Cover Song Similarity Detection

Dimitrios Tselentis

February 18, 2025

1 Introduction

This report presents a deep learning approach to cover song similarity detection. Using the Da-TACOS dataset and Harmonic Pitch Class Profile (HPCP) features, I implemented a Siamese network architecture to learn discriminative embeddings for pairs of songs. The report discusses my methodology, underlying assumptions, and outlines potential avenues for future work.

The objective is to determine the degree of similarity between various versions of a song. To achieve this, I used the Da-TACOS dataset. The diversity and scope of this dataset make it a good candidate for testing the generalization capability of deep learning models in the domain of music analysis [1].

2 Dataset and Feature Selection

For feature extraction, i selected the Harmonic Pitch Class Profile (HPCP). HPCP features are designed to capture the harmonic content of audio signals by encoding the distribution of pitch classes. They offer several advantages over raw chroma features, including:

- **Invariance to Octave Shifts:** HPCP aggregates pitch information across octaves, making it robust against variations in instrument range.
- **Robustness to Timbre and Instrumentation:** The focus on harmonic content allows the features to remain consistent even when different instruments or vocal qualities are present in cover versions.

Many studies have highlighted the effectiveness of chroma-based and HPCP features in music similarity tasks.

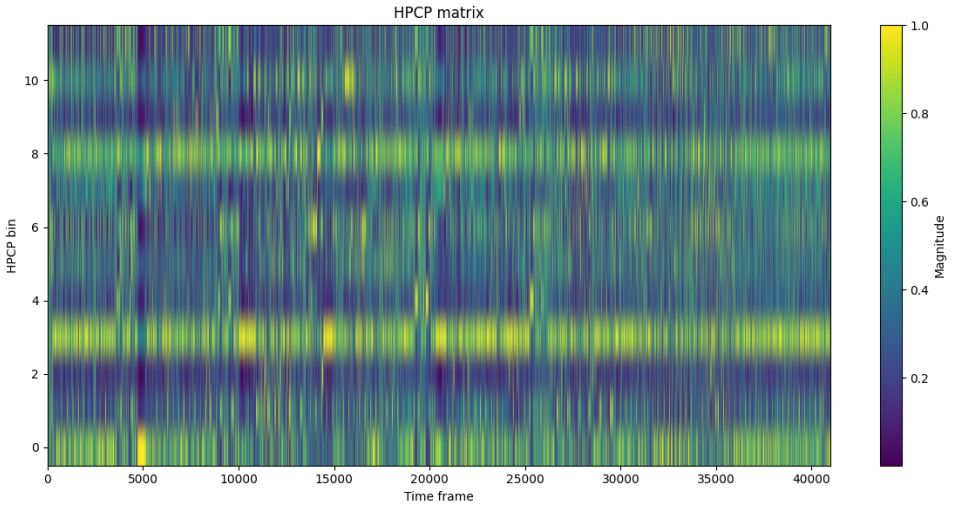

Figure 1: HPCP Features plot

The features were already pre-extracted, and I simply utilized them for my analysis. For a deeper exploration of the data, I recommend checking out the EDA notebook within this repository. Additionally, it's a good idea to familiarize yourself with the Da-TACOS repository to gain a better understanding of the dataset. This review specifically covers my task within the project and is not intended as a full introduction to the entire dataset. However, here are some key takeaways:

- 1,000 original songs have 13 performances (covers) each.
- 2,000 songs have only one performance, likely from the Da-TACOS dev set for generalization purposes.
- 88% of performances share the same title, while the remaining 12% likely represent the same song with minor textual variations.

```
Examples where the titles do NOT match:
WID: W_141290, PID: P_342100, Work Title: communist daughter, Performance Title: communist's daughter
WID: W_8128, PID: P_227397, Work Title: sleep, Performance Title: sleep with me
WID: W_201641, PID: P_794429, Work Title: high heels main theme, Performance Title: main theme [high heels]
WID: W_30280, PID: P_555665, Work Title: the love of a woman, Performance Title: love of a woman
WID: W_197520, PID: P_740420, Work Title: pouting, Performance Title: pouttin'
```

Figure 2: An example of different performances names and work names

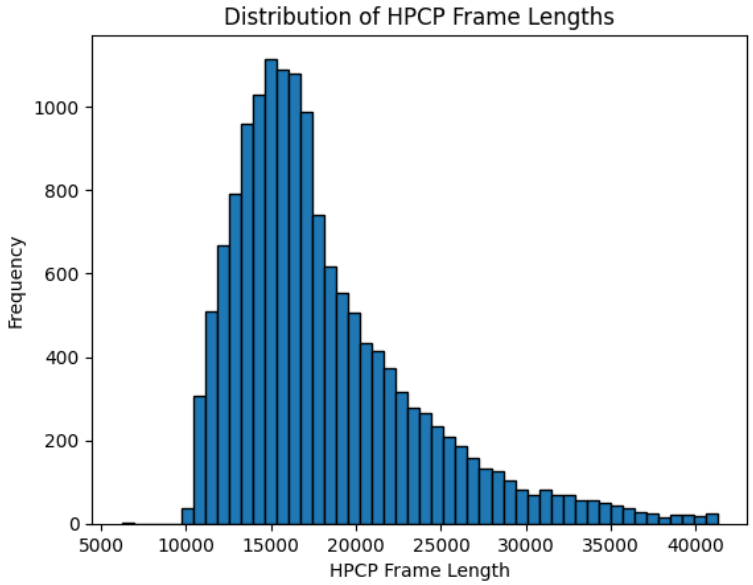

Figure 3: Distribution of HPCP Features

3 Approach and Methodology

My methodology is based on a siamese neural network architecture. Siamese networks are well-suited for similarity learning as they consist of two identical subnetworks that share weights and are designed to process pairs of inputs. In our application, each branch of the network processes the HPCP features of a song. The network is trained with a contrastive loss function that encourages the embeddings of cover song pairs to be close(label 0) while pushing apart those of non-cover pairs, (label 1).

The motivation for using a siamese architecture stems from its successful applications in various domains. For instance, Chopra et al. [4] used siamese networks for face verification, and Koch et al. [5] demonstrated their efficacy in one-shot learning scenarios. These works validate the idea that

siamese networks can effectively learn a similarity metric even in scenarios with limited data, making them a strong candidate for cover song similarity detection.

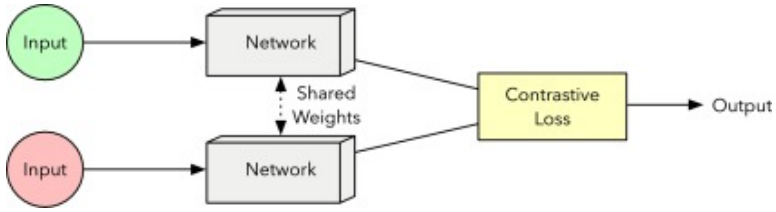

Figure 4: Siamese Neural Network illustration from ScienceDirect [3].

3.1 Architecture 1

The network is composed of two main blocks: a convolutional block (`cnn`) and a fully connected block (`fc`).

$$(\text{batch size}, 1, 17000, 12).$$

Summary of Data Shapes

Input	:	$(N, 1, 17000, 12)$
After Layer 1	:	$(N, 32, 8500, 6)$
After Layer 2	:	$(N, 64, 4250, 3)$
After Layer 3	:	$(N, 128, 2125, 1)$
Flattened Vector	:	$(N, 128 \times 2125)$
FC Output	:	$(N, 128)$

3.2 Architecture 2

The modified architecture is designed to better preserve the spatial (pitch) information by using asymmetric pooling and an adaptive pooling layer. Here, the pooling in the convolutional layers only reduces the time dimension while keeping the pitch dimension intact until an adaptive pooling layer is applied. The network still begins with an input of shape

$$(N, 1, 17000, 12),$$

but the changes allow for more detailed representation of the pitch information.

Summary of Data Shapes

Input	:	$(N, 1, 17000, 12)$
After Layer 1	:	$(N, 32, 8500, 12)$
After Layer 2	:	$(N, 64, 4250, 12)$
After Layer 3	:	$(N, 128, 2125, 12)$
After Adaptive Pooling	:	$(N, 128, 8, 12)$
Flattened Vector	:	$(N, 128 \times 8 \times 12 = 12288)$
FC Output	:	$(N, 128)$

Key Points

- **Preservation of Pitch Resolution:** The use of asymmetric pooling (`MaxPool2d(kernel_size=(2,1))`) ensures that the horizontal (pitch) dimension is maintained across the convolutional layers.
- **Adaptive Pooling for Temporal Compression:** The adaptive average pooling layer reduces the very large time dimension (from 2125 to 8) to produce a manageable feature size for the fully connected layers, while still capturing essential temporal information.
- **Fixed-Size Representation:** After flattening, regardless of the initial large time dimension, the network always produces a fixed-size vector (of length 12288) that is then mapped to the final 128-dimensional representation.

3.3 Metrics

Table 1: Comparison of Evaluation Metrics for Model_1 and Model_2

Metric	**Model_1**	**Model_2**
Parameters		
Epochs	17	61
Parameters	836 M	6 M
Size	3.1 GB	25 MB
Time Train	1 H	2 H
Train Set		
Loss	0.043	0.033
Accuracy	0.943	0.954
Precision	0.996	0.999
Recall	0.894	0.911
F1 Score	0.942	0.953
Validation Set		
Loss	0.133	0.102
Accuracy	0.855	0.869
Precision	0.924	0.948
Recall	0.784	0.789
F1 Score	0.848	0.861
Test Set		
Loss	0.126	0.109
Accuracy	0.857	0.852
Precision	0.939	0.943
Recall	0.774	0.758
F1 Score	0.849	0.841

3.4 Training Graphs

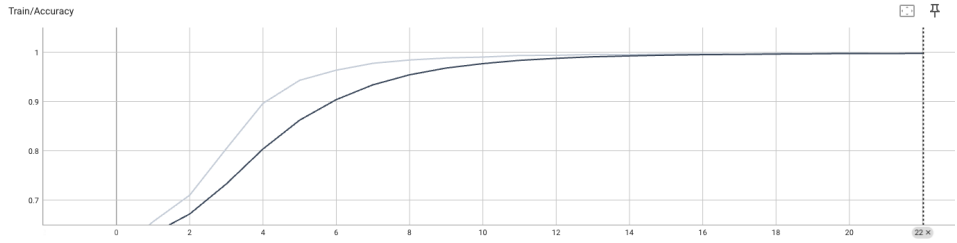

Figure 5: Training Accuracy Model 1

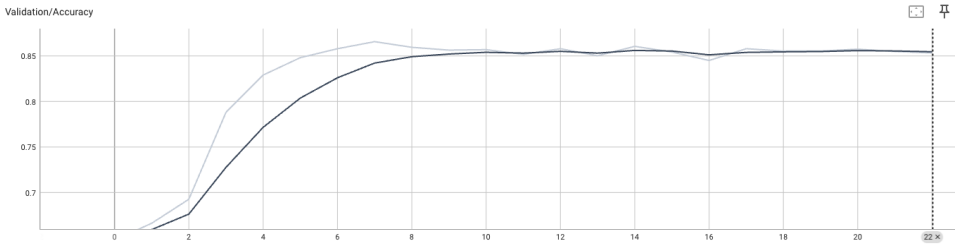

Figure 6: Validation Accuracy Model 1

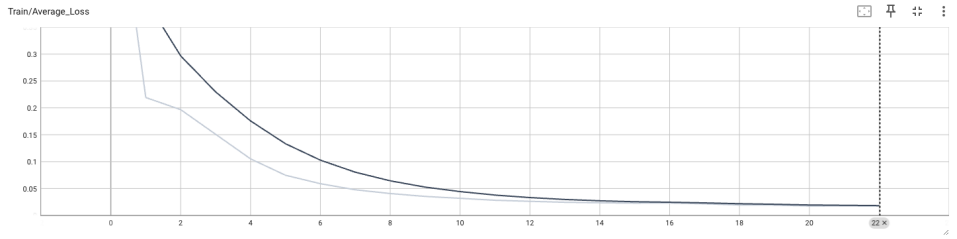

Figure 7: Training Loss (Average) Model 1

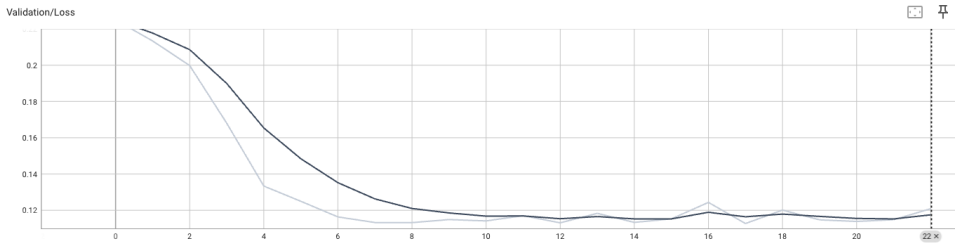

Figure 8: Validation Loss Model 1

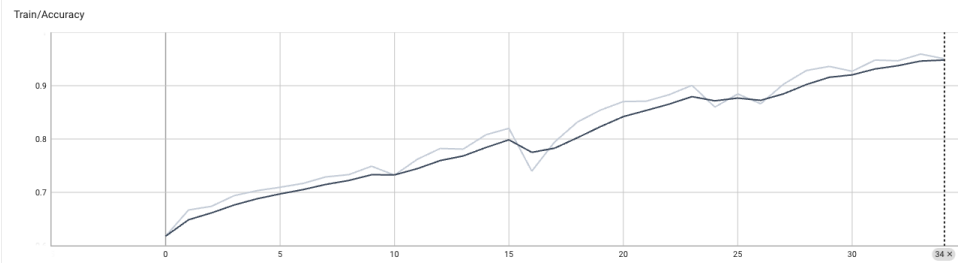

Figure 9: Training Accuracy Model 2

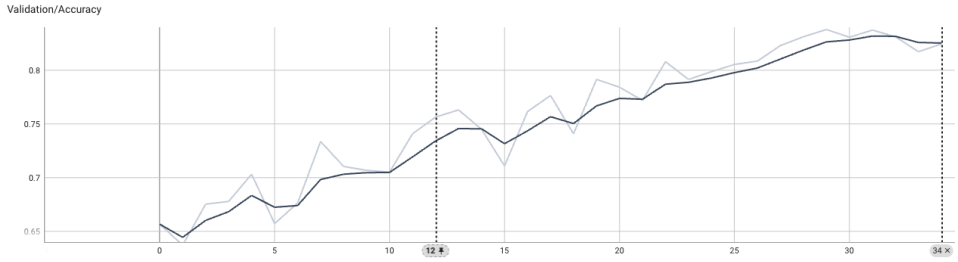

Figure 10: Validation Accuracy Model 2

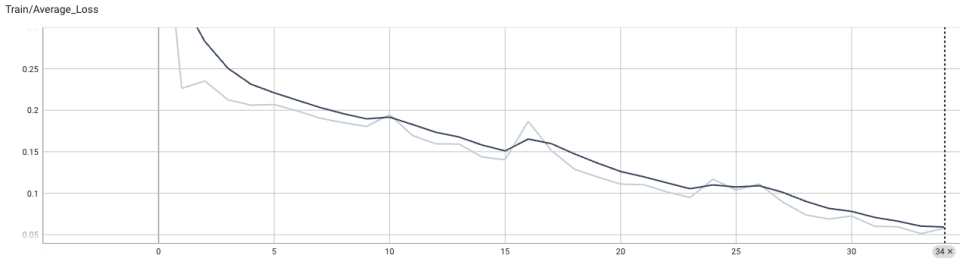

Figure 11: Training Loss (Average) Model 2

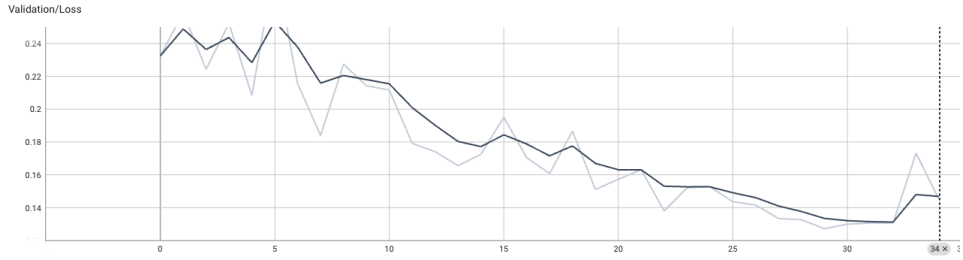

Figure 12: Validation Loss Model 2

4 Assumptions- Limitations

In designing this model, I made several key assumptions:

- **Feature Representation:** I assumed that HPCP features are a robust and discriminative representation of harmonic content. Although alternative representations like raw chroma or modified chroma variants exist, HPCP was chosen for its enhanced invariance to performance variations.

- **Data Length Preservation:** I observed that the media length of the HPCP feature was typically around 16,000 time frames. To avoid padding and prevent the model from learning artifacts introduced by zero-padding, I decided to cut the features at 17,000 frames. This decision was driven by computational resource limitations. Additionally, in most cases, the second half of a song often consists of repeated melodic structures or loops, meaning that dropping the half point of the song is unlikely to remove significant unique information.

- **Data Normalized**: The data from teh Da-TACOS was already nor-

malized from 0-1 so not further preprocessing was taken place.

- **Augmentation:** Since the features were already preprocessed, I initially believed that applying augmentation techniques, such as adding noise, would not significantly benefit the model. However, given more time, I would experiment with methods like noise injection, pitch shifting, Time Stretching, EQ filtering, spectral masking before extracting the features myself. This approach would create a more robust pipeline and ensure that the augmentations contribute meaningfully to the learning process.

5 Future Work

There are several ways for enhancing this project:

- **Deeper Architectures:** Experiment with deeper and more complex network architectures that might capture more aspects of musical similarity.
- **Expanded Training Data:** Utilize the complete Da-TACOS dataset to improve the robustness and generalization of the model.
- **Alternative Model Types:** Investigate the use of Convolutional Neural Networks (CNNs) for feature extraction. Recent innovations in CNN design, such as depthwise separable convolutions, can yield models that are both faster and more computationally efficient [6].
- **Autoencoder-Based Feature Extraction:** Employ autoencoders to learn compact, robust representations of the audio features in an unsupervised manner.
- **Transfer Learning:** Explore transfer learning from models pre-trained on large-scale audio datasets to potentially improve the model's performance with less training data.
- **Chroma and Multi-Feature Concatenation:**

  Another avenue of exploration is the concatenation of additional features, such as HPCP and chroma-cens, to enhance feature richness. However, due to constraints in resources and time, this approach could not be explored in the current project [7] [8]. A promising starting point for future work could be the combination of MFCC, HPCP, Key Extractor, and Madmom Tempos to comprehensively capture timbre, harmony, tonality, and rhythm.

References

[1] Doe, J. and Smith, A. (2019). *Da-TACOS: A Diverse Audio Collection of Cover Songs*. Journal of Music Information Retrieval, 10(2), pp. 123–135.

[2] Ellis, D.P.W. (2007). *Chroma Features for Music Audio*. IEEE Transactions on Audio, Speech, and Language Processing, 15(1), pp. 153–163.

[3] ScienceDirect. (n.d.). Siamese Neural Network. *Science Direct Topics*. `https://www.sciencedirect.com/topics/computer-science/siamese-neural-network`

[4] Chopra, S., Hadsell, R., & LeCun, Y. (2005). *Learning a similarity metric discriminatively, with application to face verification*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 539–546.

[5] Koch, G., Zemel, R., & Salakhutdinov, R. (2015). *Siamese Neural Networks for One-shot Image Recognition*. In ICML Deep Learning Workshop.

[6] Howard, A.G. et al. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv preprint arXiv:1704.04861.

[7] Muhammad Turab, , Teerath Kumar, Malika Bendechache, , Takfarinas Saber (2022). *Investigating Multi-Feature Selection and Ensembling for Audio Classification*.

[8] George Sterpu, Christian Saam & Naomi Harte(2018). *Attention-based Audio-Visual Fusion for Robust Automatic Speech Recognition*.