Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

CLAP Embeddings for Artist Similarity with Graph Neural Networks

Distributed Systems Lab Report

Luca Strässle

lucastr@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Florian Grötschla, Luca Lanzendörfer
Prof. Dr. Roger Wattenhofer

August 26, 2024

Acknowledgements

I want to thank my supervisors Florian Grötschla and Luca Lanzendörfer for their valuable insights and support. I am also thankful to Prof. Dr. Roger Wattenhofer for giving me the opportunity to contribute to the research at the Distributed Computing Group. Finally, I want to thank my girlfriend, family, and friends for their patience, understanding, and support.
Additionally, I would like to acknowledge the usefulness of ChatGPT and Grammarly for their assistance in refining my formulations and ensuring correct grammar.

Abstract

Music recommendation systems are essential in streaming services, helping users discover new music and aiding artists in gaining exposure. This thesis investigates the use of contrastive learning-based representations, specifically from the CLAP model, to enhance music recommendation systems. By predicting hidden edges in an artist graph using graph neural networks, we compare CLAP features with traditional audio features from AcousticBrainz. We use two datasets containing artist similarities from AllMusic, one of which was collected specifically for this work. Our findings show that CLAP features outperform AcousticBrainz features. This thesis highlights the potential of contrastive learning-based models to improve music recommendation systems.

Contents

Chapter 1

Introduction

Music and artist recommendations have become a cornerstone of streaming services. Many users now rely on algorithmically generated playlists, tailored to their individual music tastes, to find music that suits their mood and to discover new artists. These algorithms are immensely important for artists as well; being included in algorithmically generated playlists can significantly boost an artist's listener base, while exclusion can make it challenging for them to be discovered. The data used in music recommendation systems can be classified into two main categories: relational data and descriptive data. Relational data captures the relationships between artists or tracks, which can be manually curated through similarities or tags, or derived from user listening behavior and the interactions between users. Descriptive data, on the other hand, aims to capture the essence of an artist's music, traditionally through the extraction of musical attributes such as melody, harmony, or rhythm from a given track.
Recently, contrastive learning approaches have gained popularity for representing various kinds of data. CLAP [1] is a model that leverages contrastive learning to map text and audio into a joint multi-modal space. This introduces a novel way of representing music, and in this thesis, we aim to explore the utility of this representation as descriptive data in music recommendation systems.

Our research addresses the same problem as [2], focusing on predicting hidden edges in a graph of artists, where similar artists are connected based on similarities on AllMusic. By employing graph neural networks, we utilize this graph to refine artist representations. Our primary objective is to identify the best features to represent an artist's music. Specifically, we seek to determine if the modern approach of using CLAP representations can outperform the traditional approach of using low-level audio data from AcousticBrainz, as utilized in [2].
For evaluation, we used the dataset proposed in [2], extended with CLAP features. Additionally, we used our own version of the dataset, which followed a similar data collection process with some differences: incorporating an additional type of feature, being more strict in the track selection, and ensuring that CLAP and AcousticBrainz features are based on the same tracks.

Chapter 2

Related Work

This section begins with background on graph neural networks and then introduces the two principal works that form the basis of this thesis. Finally, related but less central works are discussed.

2.1 Graph Neural Networks

Neural Networks are highly successful in solving tasks in various domains such as image recognition or natural language processing. For example, Convolutional Neural Networks have become fundamental to image processing, excelling in tasks like image classification [3]. Traditional neural networks require highly structured input data like scalars, vectors, matrices, or tensors. However, some real-world data like social networks is best represented as a graph. Graph Neural Networks (GNNs) enable the application of deep learning techniques to graph data.
GNNs operate on graphs defined as $G = (V, E)$, where V represents the set of nodes and E represents the set of edges connecting these nodes. Each node $v \in V$ can have associated features x_v, and each edge $e \in E$ can have associated features y_e. The core mechanism of GNNs is message passing, which typically involves multiple rounds. A single message passing round consists of two steps: the aggregate step, where nodes gather information from their neighbors, and the update step, where the nodes' representations are updated based on their previous representation and the aggregated information from their neighbors. With $h_u^{(t)}$ representing node u's state at time step t and $\mathcal{N}(v)$ representing the set of nodes adjacent to node v, the aggregate and update steps are defined as

$$a_v^{(t)} = AGGREGATE\left(\left\{\left\{h_u^{(t-1)} \mid u \in \mathcal{N}(v)\right\}\right\}\right) \tag{2.1}$$

$$h_v^{(t)} = UPDATE\left(h_v^{(t-1)}, a_v^{(t)}\right), \tag{2.2}$$

where AGGREGATE is a permutation-invariant function that can take any number of inputs, ensuring that the order of the nodes does not affect the aggregated

information. [4] GNNs can be used to solve node-level tasks, such as node classification or node regression, edge-level tasks, such as edge classification or edge prediction, and graph-level tasks, such as graph classification [5].

2.2 Artist Similarity with Graph Neural Networks

The work presented in [2] served as the foundation for this thesis. The authors employed graph neural networks to predict whether two artists are similar. They utilized two different graph datasets. One was the OLGA dataset, which incorporated similarity relations from AllMusic and low-level audio features from AcousticBrainz as artist features and was also used in this thesis. To demonstrate the scalability of their approach, the authors also used a larger proprietary dataset. In this dataset, listener feedback from a music streaming service was used to define artist similarity and track attributes annotated by experts were used as artist features.
The graph neural network used consisted of a block of graph convolutions followed by a block of fully connected layers and was trained using the triplet loss. The results were evaluated using the NDCG_K metric in an ablation study, where the number of graph layers was varied, and artist features were either random or those provided in the dataset. It was shown that having one or more graph layers outperformed a deep neural network alone. Additionally, the performance generally improved with more graph layers. Furthermore, the use of non-random artist features outperformed random artist features in all cases. Interestingly, when using random features, one graph layer in the OLGA dataset and two graph layers in the proprietary dataset were enough to outperform a deep neural network with the features given in the datasets.

2.3 CLAP

Contrastive Language-Audio Pretraining (CLAP) [1] is a contrastive learning technique designed to learn the (dis)similarity of audio and text pairs. This method is inspired by the CLIP model [6], which was developed by OpenAI and performs a similar task for image and text pairs. CLAP generates representations of both audio and text in a joint multimodal space. The audio is processed through an audio encoder and then through a learnable linear projection to map it into the joint multimodal space. Similarly, the text is processed through a text encoder and a learnable linear projection. In this joint multimodal space, audio and text embeddings are comparable, and similarity can be measured. During training, given a set of pairs of corresponding audio and text, a similarity matrix is constructed. This matrix holds the similarity of each possible pair of audio and text embeddings, where the values on the diagonal correspond to correct

pairs and the others correspond to incorrect pairs. Symmetric cross-entropy loss is then applied to this similarity matrix to train the two encoders as well as the linear projections.

2.4 Other Work

Many different approaches have been studied to detect similarities in musical data. In [7] a graph autoencoder is used to learn latent representations on nodes in an artist graph, which are then used by a "gravity-inspired" decoder for ranking similar artists. In [8], a basic DCNN model as well as a Siamese DCNN model are used to extract features from artist labels. These features are then utilized in a genre classification task.
In addition, hybrid recommendation systems using GNNs have been studied for other applications as well. In [9] GNNs are used in conjunction with sentence transformer embeddings to predict anime recommendations for users. The underlying graph captures user-anime interactions, and BERT is used to create embeddings of the synopsis of an anime.

Chapter 3

Method

This thesis comprises two main components: the preparation of the two graph datasets used and the implementation, tuning, and evaluation of the Machine Learning pipeline. In this section, the most critical steps and concepts involved in these processes are described.

3.1 Data Collection

Two graph datasets were used: the OLGA dataset presented in [2] complemented with CLAP embeddings, and our own version called DISCO-OLGA. Both datasets contain various types of node features representing an artist's music. The exact composition of the datasets is described in Chapter 4. Below, the data sources used for both datasets, the generation of CLAP embeddings, and the collection process for the rest of the DISCO-OLGA dataset are explained.

3.1.1 Data Sources

In the data collection process for the OLGA and DISCO-OLGA datasets, the following sources were used:

- **MusicBrainz** (`musicbrainz.org`): MusicBrainz is an open music encyclopedia that collects music metadata, allowing anyone to contribute. It provides detailed information about artists and their music and offers an API for querying the database.
- **AcousticBrainz** (`acousticbrainz.org`) [10]: AcousticBrainz was an open platform for crowdsourcing acoustic information from music recordings. Users could run a feature extractor on audio files and upload the results to the AcousticBrainz server, where the data was made publicly available. The feature extractor, provided by AcousticBrainz, was based on the open-source Essentia audio analysis library. These features include information such as beats per minute (bpm), pitch, loudness, or spectral shape of the

signal. The full AcousticBrainz dataset can be downloaded or queried via an API, using MusicBrainz IDs as identifiers in the dataset.

- **AllMusic** (`allmusic.com`): AllMusic is an online music database that provides comprehensive information about artists, albums, and songs. Each artist and track is assigned a unique AllMusic ID, which MusicBrainz provides for some artists. AllMusic offers lists of similar artists, songs, and associated moods and themes for each artist.
- **YouTube** (`youtube.com`): YouTube is an online video-sharing platform that hosts a vast array of content, including a large music database. Many artists upload their songs to YouTube, making it a significant resource for music.

3.1.2 CLAP Embeddings

In Section 2.3 the structure and basic workings of a CLAP model were explained. We saw that after training, text and audio that are similar to each other should be close in the joint multimodal space. This implies that two audios that are similar to each other should be close in this space as well. Using the audio encoder of a pre-trained CLAP model, a representation of a music track in the joint multimodal space can be generated. In the problem of predicting artist similarity, this can be useful as an artist feature. Therefore, we generated CLAP embeddings for every artist in both datasets to represent their music.
For this, one track for every artist was selected. Using PyTube, YouTube was searched with the string `<artist_name track_name>` for every track, and the first result was downloaded as an mp3. Instead of selecting the first result, we considered using a more sophisticated technique to choose the best fitting out of the first couple of results, however, on manual inspection we found that this was not necessary, as the YouTube algorithm suggested the best fitting result first in almost all cases.
Finally, the audio encoder of LAION's music CLAP model (`music_audioset_epoch_15_esc_90.14`) was used to generate 512-dimensional embeddings of the track's mp3s. LAION is a non-profit organization, that provides datasets, tools, and models for machine learning research, including multiple pre-trained CLAP models [11]. As a last step, the embeddings were standardized.

3.1.3 Collection of DISCO-OLGA Dataset

The full OLGA dataset was only made available to us about halfway through this thesis. Initially, only the MusicBrainz IDs of the artists and their tracks were available on GitLab. We used this list of artists as a starting point to rebuild the dataset. Rebuilding it allowed us to add a few additional elements to the dataset with minimal extra effort. Our data collection process was based on OLGA's data

collection process described in [2]. In the following sections, the data collection process for our own DISCO-OLGA dataset is explained in more detail. Fig. 3.1 visualizes this data collection process.

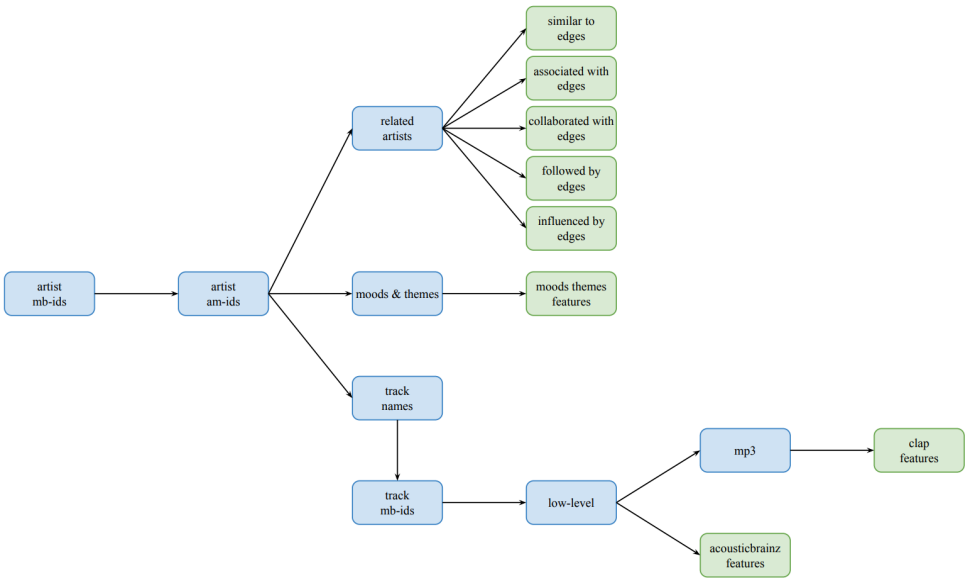

Figure 3.1: The process of data collection for the DISCO-OLGA dataset

Artist MusicBrainz ID to AllMusic ID

As a first step, the artist MusicBrainz IDs were transformed into AllMusic IDs. This transformation was necessary to obtain the similarity relations from AllMusic. For about two-thirds of the artists, MusicBrainz stored the AllMusic ID, which was retrieved by querying the MusicBrainz API. If the AllMusic ID was not available, the artist's name was used instead. Using the Selenium WebDriver, the name was entered into AllMusic's search page (`www.allmusic.com/search/artists/<name>`), and the AllMusic ID of the first search result was used. In a small number of cases, the same AllMusic ID appeared twice; in these instances, the incorrect classification was identified, and the correct AllMusic ID was found manually.

Scraping Related Artists, Tracks and Moods & Themes from AllMusic

On AllMusic, artist pages are divided into multiple subpages, each containing specific types of information about the artist. For all artists, the HTML code of the subpages *Related Artists*, *Songs*, and *Moods & Themes* was scraped using

the Selenium WebDriver. After opening the webpage, the WebDriver waited for the page to load. To identify if the page was fully loaded, a common element named `sidebarNav` present on all these webpages was waited for. If, after ten seconds, this element was still not present, a retry was initiated. Once the page was loaded, the source code was saved in an HTML file.
An artist's *Related Artists* subpage lists artists that are related in various ways. There are five kinds of relations: *Similar To*, *Influenced By*, *Followed By*, *Associated With* and *Collaborated With*. For the OLGA dataset, only the *Similar To* relations were used. For the DISCO-OLGA dataset, all kinds of relations were extracted from the HTML files. They were stored separately so that later on one could decide which relations to use. For each artist, five lists of AllMusic IDs, one for each kind of relation, were extracted from the HTML files using BeautifulSoup. These were then transformed into tuples representing edges, filtering out edges involving an artist with an AllMusic ID that was not in our list. Finally, these tuples were transformed into sparse matrices.
Since there was some error rate in the mapping from artist's MusicBrainz to AllMusic IDs the MusicBrainz track IDs available on GitLab did not correspond to the correct artist in some cases. Therefore we decided to get a list of songs from AllMusic instead. With BeautifulSoup AllMusic IDs and names of all tracks for every artist in the set were extracted from the HTML of the *Songs* subpages. Tracks from more than one artist in our set were excluded because they do not represent an individual artist's music well.
In the *Moods & Themes* subpage AllMusic lists a subset of 507 distinct moods and themes that represent an artist's music best. Again with BeautifulSoup the names and IDs of all moods and themes for every artist were extracted. Those were then encoded in the form of binary vectors which can be used as node features. These features do not exist in the OLGA dataset.

Track Name to MusicBrainz ID

In order to get the low-level audio features from AcousticBrainz, the MusicBrainz IDs were required for all tracks. MusicBrainz does not list tracks but recordings. All recordings of every track are listed, including all releases or also live versions. This can lead to a track being listed on MusicBrainz multiple times. As an example, Queen's Bohemian Rhapsody is listed over 150 times. For each track, the MusicBrainz API was queried for recordings with the artist name and track name. Each result comes with a score between 0 and 100. The MusicBrainz IDs of all recordings with a score of 100 were stored for each track. They were sorted by release date, listing the recording with the oldest release date first, because that is most likely the original version of the track.

Track MusicBrainz ID to AcousticBrainz Low-Level Audio Data

AcousticBrainz stores low-level audio data in JSON files, which can be downloaded from their website. For each track, the MusicBrainz IDs obtained in the previous step were looped through until a match was found in the AcousticBrainz dataset. The corresponding JSON file containing low-level audio data was then stored.
To generate AcousticBrainz feature vectors, one track was randomly selected per artist. Common attributes holding numerical values in the JSON files of all artists were identified. These attributes were extracted, flattened, and standardized, resulting in a 2,589-dimensional feature vector per artist.

Download of Track MP3 and Generation of CLAP Features

As explained in Section 3.1.2 PyTube and LAION's CLAP model were used to download track mp3s and generate CLAP features. For the search, the artist and track names available in the metadata of the AcousticBrainz low-level audio data were used.

3.2 Graph Neural Networks

The GNN used in this thesis is similar to the one described in [2]. Initially, the features are mapped to the inner dimension using a single linear layer. This is followed by a series of graph convolutional layers and an MLP with one hidden layer. Finally, another linear layer is applied to reach the output dimension. In the following, the most important components of the pipeline are highlighted, and the implementation is described in detail.

3.2.1 Triplet Loss

As a loss function, triplet loss was used. First introduced in [12], triplet loss is one of the most commonly used loss functions in various similarity problems. In our case, a triplet consists of three artists: the anchor a, which is any artist in the dataset; a positive p, which is an artist similar to the anchor; and a negative n which is an artist dissimilar to the anchor. Triplet loss then pulls the anchor and positive closer together and pushes the anchor and negative away from each other. It is defined as

$$\mathcal{L}(a, p, n) = [d(a, n) - d(a, p) + \Delta]^{+}, \tag{3.1}$$

where $d(\cdot)$ is a distance function like Euclidean or cosine, $[\cdot]^{+}$ is the ramp function, and Δ is the margin, a hyperparameter that ensures a minimum separation between anchor and negative relative to anchor and positive.

3.2.2 NDCG$_K$ Metric

To evaluate performance the Normalized Discounted Cumulative Gain (NDCG) was used in this thesis. NDCG is a metric that is often used in recommender systems. It not only considers the correctness of predictions but also their order. NDCG$_K$ takes the first K predictions into account.
In our case, NDCG is calculated for every artist. Given an artist a, all other artists are sorted based on distance to a (Euclidean or cosine). The K closest artists form the predicted list of similar artists $\hat{s}$. Additionally, an ideal list of similar artists s is required for calculating NDCG. It is defined as

$$NDCG_K = \frac{\sum_{k=1}^{K} g(\hat{s}, a)d(k)}{\sum_{k=1}^{K} g(s, a)d(k)}, \tag{3.2}$$

where the gain $g(\cdot, a)$ is one if the two artists are actually similar and zero otherwise. $d(k) = \log_2^{-1}(k+1)$ is the discounting factor that puts more weight on closer artists. For this thesis, a high cut-off of $K = 200$ was chosen, which is what was also used in [2]

3.2.3 Train/Test/Validation Split

Instead of splitting the edges into training, validation, and test sets, the artists are partitioned into the three sets. This splits the edges into five relevant groups.

- **Train - Train**: Edges between two nodes of the training set. Those edges are used for training.
- **Train - Validation**: Edges between a node in the train and a node in the validation set. Those edges are the known validation edges. They are used together with train-train edges as the graph that the GNN runs on during validation.
- **Validation - Validation**: Edges between two nodes of the validation set. Those edges are the actual validation edges. They should be predicted during validation and they are used to compute the metrics.
- **Train/Validation - Test**: Edges between a node in the train or validation set and a node in the test set. They are used together with train-train, train-validation and validation-validation edges as the graph that the GNN runs on during testing.
- **Test - Test**: Edges between two nodes of the test set. Those edges are the actual test edges. They should be predicted during testing and they are used to compute the metrics.

With this technique, evaluation artists have never been seen during training and the evaluation edges that should be predicted remain hidden.

3.2.4 Implementation

The basis for the implementation was code [13] written for a similar problem on the LastFM dataset [14]. This code was in turn based on code written for [15], where the authors set up a handy pipeline using PyTorch Geometric with GraphGym. There were five major adjustments that had to be made to the code, which will be described in the following.

Triplet Loss

The triplet loss as described in Section 3.2.1 is implemented by PyTorch in the function `TripletMarginWithDistanceLoss`. It allows to pass a distance function and two options were implemented. For Euclidean distance, PyTorch's `PairwiseDistance` function was used. For cosine distance the following was used with PyTorch's `cosine_similarity` function.

$$1 - cosine\_similarity(x, y) \tag{3.3}$$

Triplets

The existing code for the LastFM dataset used cross-entropy loss on randomly sampled sets of positive and negative edges. The data consisted of tuples with two node indices and a label. For prediction, the similarity between the two nodes was computed, and a threshold was used for binary classification. In this work, where triplet loss was used, triplets consisting of an anchor node, a positive node similar to the anchor, and a negative node dissimilar to the anchor were required. Given a set of nodes with a set of positive edges between them, where all non-positive edges are negative, there are two ways to construct triplets:

- For every positive edge, take both endpoints as the anchor once.
- For every positive edge, randomly select one endpoint as the anchor.

Given m positive edges, the first option creates $2m$ triplets, with m distinct positive and $2m$ distinct negative edges, while the second option creates m triplets, with m distinct positive and m distinct negative edges. Both options were implemented, with a parameter to choose the strategy. To sample the negative edges, PyTorch Geometric's `structured_negative_sampling` was used in both cases.

Data Loader

The data loader loads data from a given source and brings it into the correct format. The input data, in our case stored on the cloud, is downloaded. Edges are

created from sparse matrices and split into the groups described in Section 3.2.3. Based on a parameter, the correct node features are selected. Finally, everything is stored in PyTorch Geometric's `Data` object. The Data Loader is implemented with a class inheriting from the `InMemoryDataset` class of PyTorch Geometric.

Sampler

Training is done in multiple epochs, where in each epoch there is a run through multiple batches of data. The sampler is called in every batch and returns the correct underlying graph, as well as the triplets to train or evaluate on. The graphs and the triplets for evaluation are fixed. In the triplets used for training, the positive edges are fixed as well, and all positive training edges are used in every batch. However, the negative edges are randomly resampled for each batch. The Loader is implemented with a class inheriting from PyTorch's `DataLoader` class.

NDCG$_K$ Metric

We implemented the NDCG$_K$ metric, as described in Section 3.2.2, ourselves for this thesis. The first step involves calculating similarities between all pairs of points. Again, there are two options: Euclidean and cosine similarity. For Euclidean similarity, we use PyTorch's `pairwise_distance` function. The distance d is then transformed into a similarity using $\frac{1}{1+d}$. For cosine similarity, we use PyTorch's `cosine_similarity` function. Subsequently, the indices of the closest K nodes are extracted for all artists using PyTorch's `topk` function. Next, a mask is built that indicates which of the closest K nodes is actually similar for each node. This mask matrix is then multiplied by a vector holding the discounting factors to compute the discounted cumulative gains (DCG) for all nodes. To calculate the ideal discounted cumulative gain (IDCG), the first q discounting factors are summed for each node, where q corresponds to the number of edges each node participates in. The NDCG$_K$ for each node is obtained by dividing the DCG by the IDCG. Finally, the average of the NDCG$_K$ values for each artist is taken, excluding nodes that do not participate in any edge. Due to the necessity of computing the distance between all pairs of nodes, which takes $\mathcal{O}(n^3)$ time, NDCG$_K$ is computed only for validation and testing.

Chapter 4

Datasets

Table 4.1 provides a summary of the two datasets. The *acousticbrainz* features in the OLGA dataset are aggregated from up to twenty-five tracks per artist, whereas in the DISCO-OLGA dataset, they are derived from only a single track.

	OLGA	DISCO-OLGA
nodes	17,673 artists	16,864 artists
edges	101,029 *Similar To*	99,121 *Similar To* 40,307 *Influenced By* 20,135 *Followed By* 3,474 *Associated With* 2,412 *Collaborated With*
features	acousticbrainz clap	acousticbrainz clap moods-themes

Table 4.1: Table Summarizing the two Datasets

Notably, the OLGA dataset includes more artists than the DISCO-OLGA dataset, despite the data collection process for DISCO-OLGA, detailed in Section 3.1.3, beginning with OLGA's list of artists. This discrepancy arises because some artists were "lost" at various stages of the process. Initially, there were 17,673 artist MusicBrainz IDs. AllMusic IDs were found for 17,633 of these artists. Subsequently, tracks were identified on AllMusic for 17,438 artists, and MusicBrainz IDs for at least one track were found for 17,399 artists. Finally, low-level audio data was not available on AcousticBrainz for 535 artists, resulting in the 16,864 artists included in the DISCO-OLGA dataset.

As explained in Section 3.2.3 the dataset was split into train, test, and validation sets by artists. In the OLGA dataset, masks were provided on GitLab [16] indicating the split each artist belongs to. These masks were reused for the DISCO-OLGA dataset. In Table 4.2 there's a summary of how many artists are in each split and how many edges (of type *Similar To*) are of which edge group (as defined in Section 3.2.3) for both datasets.

	OLGA	DISCO-OLGA
train nodes	14,139	13,489
validation nodes	1,767	1,679
test nodes	1,767	1,696
train-train edges	64,595	63,277
train-validation edges	17,183	16,841
validation-validation edges	1,124	1,082
train/validation-test edges	17,243	17,037
test-test edges	884	884

Table 4.2: Table Summarizing the Splits of the two Datasets

Chapter 5

Experiments and Results

To evaluate our approach, we conducted a series of experiments. This chapter begins with a detailed description of the experimental setup, outlining our strategy and specifying the key hyperparameters, which were optimized through tuning. Following this, we present the results for both datasets. The chapter finishes with a comprehensive discussion of the findings.

5.1 Experimental Setup

As mentioned in Chapter 3, the triplet loss was utilized. For sampling the negative edges of the triplets, using both endpoints as anchors once performed better than randomly selecting one endpoint as an anchor. PyTorch's `AdamW` optimizer was employed with PyTorch's `ReduceLROnPlateau` scheduler. The Euclidean distance was used to measure the distance between two node embeddings for the Triplet Loss and the NDCG_K calculation. For the graph layers, we experimented with PyTorch Geometric's implementations of `SAGEConv` [17], `GatedGraphConv` [18], and `GINEConv` [19], with `SAGEConv` showing the best performance. The inner dimension of the network is 256 and the output dimension is 100.

Two aspects were varied in the experiments:

- **The number of graph layers:** This was varied from zero to four. With zero graph layers, the model consists of a linear layer, an MLP layer with one hidden layer, and another linear layer, meaning no information from the graph is learned. This configuration serves as our baseline. As the number of graph layers increases, nodes can gather information from a larger neighborhood, enhancing the model's capacity to learn from the graph structure.
- **The node features:** Using *random* features served as the baseline. Experiments were conducted using *acousticbrainz* features, *clap* features, and, in the case of DISCO-OLGA, *moods-themes* features alone. Additionally, we conducted runs of all combinations of the non-random features.

Each combination of the number of graph layers and node features was run three times. This resulted in a total of $5 \cdot 4 \cdot 3 = 60$ runs for OLGA and $5 \cdot 8 \cdot 3 = 120$ runs for DISCO-OLGA.

5.2 Results

In the following, the results for both datasets are presented. The NDCG_K metric, as explained in Section 3.2.2, is reported here with $K = 200$. Additionally, accuracy, precision, recall, f1-score, and area under the curve (AUC) were measured as well and are reported together with NDCG_{200} in tables in Appendix A.

5.2.1 OLGA

In Fig. 5.1, the results for the OLGA dataset are visualized. Generally, using more graph layers improves performance, with the most significant improvements occurring up to two graph layers. Beyond two layers, the performance gains are small. The use of *random* features consistently performs the worst in all cases. In contrast, using *acousticbrainz* features achieves the best performance. Using only *clap* features performs better than using *random* features but worse than using *acousticbrainz* features. Using both *acousticbrainz* and *clap* features performs similarly or slightly worse than using *acousticbrainz* alone. The introduction of graph layers makes the most significant difference, as even with one graph layer, *random* node features outperform all options without any graph layers.

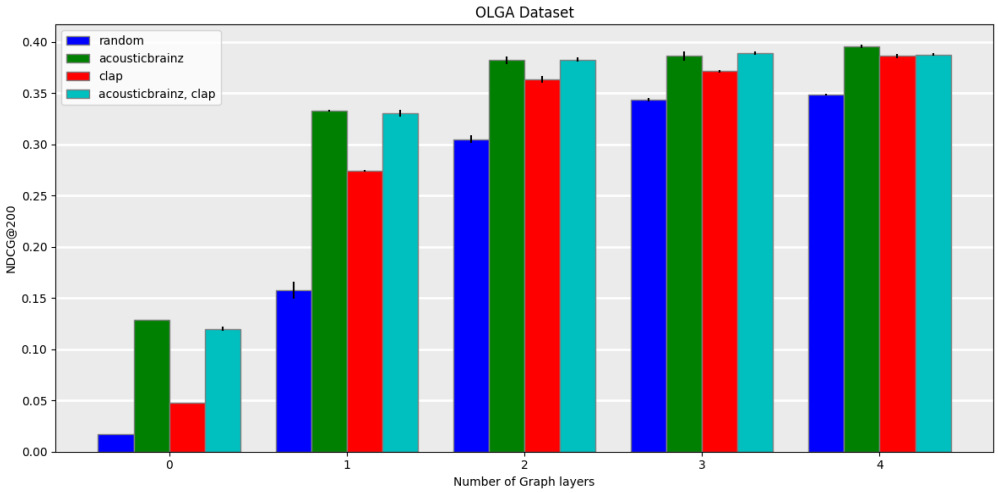

Figure 5.1: NDCG_{200} for Different Node Features and Different Number of Graph Layers for the OLGA Dataset

In [2], the authors also used *random* and *acousticbrainz* features for 0, 1, 2, and 3 graph layers. Overall, the NDCG scores they reported were higher, but the same general trends were observed. They used distance-weighted sampling [20] for selecting triplets, which aims to select more informative and stable samples. In this work, this method was not implemented, which might account for the difference in scores.

5.2.2 DISCO-OLGA

In Fig. 5.2, the results for the DISCO-OLGA dataset are visualized for the same node feature combinations as in Fig. 5.1 for the OLGA dataset. As with OLGA, performance generally improves with more graph layers, and using *random* features performs the worst. Using *random* features only achieves similar scores as in OLGA, and *acousticbrainz* as well as *clap* in combination with *acousticbrainz* features perform a bit worse overall. However, unlike OLGA, using only *clap* features performs the best in all cases using one or more graph layers. Combining *acousticbrainz* and *clap* features performs better than using *acousticbrainz* features alone but worse than using *clap* features alone. Additionally, as with OLGA, using *random* features with only a single graph layer already performs significantly better than any of the feature combinations with no graph layer.

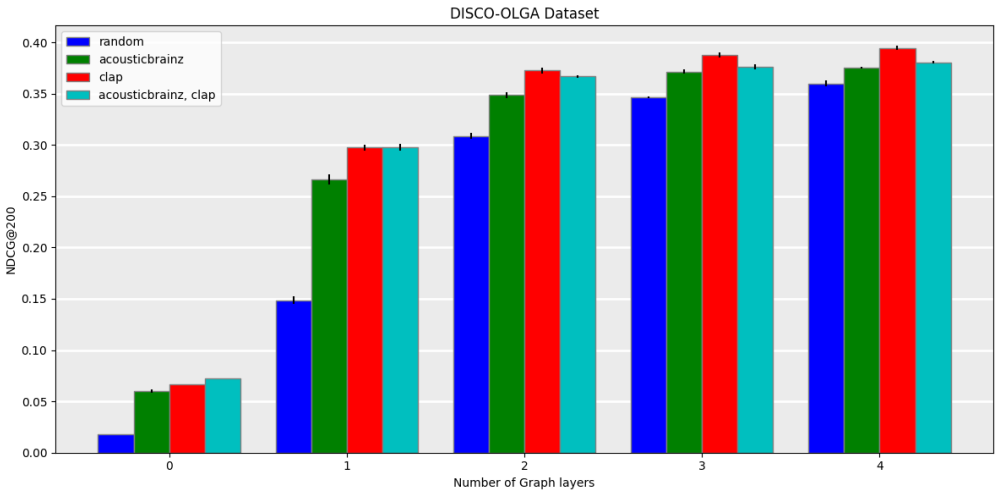

Figure 5.2: $NDCG_{200}$ for Different Node Features and Different Number of Graph Layers for the DISCO-OLGA Dataset

In the DISCO-OLGA dataset, there are additional *moods-themes* features, which are not available in the OLGA dataset. The results of all feature combinations including *moods-themes* and of just *clap* features as a reference are visualized in Fig. 5.3. Using *moods-themes* alone performs similarly to *clap* only

if no graph layers are used, but significantly worse with one or more graph layers. Not plotted here, but with four graph layers, it performs just as poorly as using *random* features. Combining *moods-themes* with other features outperform just using *clap* for zero or one graph layer. For more graph layers, only the combination of *clap* and *moods-themes* features performs similarly to just using *clap*, but it cannot improve performance. The other combinations do not come close. Using *moods-themes* features leads to a significant improvement for zero or one graph layer but no improvement for more.

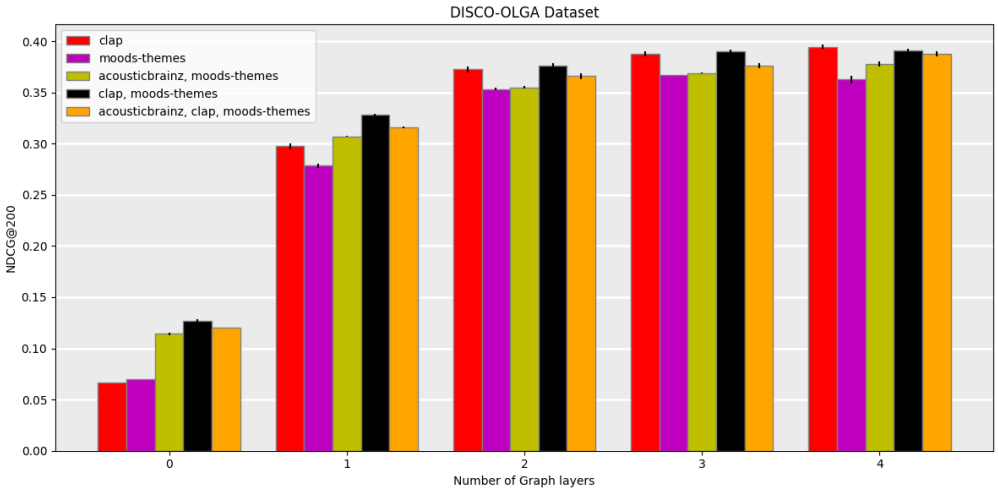

Figure 5.3: $NDCG_{200}$ for Node Features with Moods & Themes and Different Number of Graph Layers for the DISCO-OLGA Dataset

5.3 Discussion

In the OLGA dataset, *acousticbrainz* features outperform *clap* features, whereas, in the DISCO-OLGA dataset, it is the other way around. However, the scores of the *clap* features in the DISCO-OLGA dataset are lower than those of the *acousticbrainz* features in the OLGA dataset. These observations can be explained by the fact that the *acousticbrainz* features in the OLGA dataset are derived from multiple tracks by each artist, while in the DISCO-OLGA dataset, they are based on a single track, similar to the *clap* features, which are based on a single track in both datasets. This indicates that *clap* features are actually outperforming *acousticbrainz* features in a fair comparison.
Additionally, using just the *clap* features yields better performance in the OLGA dataset compared to the DISCO-OLGA dataset. While the exact reason for this is unclear, it could be attributed to the track selection. For the DISCO-OLGA

dataset, we excluded tracks featuring multiple artists from our list, whereas the OLGA dataset included such collaborations. Moreover, in the OLGA dataset, tracks were sourced from MusicBrainz, while in the DISCO-OLGA dataset, tracks were sourced from AllMusic. AllMusic generally lists fewer tracks than MusicBrainz but includes the most relevant ones, while MusicBrainz just includes all tracks. Therefore, selecting a track from AllMusic's list is more likely to yield a track that well represents the artist.
Finally, we observe that with enough graph layers, the performance of *clap* features based on just one well-selected track gets close to the performance of *acousticbrainz* features based on twenty-five tracks. This is a particularly interesting result because using one track per artist requires much less effort than using twenty-five tracks. Moreover, it suggests that if multiple tracks were used for *clap* features, they would likely outperform the multiple-track version of the *acousticbrainz* features.

Adding graph layers to the network significantly improves the score across all cases. Notably, a single graph layer with random features outperforms any feature combination without a graph layer, highlighting the importance of the contextual information embedded in the graph topology for predicting artist similarity. This underscores that the use of graph layers is crucial for enhancing performance in this artist similarity task. The most substantial performance gains are observed with up to two graph layers. As noted in [2], this is likely because most similar artists are connected through at least one common artist.

Combining different types of features generally did not lead to a major improvement in scores. Using *clap* and *acousticbrainz* features together performed similarly to or slightly worse than the better of the two alone, yet always better than the worse of the two alone. This suggests that *clap* and *acousticbrainz* features generally capture the same information. This is reasonable, as both feature types represent an artist's tracks, and in the case of the DISCO-OLGA dataset, even the same track. It is noteworthy that two very different techniques extract such similar information from a given track.
Combining *moods-themes* features with other types of features yields better results. Without graph layers, where features play a more crucial role due to the lack of graph topology information, *moods-themes* features combined with *acousticbrainz* and/or *clap* features significantly outperform the use of any single type of feature. This is expected, as *moods-themes* features provide different information compared to *clap* or *acousticbrainz* features. However, with the addition of more graph layers, these combinations do not achieve better scores than using *clap* features alone. This suggests, that the *moods-themes* features hold the same information, that is contained in the graph as well.

Chapter 6

Conclusion and Future Work

In the following, the key findings of this thesis are summarized.

- Features generated with CLAP outperform those from AcousticBrainz in a fair comparison and require less tracks to achieve high performance.
- Combining both relational and descriptive data yields the best performance.
- Merging features from CLAP with those from AcousticBrainz does not enhance the score, as they hold similar information. However, integrating either with Moods & Themes features improves the score when no graph layers are used. With sufficient graph layers, the Moods & Themes features become redundant, as their information is contained in the graph as well.

We were able to show that meaningful representations of an artist's music can be generated using CLAP. This approach to generating descriptive data shows better results than the more traditional approach with AcousticBrainz. The fact that fewer tracks are required to achieve similar performance makes it a more feasible option to implement. Additionally, as contrastive learning is a relatively new area of research, it is reasonable to assume that audio encoders like CLAP will improve in the future. As only the pre-trained audio encoder would need to be exchanged, such advancements would be very easy to integrate.

This work opens the door to many interesting questions that could be worked on in the future. First of all, instead of using the CLAP embedding of a single track per artist as a feature, one could examine the impact on performance when using CLAP embeddings of multiple tracks per artist. While twenty-five tracks probably won't be necessary, having two or three songs per artist would probably lead to better performance.
In this work, only CLAP was considered as a music embedding model. There are many other music embedding models, such as Jukebox [21] or Deep-Music-Autoencoder [22]. Future work could evaluate different music embedding models. Finally, in this thesis for the DISCO-OLGA dataset, additional edges based on other relations were collected. While these edges were not used in the current work, they could be utilized for message passing in future research. So far,

only one edge type has been used, but graph neural networks can accommodate different kinds of edges as well. Given the positive impact of the *Similar To* edges, incorporating other types of edges could be promising.

Bibliography

[1] B. Elizalde, S. Deshmukh, M. A. Ismail, and H. Wang, "Clap learning audio concepts from natural language supervision," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[2] F. Korzeniowski, S. Oramas, and F. Gouyon, "Artist similarity with graph neural networks," *CoRR*, vol. abs/2107.14541, 2021. [Online]. Available: https://arxiv.org/abs/2107.14541

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[4] R. Wattenhofer and P. A. Papp, "Principles of distributed computing, chapter 11: Graph neural networks." [Online]. Available: https://disco.ethz.ch/courses/podc/lecturenotes/chapter11.pdf

[5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[6] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8748–8763. [Online]. Available: https://proceedings.mlr.press/v139/radford21a.html

[7] G. Salha-Galvan, R. Hennequin, B. Chapus, V.-A. Tran, and M. Vazirgiannis, "Cold start similar artists ranking with gravity-inspirednbsp;graphnbsp;autoencoders," in *Proceedings of the 15th ACM Conference on Recommender Systems*, ser. RecSys '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 443–452. [Online]. Available: https://doi.org/10.1145/3460231.3474252

[8] J. Park, J. Lee, J. Park, J.-W. Ha, and J. Nam, "Representation learning of music using artist labels," 2018.

[9] S. R. Javaji and K. Sarode, "Hybrid recommendation system using graph neural network and bert embeddings," 2023.

[10] A. Porter, D. Bogdanov, R. Kaye, R. Tsukanov, and X. Serra, "Acousticbrainz: a community platform for gathering music information obtained from audio," in *Müller M, Wiering F, editors. ISMIR 2015. 16th International Society for Music Information Retrieval Conference; 2015 Oct 26-30; Málaga, Spain. Canada: ISMIR; 2015.* International Society for Music Information Retrieval (ISMIR), 2015.

[11] Y. Wu, K. Chen, T. Zhang, Y. Hui, T. Berg-Kirkpatrick, and S. Dubnov, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[12] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[13] fgroetschla, "Lastfm," https://gitlab.ethz.ch/fgroetschla/lastfm, 2024.

[14] I. Cantador, P. Brusilovsky, and T. Kuflik, "2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011)," in *Proceedings of the 5th ACM conference on Recommender systems*, ser. RecSys 2011. New York, NY, USA: ACM, 2011.

[15] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a General, Powerful, Scalable Graph Transformer," *Advances in Neural Information Processing Systems*, vol. 35, 2022.

[16] Fdlm, "Olga," https://gitlab.com/fdlm/olga, 2024.

[17] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2018.

[18] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," 2017.

[19] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," 2020.

[20] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Sampling matters in deep embedding learning," 2018.

[21] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[22] jakezhaojb, "Deep-music-autoencoder," https://github.com/jakezhaojb/Deep-Music-Autoencoder.

Appendix A

Results Extended

In addition to measuring $NDCG_{200}$, all positive edges between two test nodes and an equal number of randomly sampled negative edges were used to measure accuracy, precision, recall, F1-score, and area under the curve (AUC). The following tables report all results for both datasets, including means and standard deviations. Interestingly, all values are relatively high for runs with at least one graph layer and non-random node features.

A.1 OLGA

# graph layers	0	1	2	3	4
random	0.66 ± 0.00	0.74 ± 0.01	0.88 ± 0.00	0.93 ± 0.00	0.93 ± 0.00
acousticbrainz	0.78 ± 0.00	0.92 ± 0.00	0.93 ± 0.00	0.94 ± 0.00	0.94 ± 0.00
clap	0.67 ± 0.00	0.91 ± 0.00	0.93 ± 0.00	0.94 ± 0.00	0.94 ± 0.00
acousticbrainz clap	0.77 ± 0.00	0.92 ± 0.00	0.94 ± 0.00	0.94 ± 0.00	0.95 ± 0.00

Table A.1: OLGA Accuracy

# graph layers	0	1	2	3	4
random	0.23 ± 0.00	0.92 ± 0.01	0.95 ± 0.00	0.92 ± 0.01	0.91 ± 0.01
acousticbrainz	0.80 ± 0.00	0.93 ± 0.00	0.92 ± 0.00	0.94 ± 0.01	0.94 ± 0.00
clap	0.51 ± 0.00	0.89 ± 0.00	0.90 ± 0.00	0.93 ± 0.00	0.92 ± 0.01
acousticbrainz clap	0.80 ± 0.01	0.93 ± 0.00	0.92 ± 0.01	0.93 ± 0.00	0.94 ± 0.00

Table A.2: OLGA Precision

# graph layers	0	1	2	3	4
random	0.00 ± 0.00	0.24 ± 0.02	0.67 ± 0.01	0.85 ± 0.00	0.88 ± 0.00
acousticbrainz	0.45 ± 0.00	0.81 ± 0.01	0.88 ± 0.00	0.89 ± 0.00	0.88 ± 0.00
clap	0.32 ± 0.00	0.81 ± 0.01	0.89 ± 0.00	0.89 ± 0.01	0.90 ± 0.00
acousticbrainz clap	0.40 ± 0.01	0.81 ± 0.00	0.89 ± 0.01	0.89 ± 0.00	0.90 ± 0.00

Table A.3: OLGA Recall

# graph layers	0	1	2	3	4
random	0.01 ± 0.00	0.38 ± 0.02	0.79 ± 0.01	0.88 ± 0.00	0.89 ± 0.00
acousticbrainz	0.58 ± 0.00	0.87 ± 0.01	0.90 ± 0.00	0.91 ± 0.00	0.91 ± 0.00
clap	0.39 ± 0.00	0.85 ± 0.01	0.89 ± 0.00	0.91 ± 0.00	0.91 ± 0.00
acousticbrainz clap	0.54 ± 0.01	0.87 ± 0.00	0.91 ± 0.00	0.91 ± 0.00	0.92 ± 0.00

Table A.4: OLGA F1-Score

# graph layers	0	1	2	3	4
random	0.51 ± 0.00	0.85 ± 0.01	0.95 ± 0.00	0.96 ± 0.00	0.96 ± 0.00
acousticbrainz	0.82 ± 0.00	0.96 ± 0.00	0.98 ± 0.00	0.98 ± 0.00	0.98 ± 0.00
clap	0.65 ± 0.00	0.96 ± 0.00	0.97 ± 0.00	0.98 ± 0.00	0.98 ± 0.00
acousticbrainz clap	0.80 ± 0.00	0.96 ± 0.00	0.98 ± 0.00	0.98 ± 0.00	0.98 ± 0.00

Table A.5: OLGA AUC

# graph layers	0	1	2	3	4
random	0.02 ± 0.00	0.16 ± 0.01	0.31 ± 0.00	0.34 ± 0.00	0.35 ± 0.00
acousticbrainz	0.13 ± 0.00	0.33 ± 0.00	0.38 ± 0.00	0.39 ± 0.00	0.40 ± 0.00
clap	0.05 ± 0.00	0.27 ± 0.00	0.36 ± 0.00	0.37 ± 0.00	0.39 ± 0.00
acousticbrainz clap	0.12 ± 0.00	0.33 ± 0.00	0.38 ± 0.00	0.39 ± 0.00	0.39 ± 0.00

Table A.6: OLGA NDCG_{200}

A.2 DISCO-OLGA

# graph layers	0	1	2	3	4
random	0.66 ± 0.00	0.73 ± 0.01	0.87 ± 0.00	0.92 ± 0.00	0.93 ± 0.00
acousticbrainz	0.70 ± 0.00	0.89 ± 0.00	0.93 ± 0.00	0.94 ± 0.00	0.95 ± 0.00
clap	0.71 ± 0.00	0.92 ± 0.00	0.94 ± 0.00	0.94 ± 0.00	0.94 ± 0.00
moods-themes	0.47 ± 0.00	0.90 ± 0.00	0.93 ± 0.00	0.93 ± 0.00	0.93 ± 0.00
acousticbrainz clap	0.72 ± 0.00	0.91 ± 0.00	0.94 ± 0.00	0.94 ± 0.00	0.94 ± 0.00
acousticbrainz moods-themes	0.77 ± 0.00	0.91 ± 0.00	0.93 ± 0.00	0.94 ± 0.00	0.94 ± 0.00
clap moods-themes	0.78 ± 0.00	0.92 ± 0.00	0.94 ± 0.00	0.95 ± 0.00	0.94 ± 0.00
acousticbrainz clap moods-themes	0.76 ± 0.00	0.92 ± 0.00	0.94 ± 0.00	0.94 ± 0.00	0.94 ± 0.00

Table A.7: DISCO-OLGA Accuracy

# graph layers	0	1	2	3	4
random	0.09 ± 0.00	0.96 ± 0.02	0.96 ± 0.00	0.92 ± 0.00	0.93 ± 0.01
acousticbrainz	0.61 ± 0.01	0.91 ± 0.00	0.93 ± 0.01	0.93 ± 0.01	0.94 ± 0.00
clap	0.60 ± 0.00	0.91 ± 0.01	0.92 ± 0.00	0.94 ± 0.00	0.95 ± 0.00
moods-themes	0.37 ± 0.00	0.85 ± 0.00	0.86 ± 0.00	0.86 ± 0.00	0.85 ± 0.00
acousticbrainz clap	0.67 ± 0.00	0.92 ± 0.00	0.93 ± 0.00	0.94 ± 0.01	0.95 ± 0.00
acousticbrainz moods-themes	0.78 ± 0.01	0.94 ± 0.00	0.93 ± 0.00	0.93 ± 0.00	0.95 ± 0.00
clap moods-themes	0.80 ± 0.01	0.94 ± 0.00	0.94 ± 0.00	0.95 ± 0.00	0.95 ± 0.00
acousticbrainz clap moods-themes	0.79 ± 0.00	0.93 ± 0.00	0.94 ± 0.00	0.94 ± 0.00	0.96 ± 0.00

Table A.8: DISCO-OLGA Precision

# graph layers	0	1	2	3	4
random	0.00 ± 0.00	0.21 ± 0.01	0.63 ± 0.01	0.84 ± 0.00	0.85 ± 0.01
acousticbrainz	0.27 ± 0.01	0.76 ± 0.01	0.87 ± 0.01	0.89 ± 0.00	0.90 ± 0.01
clap	0.36 ± 0.00	0.83 ± 0.01	0.89 ± 0.00	0.88 ± 0.00	0.88 ± 0.01
moods-themes	0.89 ± 0.00	0.86 ± 0.00	0.93 ± 0.00	0.94 ± 0.00	0.94 ± 0.00
acousticbrainz clap	0.29 ± 0.00	0.79 ± 0.00	0.88 ± 0.00	0.88 ± 0.01	0.87 ± 0.00
acousticbrainz moods-themes	0.42 ± 0.01	0.79 ± 0.01	0.86 ± 0.00	0.89 ± 0.00	0.87 ± 0.00
clap moods-themes	0.44 ± 0.00	0.82 ± 0.00	0.87 ± 0.00	0.90 ± 0.01	0.88 ± 0.00
acousticbrainz clap moods-themes	0.40 ± 0.00	0.81 ± 0.00	0.85 ± 0.01	0.88 ± 0.00	0.87 ± 0.00

Table A.9: DISCO-OLGA Recall

# graph layers	0	1	2	3	4
random	0.00 ± 0.00	0.34 ± 0.02	0.76 ± 0.01	0.88 ± 0.00	0.89 ± 0.00
acousticbrainz	0.37 ± 0.00	0.83 ± 0.00	0.90 ± 0.00	0.91 ± 0.00	0.92 ± 0.01
clap	0.45 ± 0.00	0.87 ± 0.00	0.90 ± 0.00	0.91 ± 0.00	0.91 ± 0.00
moods-themes	0.53 ± 0.00	0.86 ± 0.00	0.89 ± 0.00	0.90 ± 0.00	0.89 ± 0.00
acousticbrainz clap	0.41 ± 0.00	0.85 ± 0.00	0.91 ± 0.00	0.91 ± 0.00	0.91 ± 0.00
acousticbrainz moods-themes	0.54 ± 0.00	0.86 ± 0.00	0.90 ± 0.00	0.91 ± 0.00	0.91 ± 0.00
clap moods-themes	0.57 ± 0.00	0.88 ± 0.00	0.90 ± 0.00	0.92 ± 0.00	0.91 ± 0.00
acousticbrainz clap moods-themes	0.53 ± 0.00	0.86 ± 0.00	0.90 ± 0.00	0.91 ± 0.00	0.91 ± 0.00

Table A.10: DISCO-OLGA F1-Score

# graph layers	0	1	2	3	4
random	0.49 ± 0.00	0.85 ± 0.00	0.95 ± 0.00	0.97 ± 0.00	0.97 ± 0.00
acousticbrainz	0.71 ± 0.00	0.95 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.98 ± 0.00
clap	0.70 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.98 ± 0.00	0.98 ± 0.00
moods-themes	0.66 ± 0.00	0.96 ± 0.00	0.98 ± 0.00	0.98 ± 0.00	0.98 ± 0.00
acousticbrainz clap	0.73 ± 0.00	0.96 ± 0.00	0.98 ± 0.00	0.98 ± 0.00	0.98 ± 0.00
acousticbrainz moods-themes	0.82 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.98 ± 0.00	0.98 ± 0.00
clap moods-themes	0.83 ± 0.00	0.97 ± 0.00	0.98 ± 0.00	0.98 ± 0.00	0.98 ± 0.00
acousticbrainz clap moods-themes	0.83 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.98 ± 0.00	0.98 ± 0.00

Table A.11: DISCO-OLGA AUC

# graph layers	0	1	2	3	4
random	0.02 ± 0.00	0.15 ± 0.00	0.31 ± 0.00	0.35 ± 0.00	0.36 ± 0.00
acousticbrainz	0.06 ± 0.00	0.27 ± 0.00	0.35 ± 0.00	0.37 ± 0.00	0.38 ± 0.00
clap	0.07 ± 0.00	0.30 ± 0.00	0.37 ± 0.00	0.39 ± 0.00	0.39 ± 0.00
moods-themes	0.07 ± 0.00	0.28 ± 0.00	0.35 ± 0.00	0.37 ± 0.00	0.36 ± 0.00
acousticbrainz clap	0.07 ± 0.00	0.30 ± 0.00	0.37 ± 0.00	0.38 ± 0.00	0.38 ± 0.00
acousticbrainz moods-themes	0.11 ± 0.00	0.31 ± 0.00	0.35 ± 0.00	0.37 ± 0.00	0.38 ± 0.00
clap moods-themes	0.13 ± 0.00	0.33 ± 0.00	0.38 ± 0.00	0.39 ± 0.00	0.39 ± 0.00
acousticbrainz clap moods-themes	0.12 ± 0.00	0.32 ± 0.00	0.37 ± 0.00	0.38 ± 0.00	0.39 ± 0.00

Table A.12: DISCO-OLGA NDCG_{200}