

MSc in Artificial Intelligence NCSR Demokritos & University of Piraeus

Song similarity retrieval

Multimodal Machine Learning

Students: Yiannis Savvas, Chaido Poulidou



Sections

1 Introduction.....	2
2 Architecture pipeline.....	3
2.1 Datasets.....	4
2.2 Classifier models.....	4
2.3 Metadata.....	8
2.4 Autoencoder.....	9
3. Similarity.....	9
4. Experiments.....	9
5. Conclusion.....	11
6. References.....	12

1 Introduction

In this project, we deal with the issue of a song recommendation system based on closest similarity metric in 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 like collaborative filtering, we have decided to utilize representation learning with two modalities: the song's raw audio and text metadata. To further enrich and refine our representations, we use both supervised and unsupervised feature embeddings with the use of a CNN and Autoencoder architecture.

We explain our data preprocessing in section 2, where we used audio feature extraction to extract the Mel-spectrograms that are used to train the supervised model. In section 3, we show how we used a Convolutional Neural Network (CNN) for genre classification on the GITZAN dataset based on these Mel-spectrograms. Afterwards, we use the CNN to obtain the learned embeddings for relevant genre representation. We also extract deep audio features to classify DEAM dataset for emotion detection and obtain learned embeddings as well. Furthermore, in section 4, we will present how we additionally took advantage of further metadata and an Autoencoder architecture for a mixed representation for all songs in a database that is comprised of 1500 songs from the million-song dataset.

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 and will recommend the top 5 closest songs.

2 Architecture pipeline

Firstly, we need to specify what we consider as similarity in songs. Similarity metrics do not only exist in the audio input features, but in different characteristics of each song, like its genre or the instruments and emotion. The quantization of similarity is after a base point subjective. In our project, we will consider it as a factor of metadata features and music genre combined with emotion.

The idea behind our approach comes from the ability of the autoencoder to perform effective dimensionality reduction by capturing the essential features in a lower-dimensional latent space. This is beneficial for reducing the complexity of data and speeding up subsequent processing tasks. Another benefit is that we can combine features from different modalities and learn robust representations that are less sensitive to noise and variations in the data, since unsupervised learning is utilized. In our project, we will use both methods sequentially, obtaining a final general robust representation from different supervised tasks.

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:

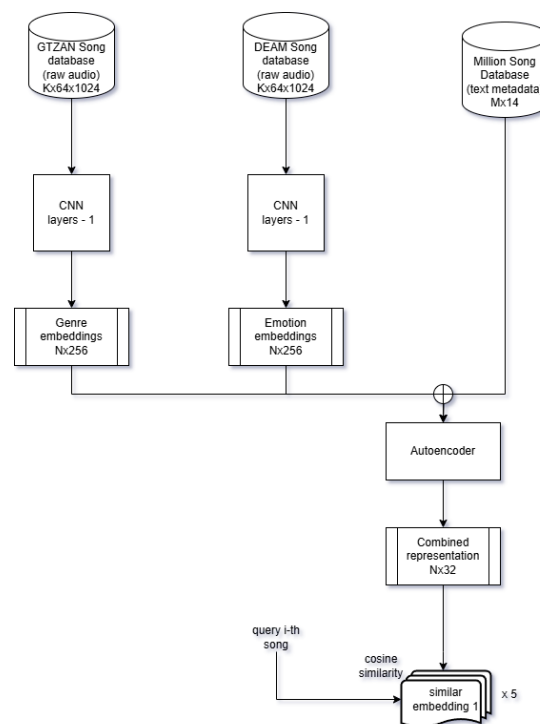


Figure 0: Song representation retrieval pipeline

In the following chapters, we further discuss the nature of 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-spectrograms. We choose to use the mel-spectrograms of each audio file, because it contains a discernible way to represent a song's time and frequency domain features. We will use these mel-spectrograms for the CNN genre classification mentioned in the following section. For this reason, we concluded to using the GTZAN dataset from Kaggle [1], which contains raw audio input of 10 genres with 100 audio clips each, so we have balanced dataset. Each song's duration is 30 seconds and divided per genre in different folders.

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

For the emotion representation, we used the DEAM dataset [5], which contains both raw audio input files with their labeled data. Each song is annotated either on short term feature length by 500 ms or on the whole song, using averaged values. The dataset is also labeled to valence and arousal values by experts. Based on the range of these labels, we can extract the given emotion on the whole song. Out of the 2058 annotations, only 1802 were consistent in both audio and metadata entries. Further preprocessing is needed and explained in later sections.

Finally, 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 [2]. Further details are explained in relevant section.

2.2 CNN genre classifier

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

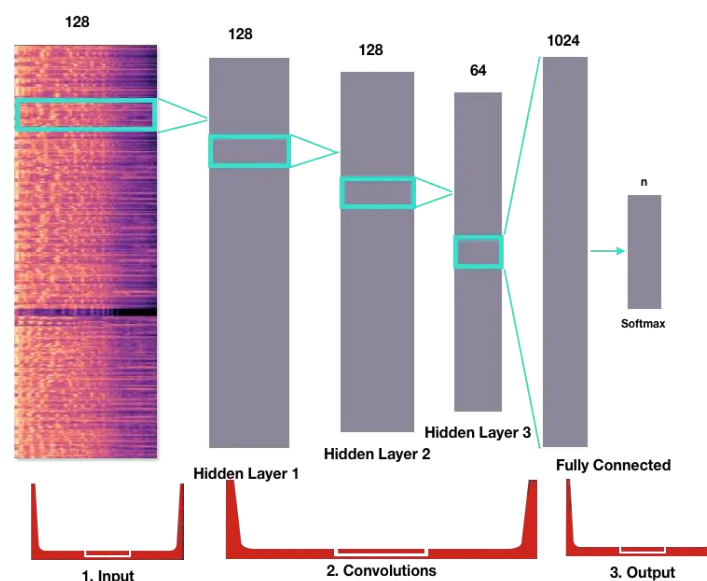


Figure 1: Implementation of CNNs for classification on mel-spectograms. Source [3].

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 be similar in general.

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, hip-hop, country, pop

Since our dataset is already balanced with 100 songs per class, we use Stratified split of 80-10-10 to create the training, validation and test dataset.

2.2.1 Representation output

We train our CNN in batches. After training, we obtain the representations of genre dependencies regarding input mel-spectograms.

Our training dataset input dimensions are 64x1024 (number of mel bins x feature length) and the final embedding layer of the CNN gives latent embedding of dimension size 256. Essentially, the last Fully Connected 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.

2.2.2 Evaluation

We trained our CNN for 100 epochs for the whole training dataset. We used a best model approach, where we saved the final model to be the one that had the lowest validation loss in the whole epochs span.

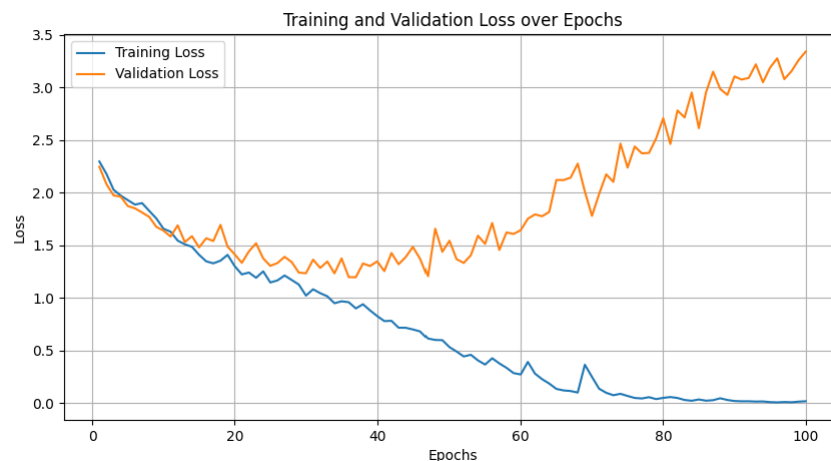


Figure 2: CNN Genre classifier training and validation loss, 100 epochs

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 and validation loss decreases steadily and consistently over the epochs until a certain epoch, from which overfitting is starting to occur. In Figure 3, the Confusion Matrix of the test set is presented. The test set was completely hidden in the whole training-evaluation procedure.

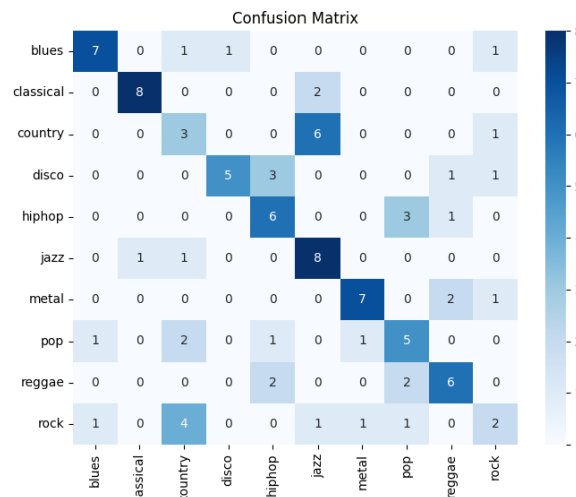


Figure 3: CNN genre classification confusion matrix.

From Figure 3, it is evident that the model classifies well 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 CNN emotion classifier

To further enrich the similarity search space, we also trained a CNN to perform emotion classification on the DEAM music dataset. The dataset contains 1802 audio samples with a duration of 45 each. From the given dataset, we used only 1744 of the samples due to incorrect and incomplete annotation.

It also provides us with labeled metadata on arousal and valence values per song. We used the song average values of valence and arousal that has been annotated by experts. Valence and arousal are audio properties closely related to emotion. Based on Thayer's theory, depending on the combination of values of valence and arousal of an audio signal, we can determine it's emotion.

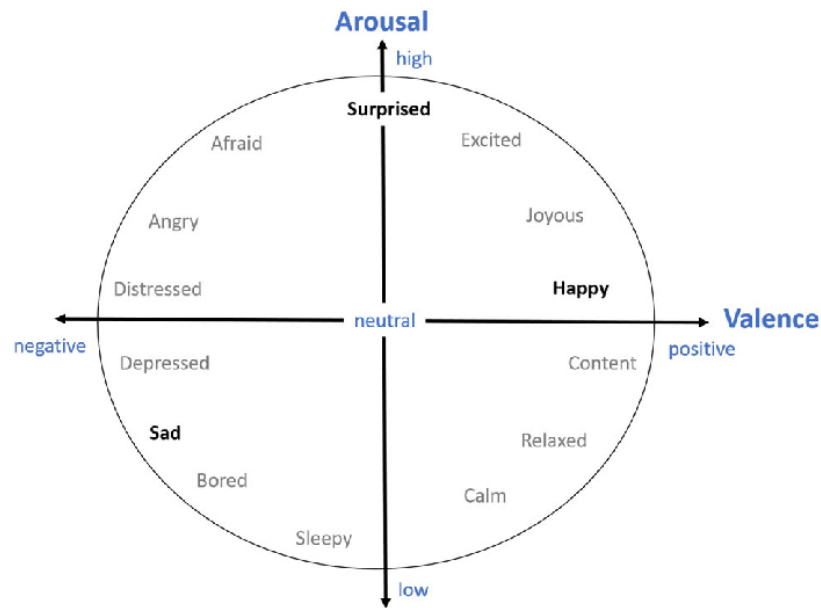


Figure 4: Range of arousal and valence values to given emotions inferred.

Based on the Thayer's model, we set 4 classes / emotions. Since the values of arousal and valence in the provided DEAM annotations are in range 1-9, we set a threshold of value 5 and create a new column 'emotion' based on the values of valence and arousal around that threshold. The emotions/labels created are:

- **happy**: Valence > 5 and Arousal > 5
- **calm**: Valence > 5 and Arousal < 5
- **anger**: Valence < 5 and Arousal > 5
- **sad**: Valence < 5 and Arousal < 5

The total count of songs per emotion/label is seen in Figure 5. It is evident that it is an unbalanced dataset and can lead to false negative classification for the weaker classes. To combat this problem, we used librosa on the weak classes "calm" and "angry" to create additional augmented files with pitch shift up, pitch shift down and noise addition.

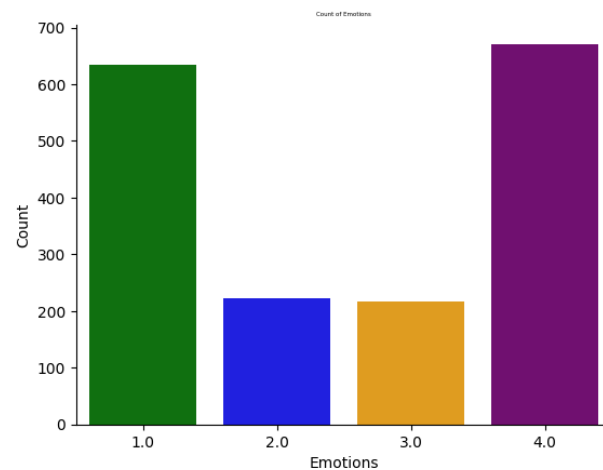
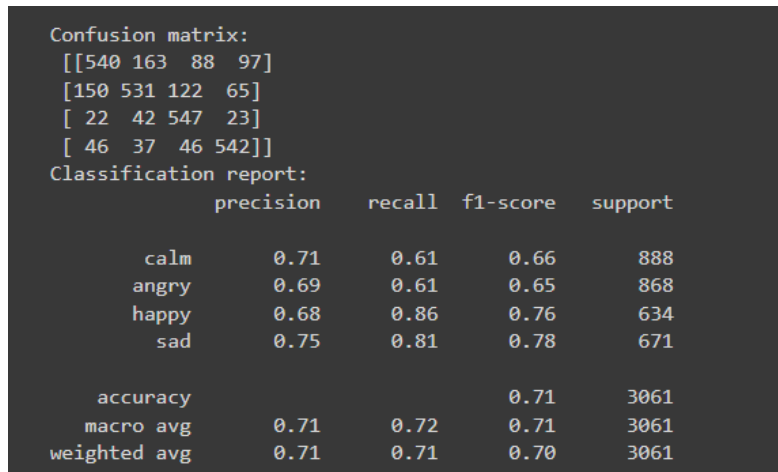


Figure 5: Count of DEAM instances per emotion.

2.3.1 Training and representation output

For training the model, we used the python audio classification library `deep_audio_features` [6]. This library uses a CNN internally for supervised classification, given an input dataset of audio files divided per class in folders.

We also used the `deep_audio_features` library for the model evaluation. The classification report after training on the test dataset can be seen in Figure 6.



```
Confusion matrix:
[[540 163  88  97]
 [150 531 122  65]
 [ 22  42 547  23]
 [ 46  37  46 542]]
Classification report:
```

	precision	recall	f1-score	support
calm	0.71	0.61	0.66	888
angry	0.69	0.61	0.65	868
happy	0.68	0.86	0.76	634
sad	0.75	0.81	0.78	671
accuracy			0.71	3061
macro avg	0.71	0.72	0.71	3061
weighted avg	0.71	0.71	0.70	3061

Figure 6: Classification report CNN emotion classifier.

The representations were collected by inferring the trained classifier and obtaining the last fully connected layer of the classification. The final emotion embedding is of length 256.

2.3 Metadata

To embed further information about the songs, we will try to also use metadata information based on the Million Song Dataset [2]. 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:

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 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. The input data is the concatenation of CNN embeddings created by the trained classifiers and the metadata of each of the 1500 entries, so final data dimensions of $N \times 526$ are inputted to the network.

Our Autoencoder consists of the encoder and decoder layer. Each consists of 3 Linear layers of output dimensions 248, 128, 64 and 128, 248 and 526 respectively. The model will be trained to output a decoding as close to the initial input as possible. For training, we used a standard MSE loss with Adam optimizer.

After training we will use the output of its final encoder layer as the final combined feature representation vector, mapping the initial 526 features of each song to a latent vector of shape 64. Each entry is normalized using Standard Scaler. Also 20 songs were kept out of the training procedure to be used afterwards for querying song similarities.

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 project, 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 :

$$K(X, Y) = \frac{\langle X, Y \rangle}{(|X| * |Y|)}$$

To get the recommendation on a requested song, we query the 20 songs left out of the autoencoder training. We then perform cosine similarity to find the 5 closest songs regarding embeddings distance from the whole dataset and return their name. The higher the similarity distance, the more common features the songs have.

For evaluating the final recommendation accuracy, we will use manual evaluation for each requested song, since quantifying the accuracy for this issue poses a complex problem and might require training a different Deep Learning model. The experiments of inference are displayed in the below section.

4. Experiments

For each experiment, we try to manually evaluate resemblance for the 20 different samples. We will request the representation of 20 random different songs from the dataset and ask for the top 5 similar songs. The model's performance will be manually assessed on whether the recommended 5 songs are close to the requested song's

genre, song's emotion and whether it has any similar acoustic features. This will showcase the model's ability to capture genre and emotion similarity, which means it is able to give good low-dimensional representations as trained.

In this section we will only present you with 3 out of 20 experiments. The rest of the results can be found in the "Top_5_songs.csv" in our GitHub:

Experiment 1:

Input query song: "Marie-ELLEGARDEN"

The song is of punk genre with high valence and arousal/energy metrics, labelled with happy emotion.

From executing the query, the recommended 5 songs closer to this are:

Marie	The Violence
Marie	Love Steals Us From Loneliness
Marie	Plagues
Marie	Ya Can't Go Home
Marie	Breathing

Taking a look on the closest recommended song labels in the initial annotated dataset, we see that all songs recommended belong to alt punk or rock genre. After evaluating the songs by listening, we can confirm that they also share the same chord progressions and thus emotions to one another. There are generally other features that are similar after listening, like tempo, signature and similar instruments are detected among all songs. This shows that, even if our genre classifier struggled in rock song classification, the fact that we enrich our songs embeddings with further information helps the Autoencoder to learn better underlying dependencies taking advantage of the further features.

Experiment 2:

Input query song: "Beni Beni – Niyaz"

The song is of World genre, specifically of more traditional oriental origins.

Recommended songs:

Beni Beni	Tamatant Tilay
Beni Beni	Sans Toi
Beni Beni	Still Around
Beni Beni	Choctaw Hayride
Beni Beni	It Beats 4 U

Again, the strongest characteristic of genre is greatly considered by the representations, as similar genres are selected. In particular, the most recommended song is also of oriental origin which shows how well the model understands the same undertones. All the songs after listening exhibit similar tempo, instruments and loudness, while emotion is also shown to be close.

Notice how even if the last suggested song “It beats 4 U” is instead indie-rock, it shares again similar audio features and energy.

Experiment 3:

Input query song: “The Past Should Stay Dead – Emarosa”

The song belongs to the hard-rock genre.

Recommended songs:

The Past Should Stay Dead	The Violence
The Past Should Stay Dead	No One Really Wins
The Past Should Stay Dead	Deliverance
The Past Should Stay Dead	Bragarful
The Past Should Stay Dead	Leaving

The top 5 songs selected belong in the rock-metal genre. Heavy guitar riffing is very common in all of 5 songs selected and in the query song. This justifies the selection and proves the robustness of the autoencoder to combine the different characteristic that rock music provides.

5. Conclusion

The songs recommended 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.

While single CNN embeddings are highly effective for tasks like image classification and object detection, autoencoders offer a complementary approach for learning robust, unsupervised representations that can generalize well across different tasks and datasets. This is seen from our experiments, where it is also evident that the use of an Autoencoder to compress information is effective and useful for high-dimension input. Our approach also manages to give accurate results while maintaining low computational cost, something that cannot be achieved with single supervised models.

6. Future Work and optimization

From our implementation, we saw both benefits of supervised embeddings and unsupervised learning. It is evident that the model could benefit on further usage of unsupervised mechanisms to catch unseen feature dependencies that lead to similarity. To further widen this area of no supervision, we could search for ways to find metrics that are not yet in our input dataset.

One way we can learned metric spaces is by using Siamese networks. With a Siamese network, the model is trained to learn a custom metric space that better captures the

similarities between songs, given pairs of similar and dissimilar examples. It uses a similar pair of networks with shared weights and map input data to a latent space, where similar input data are mapped close to each other and dissimilar further away. To create similar embeddings for the close data, it uses contrastive loss function.

An easier method that relies on the way we separate the data before applying cosine similarity, is to use a clustering method in the latent representation space to get close embeddings in the same cluster. This will help in more efficient searching since the search only occurs on the created clusters.

Our github link for the project is <https://github.com/johnsaveus/Multimodal>.

7. 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/undefinenu/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>
- [5] DEAM dataset (Music emotion classification)
<https://www.kaggle.com/datasets/imspars/deam-mediaeval-dataset-emotional-analysis-in-music>
- [6] Deep-audio-features GitHub,
https://github.com/tyiannak/deep_audio_features