**Song similarity retrieval**

**Multimodal Machine Learning**

MSc in Artificial Intelligence, NCSR Demokritos & University of Piraeus

Sections

[1 Introduction 1](#_Toc170580857)

[2 Architecture pipeline 2](#_Toc170580858)

[2.1 Datasets 2](#_Toc170580859)

[2.2 Classifier models 3](#_Toc170580860)

[2.3 Metadata 5](#_Toc170580861)

[2.4 Autoencoder 5](#_Toc170580862)

[3. Similarity 6](#_Toc170580863)

[4. Experiments 6](#_Toc170580864)

[5. Conclusion 6](#_Toc170580865)

[6. References 6](#_Toc170580866)

## 1 Introduction

In this project, we deal with the issue of a song recommendation system based on closest similarity metric comparison to a song database. Our goal is to find a well-represented mapping of the different song characteristics that depend on song similarity to a latent space. Among the many different approaches for this problem, we have decided to utilize representation learning techniques with two modalities: the song’s raw audio and text metadata. To further enrich and refine our representations, we use differ supervised feature embeddings.

We explain our data preprocessing in section 2, where we used audio feature extraction to extract the Mel-spectrograms of the training dataset songs. In section 3, we show how we used a Convolutional Neural Network (CNN) for genre classification based on these Mel-spectrograms and use their embeddings for relevant representation on the raw audio input. In further section 4, we will present how we additionally took advantage of further metadata and a final Autoencoder architecture for a final mixed representation for all songs in the database.

The final model will be able to create an embedding vector to represent a given song’s audio and metadata information. Then, it will search in the database for similar songs based on the cosine similarity value of their representations.

## 2 Architecture pipeline

As described, we will try to gather as much relevant information as possible regarding song similarity. Our architecture relies on combining multiple such representations and creating a final merged database that translated all underlying dependencies. Our general pipeline can be seen in Figure 0:

Εικόνα που περιέχει κείμενο, διάγραμμα, σκίτσο/σχέδιο, ζωγραφιά

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 0: Song representation retrieval pipeline

In the following chapters, we will discuss further on the nature if the datasets used, the underlying models and different representations to form our final latent space database.

### 2.1 Datasets

As a first step, we needed a dataset with audio data, so that we could extract the mel-spectograms. We choose to use the mel-spectograms of each audio file, because it presents a discernible way to represent a song’s time and frequency domain features. We will use these mel-spectograms for the CNN genre classification mentioned in the following section. For this reason, we concluded to using the GTZAN dataset from kaggle [ANAFORA], which contains raw audio input of 10 genres with 100 audio clips each, so we have balanced dataset. Each song is of 30 seconds duration and is divided per genre in different folders.

To extract the mel-spectograms, we utilized the librosa library, which sampled the whole duration of the song at a sampling rate of 22050.

We also used librosa for audio feature extraction to extract the chroma STFT features, MFCC, RMS and the zero cross rate. We calculated for each output feature its mean and standard deviation and saved the audio features on a separate dataframe.

To get better correlation of the final embeddings, we also needed some extra metadata values. For this reason, we used the Million song dataset found in Kaggle [ANAFORA]. Further details are explained in relevant section.

### 2.2 Classifier models

As mentioned, we will use a CNN model to classify songs into different genre labels based on the song’s mel-spectogram. Normally used for computer vision tasks, the CNN is also able to extract useful features from a given spectrogram.

Εικόνα που περιέχει στιγμιότυπο οθόνης, πολυχρωμία, γραμμή, γραφικά

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 1: Implementation of CNNs for classification on mel-spectograms. Source [3].

We believe that similar songs also might belong to same genre, so by training a genre classifier we will successfully get representations of songs based on genre. The closest a representation is to another, the more likely it is to belong to the same genre and also be similar as a whole.

The prediction happens with a last layer Softmax function that computes the final probabilities of each class. The GTZAN dataset used to train the model contains 10 genres which we use as target labels:

*disco, metal, reggae, blues, rock, classical, jazz, hiphop, country, pop*

Since our dataset is already balanced with 100 songs per class, we use Stratified split to create the training, test and validation dataset with [][][] samples respectively.

#### 2.2.1 Representation output

We train our CNN with the train dataset in batches. After training, the model is creating good representations of genre dependencies in regards to input mel-spectograms, so we can use these as representations for similarity between genres.

Our training dataset input dimensions are [][][ and the final embedding layer of the CNN after training gives latent embedding of dimensions 256. Essentially, the last Forward layer creates a high-dimensional genre-space that can be used with a similarity metric to find the closest songs. However, genre is not the only factor in song similarity, so we will use these embeddings in addition to more feature representations. In later sections we will explain these additional features and finally how they are all combined for a final representation of our database.

#### 2.2.2 Evaluation

We trained our CNN for 100 epochs for the whole training dataset. We used a best model approach, were we saved the final model to be the one that had the best performance in the 100 epochs.

Εικόνα που περιέχει κείμενο, γραμμή, γράφημα, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 2: CNN Genre classifier training and validation loss, 100 epochs

As we can see from the loss graph in Figure 2, the best model is obtained at approximately epoch 30, before the validation loss gets out of hand. The training loss decreases steadily and consistently over the epochs and eventually flattens out, which suggests the model is fitting the training data well. We can also see this in the achieved accuracies on the test dataset from the given confusion matrix:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, τετράγωνο, ορθογώνιο παραλληλόγραμμο

Περιγραφή που δημιουργήθηκε αυτόματα

From Figure 3, it is evident that the model achieves great classification ability for most categories, while it mostly confuses classical and rock music label. This could be because of similarity in the spectrograms of the two classes. However, it is an all in all good classifier and we will further proceed with using its embeddings to receive meaningful representations of each song.

### 2.3 Metadata

To embed further information about the songs, we will try to also use metadata information based on the Million Song Dataset [ANAFORA].This dataset contains around 50000 songs , 1500 of which contain audio, with 21 different metadata. We filtered down the values to the ones we believe have a relation to song similarity to:

*Track\_id, artist, year, danceability, energy, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, time\_signature.*

No further separate encoding is performed on the metadata. Instead, both the CNN embeddings and metadata are used as input to the Autoencoder network and we obtain the final encoded representations there.

### 2.4 Autoencoder

To create a combination of representations based on the different feature embeddings, we can create an Autoencoder architecture. In this way, the hidden dependencies among all features can be learned and we can map the high-dimension initial input features to a latent space.

Our Autoencoder consists of the encoder and decoder layer. Each consists of 3 Linear layers of dimensions [][][]. The model will be trained to output a decoding as close to the initial input as possible.

After training we will use the output of its final encoder layer as the final combined feature representation vector, mapping the initial [][][ features of each song to a latent vector of shape 32. Each entry is then normalized with a standard scaler for noise pro

## 3. Similarity

After transforming our song database to a new latent representation, we can now infer our model to recommend songs. To do this, we can use distance metric methods such as cosine similarity and Euclidean distance. For this projects, we took advantage of cosine similarity from the sklearn library. Cosine similarity computes the similarity between two songs represented by an embedding arrays X and Y as the normalized dot product of X and Y:

K(X, Y) = <X, Y> / (||X||\*||Y||)

To get the recommendation on a requested song, the database is queried to find its embedding representation based on is name. We then perform cosine similarity to find the 5 songs that are closer in regards to embeddings distance from the whole dataset and return their name to the log output.

For evaluating the final recommendation accuracy, we will use manual evaluation for each requested song. The experiments of inference are displayed in the below section.

## 4. Experiments

## 5. Conclusion

The songs recommended are most of the times belong to the same genre. This is expected as we trained the model based on the notion that two songs are similar to each other if they belong to the same latent representation space that consists one genre class.

## 6. References

[1] GTZAN Music Genre Classification Dataset, <https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification>

[2] Million song dataset + spotify API + last.fm dataset, <https://www.kaggle.com/datasets/undefinenull/million-song-dataset-spotify-lastfm/data>

[3] “How I taught a neural network to understand similarities in music audio” ,

<https://medium.com/@silvercloud438/how-i-taught-a-neural-network-to-understand-similarities-in-music-audio-d4fca54c1aed>

[4] Choi, K., Fazekas, G., Cho, K., & Sandler, M.B. (2017). A Tutorial on Deep Learning for Music Information Retrieval, <https://www.semanticscholar.org/reader/6d2b42413839130060cec5c4505a3cb2d15abec7>