered a complex matching pipeline. Using fuzzy string matching algorithms on track titles and artist names, we painstakingly aligned entries between these disparate sources.
- **Function:** Instead of relying on a simple heuristic average of two rankings, this model learns non-linear relationships between audio and text features. It effectively acts as a sophisticated ranker, producing a final score that balances musical style with lyrical content better than either modality could achieve alone.



System Architecture & Implementation

To transition from experimental notebooks to a functional product, we engineered a robust backend infrastructure and a native mobile client.

1. **Vector Database & Search Engine**
   We implemented a hybrid search system using FAISS and SQLite.

   - Vector Storage: We maintain two separate FAISS indexes for high-speed similarity search: one for audio embeddings (from CLAP) and one for lyrics embeddings (from Bi-encoder). Both indexes use L2 normalization to ensure cosine similarity equivalence.
   - Metadata Storage: A SQLite database stores structured song metadata (artist, title, album), file paths to MP3s and cover art, and links to the vector IDs.
   - Joint Search Algorithm: The system performs a set intersection between the top-k results from audio search and lyrics search, re-ranking candidates using our Fusion Model scores to return the most relevant tracks that satisfy both criteria.

2. **Backend API (FastAPI)**
The core logic is exposed via a RESTful API developed with FastAPI, featuring:

   - Endpoints: /search/audio, /search/lyrics, and /search/combined for flexible retrieval strategies.
   - Streaming: Implemented NDJSON (Newline Delimited JSON) streaming for the /search/combined endpoint. This allows the server to send metadata, cover art URLs, and audio streams progressively, significantly reducing perceived latency on the client side.
   - Infrastructure: The server is configured to run on a local network (0.0.0.0 host), allowing mobile devices on the same Wi-Fi to connect and stream high-quality audio directly from the server's file system.

3. **iOS Client Application**
We developed a native iOS application using Swift and SwiftUI.

   - Functionality: The app allows users to input natural language queries (e.g., "sad song about lost love") or record audio snippets.
   - Integration: It communicates with our FastAPI backend, parsing the streamed NDJSON responses to display search results in real-time.
   - Playback: The app features a built-in audio player that streams the MP3s directly from our backend, providing a seamless "Spotify-like" user experience for testing our retrieval quality.

Results

CLAP Metrics

Current Performance (on Song Describer evaluation set):

- MRR = 0.1122

Recall@K:

- Recall@1 = 0.0442
- Recall@5 = 0.1521
- Recall@10 = 0.2487
- Recall@20 = 0.3820

Precision@K:

- Precision@1 = 0.0442
- Precision@5 = 0.0304
- Precision@10 = 0.0249
- Precision@20 = 0.0191

SOTA Comparison:

- Against CLaMP 3 (2025), which reports MRR = 0.1985 on Song Describer: our best run achieves MRR = 0.1169, approximately 1.76× behind latest SOTA.
- CLAP 2023 baseline (same family as ours) reports MRR = 0.1131, supporting fairness of comparison.
- Our R@10 = 0.2487 is within 4.4% of CLAP 2023 figure (0.2601).
- Our R@1 = 0.0442 matches CLAP 2023 exactly, indicating competitive early-rank retrieval.

Lyrics/Description Encoder Metrics

Best Results (bge-m3, 25 epochs):

- MRR = 0.1169

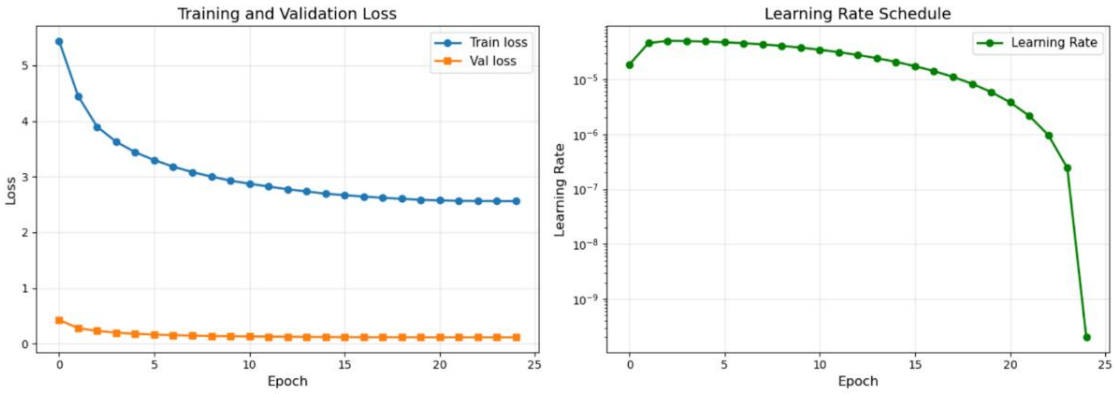
Recall@K:

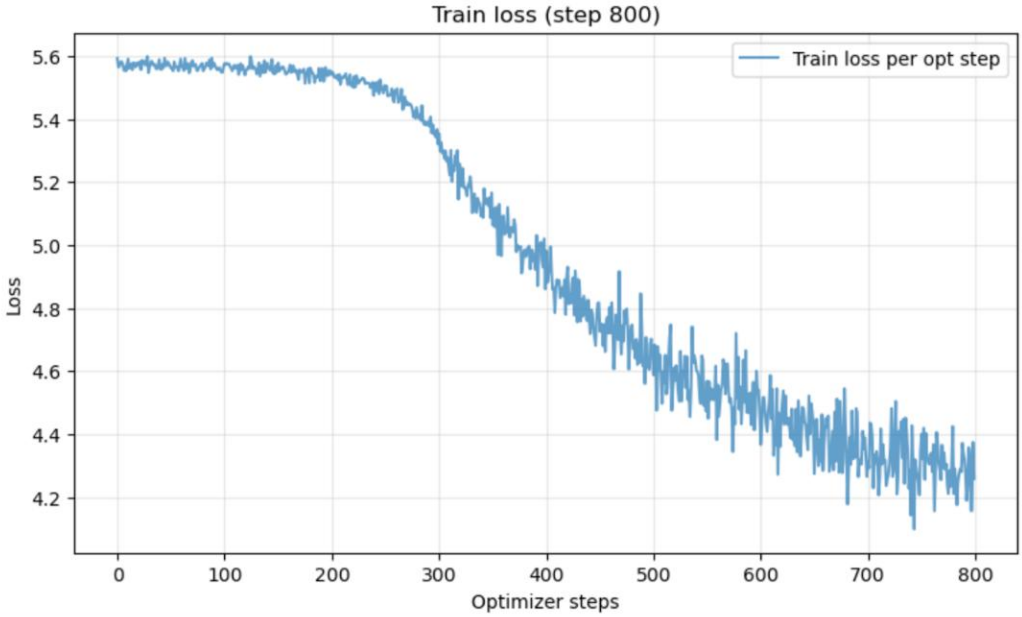
- Recall@1 = 0.0542
- Recall@5 = 0.1734
- Recall@10 = 0.2496
- Recall@20 = 0.3335

Precision@K:

- Precision@1 = 0.0542
- Precision@5 = 0.0347
- Precision@10 = 0.0250

- Precision@20 = 0.0167





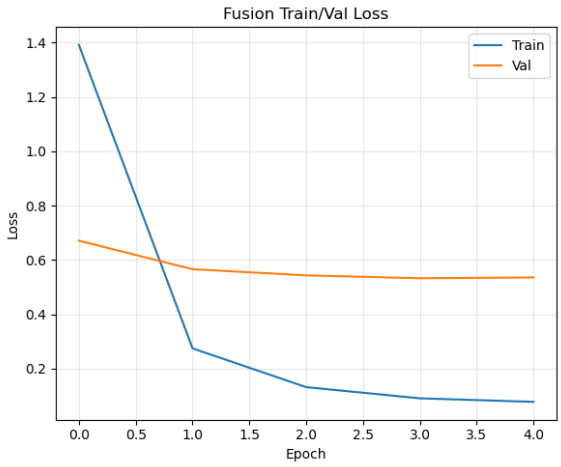
Multimodal Fusion Model Metrics

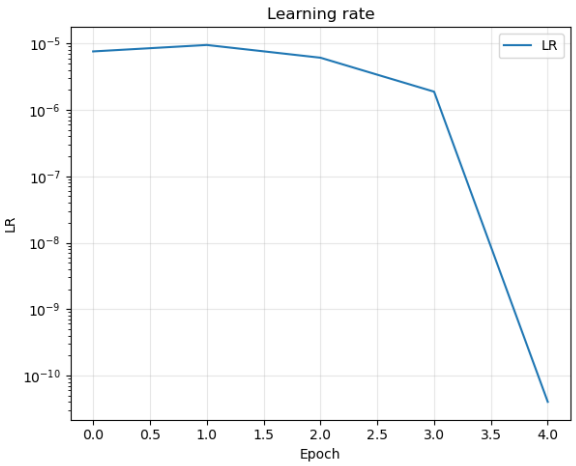
We evaluated our final fusion model, which learns to combine signals from both the CLAP audio model and the Lyrics Bi-encoder. By training a Multi-Layer Perceptron (MLP) to weigh these embeddings dynamically rather than using a simple average, we achieved the highest retrieval performance across our pipeline.

Key Metrics:

- MRR: 0.2824
- Recall@K:
    - Recall@1 = 0.2030
      Recall@5 = 0.3650
      Recall@10 = 0.4491
      Recall@20 = 0.5353
- Precision@K:
    - Precision@1 = 0.2030
      Precision@5 = 0.0730
      Precision@10 = 0.0449
      Precision@20 = 0.0268

Analysis: The significant jump in Recall@1 (20.30%) compared to the individual models demonstrates that the fusion layer effectively filters out candidates that might match well in one modality (e.g., correct mood) but fail in the other (e.g., wrong topic), resulting in much more precise top-level recommendations.





Project Timeline

Task	Responsible	Period
Phase 1: Planning & Setup		

Initial research on SOTA models (CLAP, LLMs) and dataset exploration	All Team	Sep 1 – Sep 16
Definition of project scope, goals	All Team	Sep 1 – Sep 16
Environment setup, framework experiments, and architecture design	Vladislav, Polina	Sep 1 – Sep 16
Phase 2: Data Collection & Baselines		
Collection and processing of text data (enrich-music4all, Song-Interpretation)	Polina	Sep 17 – Oct 7
Preparation of MusicCaps dataset: audio extraction and segmentation	Janna	Sep 17 – Oct 7
Implementation of Lyrics Bi-encoder baseline model	Polina	Sep 17 – Oct 7
Setup of CLAP training pipeline and initial fine-tuning experiments	Vladislav	Sep 17 – Oct 7
Phase 3: Advanced Engineering & Scaling		
Exploratory Data Analysis (EDA) on Jamendo & Sound Describer datasets	Janna	Oct 8 – Nov 4
Implementation of parquet-based pseudo-streaming for TB-scale data	Vladislav	Oct 8 – Nov 4
Optimization of CLAP: context window extension and custom scheduling	Vladislav	Oct 8 – Nov 4
Evaluation of audio retrieval models on Sound Describer Dataset	Janna	Oct 8 – Nov 4
Finalizing the bge-m3 based Lyrics Bi-Encoder and running full evaluations	Polina	Oct 8 – Nov 4
Phase 4: Fusion, Backend & Client (Final Sprint)		

Engineering the multimodal dataset (FMA + Song-Interpretation) via fuzzy matching	Polina	Nov 5 – Nov 25
Development and training of the Multimodal Fusion Model (Audio+Lyrics)	Polina	Nov 5 – Nov 25
Creation of LyricCovers 2.0 (1000 songs) demo dataset with full pipeline	Vladislav	Nov 5 – Nov 25
Backend API development (FastAPI) and Vector Database (FAISS/SQLite) implementation	Vladislav	Nov 5 – Nov 25
iOS Client Application development (Swift, UI/UX)	Janna	Nov 5 – Nov 25

Work Distribution

- **Vladislav Kalinichenko** – Backend Engineering, MLOps (Training pipelines & Optimization), Data Engineering (Pseudo-streaming), Infrastructure Architecture, Dataset Preparation.
- **Polina Korobeinikova** – NLP Engineering, Multimodal Modeling (Fusion algorithms), Data Alignment Strategy, Baseline Development, Dataset Preparation.
- **Janna Ivanova** – Mobile Development (iOS), Research & data analysis, Model Evaluation, Dataset Preparation.

Future Work

1. Production Deployment
Currently, our backend operates on a local network, restricting access to devices connected to the same Wi-Fi. A key next step is to deploy the FastAPI server and vector database to a cloud environment (e.g., AWS or Google Cloud) with GPU support. This would enable global access for the iOS application and ensure stability under higher user loads.

2. Dataset Validation & Expansion
The multimodal dataset used for our Fusion Model was created via automated fuzzy matching, which inherently introduces some noise. Future work involves performing rigorous manual validation of these audio-text pairs to improve training quality. Additionally, we aim to expand the dataset beyond the current selection, ensuring it remains up-to-date with modern tracks and diverse genres.

3. Music Library Scalability
Our demo application currently runs on a fixed library of 1,000 songs (from LyricCovers 2.0). To make the product truly viable, we plan to implement a scalable pipeline that can continuously ingest, process, and index new music releases, keeping the searchable database relevant and extensive for end-users.

4. Personalized Reranking with User Feedback
To move beyond static search relevance, we promising future direction to implement a personalization reranker. By capturing user interactions such as "likes," "dislikes," and "skips" within the iOS app, we can build a history-aware model. This reranker would dynamically re-order the candidate list retrieved by our Fusion Model, prioritizing songs that not only match the search query but also align with the user's specific musical taste and past listening behavior.

References

1. CLAP (2023): Elizalde et al., "Large-scale Contrastive Language-Audio Pretraining with Feature Fusion and Extraction."
    - arXiv:2311.10057
2. CLaMP 3 (2025): SOTA Comparison Baseline. "CLaMP 3: Universal Music Information Retrieval with Contrastive Language-Music Pre-training."
    - arXiv:2502.10362v1
3. MuLan: Huang et al., "MuLan: A Joint Embedding of Music Audio and Natural Language."
4. MusicCaps: Agostinelli et al., "MusicCaps: Capturing the Essence of Music with Generated Captions."
5. Spotify Research (2022): Mao et al., "What Do Lyrics Tell Us? Understanding Music Audio and Lyrics for Mood Classification."
6. Song Describer Dataset (SDD): Manco et al., "Song Describer: A Dataset for Detailed Music Captioning."
7. Music4All: Doh et al., "LP-MusicCaps: LLM-Based Pseudo-Music Captioning" / "Music4All-Onion".