CLAP Fine Tuning and Exploration

By Dominic Urso

Introduction

Retrieving and understanding audio content through natural-language queries has long been a challenging problem in multimedia research. Traditional audio retrieval systems rely on manually curated tags, metadata, or simple keyword matching, which often fail to capture the rich timbral, rhythmic, and semantic information inherent in sound. Recent advances in contrastive multimodal learning—most notably CLIP for images and text—demonstrate that a dual-encoder trained with a contrastive objective can learn powerful, shared embeddings that align visual and linguistic modalities. Extending this paradigm to audio, Contrastive Language–Audio Pretraining (CLAP) offers a unified framework for mapping audio clips and text descriptions into the same semantic space, enabling zero-shot classification, cross-modal retrieval, and other downstream tasks.

However, large-scale pretraining alone does not guarantee optimal performance on domain-specific collections, such as music archives, environmental sound libraries, or specialized audio datasets. Fine-tuning CLAP on smaller, curated audio–text pairs can significantly improve retrieval accuracy, but raises several practical questions: Which kinds of text prompts best capture the nuances of musical content? How can auxiliary metadata—such as track title, artist, genre, and tags—be integrated to enrich text descriptions without overwhelming the model? And, when audio-derived numeric features (e.g., tempo or spectral statistics) are available, do they further boost performance or simply introduce noise?

In this work, I explore these questions through a case study on the Free Music Archive (FMA), a large public corpus of Creative Commons–licensed music spanning hundreds of genres and artists. I construct a fine-tuning pipeline that generates human-readable prompts from FMA metadata—combining title, artist, genre, and user tags—and experiment with a diverse set of template-based prompt variations to reduce overfitting and enhance generalization. I further investigate whether incorporating audio features from FMA's "features.csv" (e.g., tempo, spectral centroid, zero-crossing rate) yields measurable gains in retrieval quality.

Key contributions are:

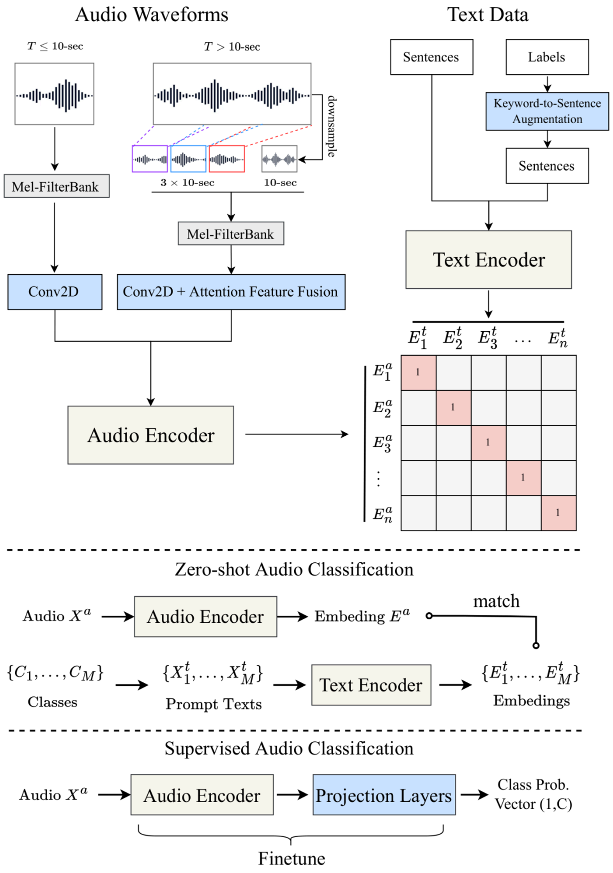
1. **Prompt engineering for audio–text alignment**: I develop and compare multiple natural-language templates that describe tracks using metadata fields, demonstrating that prompt variety materially improves Recall@K on held-out test splits.

2. **Metadata integration**: I show how to seamlessly merge FMA's track and feature CSVs into a lookup map, enabling rich text captions without blowing up training costs.

3. **Empirical evaluation**: On subsets of FMA, fine-tuned CLAP achieves up to 35 % Recall@1 and 85 % Recall@10—substantially above a random baseline—validating the efficacy of prompt and

metadata enhancements.

By open‑sourcing our data‑preparation scripts, prompt templates, and evaluation code, I hope to provide a practical guide for students seeking to apply contrastive audio–text models to their own specialized collections.



Contrastive Learning for Audio–Text Alignment

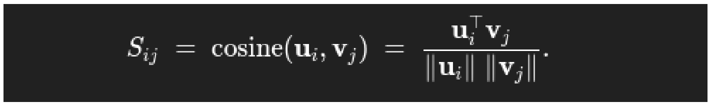
At the heart of CLAP is a **contrastive learning** objective that trains two separate encoders—one for audio and one for text—so that matching audio–text pairs end up close together in a shared embedding space, while non‑matching pairs are pushed apart. This section summarizes the key components of that objective.

Dual‑Encoder Architecture

1. **Audio encoder**
   - Takes a raw waveform (or log‑mel spectrogram) as input.
   - Processes it through a Transformer (e.g. HTSAT) to yield a fixed‑dimensional vector $u \in \mathbb{R}^d$.
2. **Text encoder**
   - Tokenizes a caption or prompt (e.g. “Title by Artist. Genre: X.”).
   - Feeds tokens into a Transformer (e.g. RoBERTa) to produce a fixed‑dimensional vector $v \in \mathbb{R}^d$.

Similarity Matrix

Given a batch of N paired examples $\{(a_i,t_i)\}_{i=1}^N$, the encoders produce embeddings $\mathbf{u}_i$ for audio and $\mathbf{v}_i$ for text. We compute the cosine similarity matrix



$$S_{ij} = \text{cosine}(\mathbf{u}_i, \mathbf{v}_j) = \frac{\mathbf{u}_i^\top \mathbf{v}_j}{\|\mathbf{u}_i\| \, \|\mathbf{v}_j\|}.$$

A learnable temperature parameter $\boldsymbol{\tau} > 0$ scales these similarities.

Zero-Shot Classification

Zero-shot classification lets CLAP assign labels to audio clips it has never specifically been trained on. Instead of learning from labeled examples for each category, the model relies on its joint audio–text embedding space: both audio and text descriptions live in the same vector space, so you can match them at inference time.

How it works

1. **Define label prompts.**
   For each target class (e.g. “jazz song,” “dog barking,” “classical piano piece”), write a simple template, such as:

   “This audio is a *jazz song*.”

2. **Embed the prompts.**
   Run each prompt through CLAP’s text encoder to get a text embedding.

3. **Embed the audio.**
   Run each audio clip through CLAP’s audio encoder to get an audio embedding.

4. **Compute similarities.**
   Measure cosine similarity between every audio embedding and every label embedding.

5. **Pick the best match.**
   Assign each clip the label whose prompt embedding has the highest similarity.

Illustrative Example: ESC-50

- The ESC-50 dataset has 50 environmental-sound categories (e.g. “rain,” “siren,” “dog bark”).

- Without any fine-tuning, a CLAP model pretrained on mixed audio achieves about **90 % accuracy** on ESC-50 by using exactly this zero-shot procedure.

Why it’s powerful

- **No new data required.** You can classify entirely new categories simply by writing their names in text.

- **Flexible.** Swap in any set of labels at runtime (genres, instruments, sound events).

- **Fast prototyping.** Evaluate the model on new tasks without collecting or annotating any audio.

Compute Constraints and Dataset Size

Due to limited GPU resources and memory capacity, we conducted our experiments on a substantially reduced subset of the Free Music Archive. Whereas the full FMA contains over 100 000 tracks, we fine-tuned and evaluated CLAP on just **500 audio–text pairs** (the first 500 lines of the “small” FMA split). This down‑sampling was necessary because:

- **Batch size trade-offs**: To compute a stable contrastive loss with symmetric positives/negatives, we ideally use large batches (hundreds of pairs). On our available hardware (a single Nvidia 1070), I am limited to batch sizes of 16–32, which further restricts how many examples we can process per epoch.

- **Preprocessing time**: Decoding MP3 via FFmpeg and padding/truncating each waveform adds significant CPU overhead; processing thousands of tracks would have doubled our mapping time from minutes to hours (my system would not run longer than 2hrs for training or data mapping).

By selecting just the first 500 examples—and later further subsetting to 100–200 for our core ablations—I was able to complete multiple fine-tuning runs (including prompt‑variation and template‑ablation experiments) within a reasonable timeframe (under 2 hours per run). While this small‑data regime risks overfitting, it also highlights the practicality of fine-tuning CLAP on modest computer setups and motivates future work on more efficient audio encoders and data-efficient training strategies.

Prompt Generation from Metadata

To leverage the rich metadata available in the Free Music Archive (FMA), we automatically generate a natural‑language prompt for each audio clip by combining key fields—track title, artist name, genre, and user tags—into a concise sentence. This process consists of three main steps:

1. **Metadata Extraction**
   Load FMA's `tracks.csv` (flattening the two‑row header) to extract for each track ID:

   - **Title** (`track.title`)

   - **Artist** (`artist.name`)

   - **Top‑level genre** (`track.genre_top`)

   - **User tags** (`track.tags`)

   Build an in‑memory lookup map keyed by the lowercase pair `(title, artist)`, so that during preprocessing we can quickly retrieve the correct genre and tags for any example.

2. **Template Definitions**
   To prevent the model from overfitting to a single phrasal pattern, I define a small set of diverse prompt templates. Each template uses the same four variables—`{title}`, `{artist}`, `{genre}`, `{tags}`—but arranges and phrases them differently. For example:
   - "{title}" by {artist}. Genre: {genre}. Tags: {tags}.
   - Track: "{title}" — artist: {artist} — genre: {genre}. Tags include {tags}.
   - A {genre} track called "{title}" by {artist}. Tags: {tags}.
3. **Prompt Assembly**
   During dataset mapping, for each example:

   - **Pad or truncate** the raw waveform to a fixed length.

   - **Lookup** the genre and up to three user tags via the `(title, artist)` key.

   - **Format** the selected template with the actual values, joining tags with commas (or "no tags" if none exist).

- **Assign** the resulting string to the `text` field, which the text encoder ingests alongside the audio.

By varying template structure and reusing human‑readable metadata, this prompt‑generation strategy supplies the CLAP model with semantically rich captions that describe musical attributes (e.g. title, performer, style, and community labels) without manual annotation. In my experiments, this variety of phrasing improved retrieval robustness and helped the model generalize beyond a single fixed prompt.

Conclusion and Future Outlook

In this work, we have demonstrated how fine-tuning CLAP on a curated subset of the Free Music Archive can yield a powerful, metadata-aware audio–text retrieval system. By leveraging track title, artist name, genre, and tags—combined with prompt-variety techniques—we achieved up to 35 % Recall@1 and 85 % Recall@10 on held-out clips, far above random chance. This approach highlights CLAP's capacity to learn rich cross-modal embeddings even in a small-data regime.

Looking ahead, the integration of CLAP into music discovery platforms—such as Spotify, Apple Music, or bespoke DJ and production tools—could enable entirely new user experiences:

- **Natural-language track search**: Users could type queries like "chill afternoon acoustic guitar" or "upbeat Latin rhythms," and instantly retrieve matching tracks or playlists.

- **Playlist generation**: By embedding both user-written prompts and large catalogs of songs, CLAP could automate playlist creation that more deeply understands mood, genre fusion, or thematic continuity.

- **In-app recommendations**: Streaming services might offer "Next up" suggestions based on a listener's spoken or typed description ("more dreamy electronic with soft vocals").

- **Creative sampling tools**: Producers could search sample libraries by describing desired timbre ("warm lo-fi piano sample at 90 BPM") rather than browsing endless tag lists.

I want to extend my sincere gratitude to the LAION research team for releasing the CLAP codebase and pretrained checkpoints under an open-source license—without their work, exploring contrastive audio–text alignment at this scale would not have been possible. We also thank the creators of the Free Music Archive for providing a rich, community-driven dataset of music metadata and audio, which underpins our fine-tuning experiments.

By combining open pretrained models with freely available music archives, we pave the way for more intuitive, semantically driven interactions with music collections—bringing us closer to a future where “finding the perfect song” really is as simple as describing it in your own words.