Deep Metric Learning for Music Information Retrieval

By

Vasileios Mouchakis

Submitted
in partial fulfilment of the requirements for the degree of

Master of Data Science

at the

UNIVERSITY OF PELOPONNESE

October 2023

Abstract

This master thesis explores the application of Deep Metric Learning (DML) for creating effective audio representations in tasks like audio classification, music retrieval, and speech recognition. DML uses deep neural networks to learn hierarchical representations from raw audio waveforms, capturing intricate relationships between audio samples. The thesis evaluates different deep neural network architectures and loss functions, including triplet loss and contrastive loss. The models are tested using various distance metrics and normalization techniques. The research aims to enhance our understanding of DML for audio representations and its potential applications. The findings contribute valuable insights to guide the design of powerful audio representations for diverse audio-related tasks.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Researcher B' Theodoros Giannakopoulos, for his invaluable guidance, support, and expertise throughout the entire duration of my master's thesis. His insightful feedback and unwavering encouragement have been instrumental in shaping the direction and quality of my research. I would also like to extend my appreciation to Panagiotis Koromilas and Sandy Bei for their valuable contributions, insightful discussions, and assistance during the various stages of my thesis. Additionally, I would like to acknowledge the Multimedia Analysis Group of the Computational Intelligence Lab (MagCIL) at the National Center for Scientific Research “Demokritos" for providing me with an intellectually stimulating environment and access to resources that greatly enriched my research. Their collaborative spirit and shared knowledge have been invaluable in shaping my understanding and enhancing the quality of my work.

Thesis Supervisor: Theodoros Giannakopoulos
Title: Researcher B’ (Demokritos)

Table Of Contents

1. Introduction

In recent years, there has been a surge of interest in developing powerful and discriminative audio representations for various applications, including audio classification, music retrieval, speech recognition, and audio-based recommendation systems. Deep Metric Learning (DML) has emerged as a promising approach to address the challenge of learning effective representations that capture the underlying semantic structure of audio data.
Deep Metric Learning aims to learn a metric space where the similarity between audio samples is explicitly modeled. Unlike traditional approaches that rely on handcrafted features or shallow representations, DML leverages deep neural networks to automatically learn hierarchical representations from raw audio waveforms. By exploiting the rich hierarchical structures within audio data, DML enables the discovery of intricate relationships and fine-grained similarities between audio samples.
The primary objective of DML is to learn a representation space where semantically similar audio samples are projected closer together, while dissimilar samples are pushed further apart. This facilitates various downstream tasks such as audio retrieval, clustering, and classification, by providing a compact and discriminative representation that captures the intrinsic properties of the audio data.
One of the key challenges in DML for audio lies in designing appropriate loss functions that can effectively measure the similarity or dissimilarity between audio samples. Various loss functions have been proposed, such as triplet loss, contrastive loss, and angular loss, each with its own strengths and limitations. These loss functions aim to optimize the embedding space by explicitly enforcing the desired proximity relationships among audio samples.
Furthermore, the choice of the deep neural network architecture plays a crucial role in DML for audio. Convolutional Neural Networks (CNNs) have been widely adopted due to their ability to capture local and global dependencies in audio signals. Recurrent Neural Networks (RNNs) are also employed to model temporal dependencies, particularly in tasks involving sequential audio data, such as speech recognition.
This thesis aims to investigate and explore the efficacy of different DML techniques in the context of audio representations. Specifically, we will analyze the performance of various deep neural network architectures and loss functions in learning discriminative audio embeddings. We will evaluate these representations on benchmark audio datasets and compare them against state-of-the-art methods to assess their effectiveness in audio-related tasks.
Overall, this research contributes to the growing field of Deep Metric Learning for audio representations and aims to enhance our understanding of the underlying principles and techniques involved. The findings from this study will not only provide valuable insights into the design and optimization of audio representations but also have the potential to advance audio-related applications in diverse domains.

1.1. Motivation

The motivation behind this thesis stems from the fundamental human desire to explore and uncover meaningful connections within the vast realm of music. Music holds a unique power to evoke emotions, transcend language barriers, and create profound experiences for individuals across cultures and backgrounds. With the ever-expanding digital music landscape, there arises a pressing need for advanced techniques that can effectively navigate and harness the wealth of musical content available to us. Deep Metric Learning (DML) emerges as a promising approach to address this challenge by enabling the development of machine learning models capable of understanding the intricate relationships between songs based on their underlying audio features. By leveraging the power of DML, we aim to create intelligent systems that can not only classify and categorize music but also provide personalized recommendations and facilitate novel music discovery experiences. By training ML models to accurately identify and retrieve the most similar songs to a given input, we strive to enhance the accessibility and enjoyment of music for both casual listeners and industry professionals alike. Furthermore, this research has the potential to contribute to a wide range of applications, including music recommendation systems, playlist generation, and content-based music retrieval, ultimately transforming the way we interact with and appreciate music in the digital age.

1.2. Related Work

In paper [1] a novel angular loss for deep metric learning is proposed. The angular loss is based on the idea of measuring the similarity between two feature vectors as the cosine of the angle between them. This makes the angular loss more robust to variations in scale and rotation than traditional distance-based losses. The paper also shows that the angular loss can achieve better performance than traditional distance-based losses on a variety of image classification and retrieval tasks.

Paper [2] presents a general framework for distance metric learning that is based on the large margin nearest neighbour classification (LMNN) algorithm. The LMNN algorithm learns a distance metric that maximizes the margin between the nearest neighbours of positive and negative examples. The margin is a measure of the separation between the two classes, and a larger margin indicates that the two classes are more well-separated. The paper shows that the LMNN algorithm can achieve good performance on a variety of classification tasks.

This paper [3] proposes a deep learning-based approach to face recognition and clustering. The FaceNet model learns a 128-dimensional embedding for each face image. This embedding is used to represent the face image in a high-dimensional space where faces that are similar in appearance are close together. The FaceNet model has been shown to achieve state-of-the-art performance on face recognition and clustering tasks.

In the pages of this publication [4], a new objective function for deep metric learning is proposed, called Deep InfoMax. Deep InfoMax is based on the idea of maximizing the mutual information between the representations of positive and negative pairs of examples. The mutual information is a measure of how much information one random variable contains about another random variable. The paper shows that Deep InfoMax can achieve good performance on a variety of metric learning tasks.

This research [5] proposes a simple framework for contrastive learning of visual representations called SimCLR. SimCLR is based on the idea of using a siamese network to learn to distinguish between augmented versions of the same image. The augmented versions of the image are created by applying random transformations to the image, such as cropping, flipping, and color jittering. The siamese network is trained to predict whether two augmented versions of the same image come from the same image or not.

Paper [6] proposes a new method for unsupervised visual representation learning called MoCo. MoCo is based on the idea of using a siamese network to learn to distinguish between positive and negative pairs of examples. The positive pairs are examples that are augmented versions of the same image, while the negative pairs are examples that are from different images. The siamese network is trained to predict whether two augmented versions of the same image come from the same image or not.

This research [7] proposes a novel loss function for siamese networks called N-pair loss. The N-pair loss is based on the idea of having a siamese network learn to distinguish between N positive pairs and N negative pairs of examples. The positive pairs are examples that belong to the same class, while the negative pairs are examples that belong to different classes. The N-pair loss is formulated as follows:

loss = sum(max(d(anchor, positive) - d(anchor, negative) + margin, 0))

where d is a distance metric, anchor is the anchor example, positive is the positive example, and negative is the negative example. The margin is a hyperparameter that controls the separation between the two classes.
The N-pair loss has been shown to be more effective than the triplet loss for siamese networks on a variety of metric learning tasks. The paper also provides a theoretical analysis of the N-pair loss, which shows that it can be used to learn a more discriminative embedding space for siamese networks.

In paper [8], a novel approach to few-shot learning is proposed, called the relation network. The relation network is a siamese network that is trained to learn a similarity function between pairs of

examples. The similarity function is then used to predict the class label of a new example, given a set of support examples.
The relation network is able to achieve good performance on few-shot learning tasks by learning to compare examples in a discriminative way. The paper shows that the relation network can outperform other few-shot learning methods on a variety of benchmark datasets.

Paper [9] proposes a novel approach to person re-identification called contrastive multiview coding. The contrastive multiview coding approach is based on the idea of using a siamese network to learn to distinguish between augmented versions of the same person. The augmented versions of the person are created by using different views of the person, such as front view, side view, and top view.
The contrastive multiview coding approach has been shown to be effective for person re-identification in challenging scenarios, such as when the person is partially occluded or when the lighting conditions are poor. The paper shows that the contrastive multiview coding approach can outperform other person re-identification methods on a variety of benchmark datasets.

1.3. Proposed Methodology

In this study, we aim to train deep learning models on audio representations for unsupervised learning in the context of music. The dataset comprises songs from diverse genres, providing a comprehensive representation of musical styles. The primary objective is to develop models capable of identifying similar audio samples from a given database when presented with a target audio input.
To achieve this, we will employ deep metric learning techniques. Specifically, we will explore loss functions that encourage the models to learn discriminative embeddings for audio representations. By optimizing these loss functions, the models will be trained to minimize the distance between embeddings of similar audio samples and maximize the distance between embeddings of dissimilar samples.
The training process will involve feeding the audio representations into deep neural network architectures. These architectures will be designed to learn hierarchical representations from the audio data. We will experiment with different network architectures and hyperparameters to identify the optimal configuration that yields the most effective similarity learning.
Overall, this proposed methodology aims to leverage unsupervised deep learning techniques on audio representations to develop models capable of finding similar audio samples from a database given a target audio input. By exploring various loss functions and training strategies, we expect to enhance the models' ability to capture and understand the underlying patterns and similarities within the diverse musical genres present in the dataset.

1.4. Next Sections

The next chapters are organized in the following way:
In chapter 2 we introduce various audio representations commonly used in deep learning applications. We discuss time and frequency domain features that form the inputs of our models, exploring their significance in capturing audio information effectively.
In chapter 3 we provide a theoretical background on deep learning models commonly employed in audio applications. We reference notable architectures found in literature to highlight their relevance in audio representation learning.
In chapter 4 we delve into distance metrics used for evaluating the similarity of audio representations. We explore the theoretical foundations of various distance measures.
In chapter 5 we discuss various audio augmentation techniques that enrich the training data and enhance the generalisation capabilities of our models.
In chapter 6 we present our methodology in detail. We outline the selection and implementation of loss functions used for deep metric learning with audio data. Additionally, we describe the data preparation process and the choice of model architectures used in our experiments. Furthermore, we discuss the training procedure and the evaluation metrics employed to assess the model's performance.

Finally, in chapter 7 we present our experiments. We describe the datasets used for training and testing the models, along with their characteristics. We present the results obtained from the evaluation of the models using various distance metrics and normalization techniques. The observations from these experiments are thoroughly discussed, providing insights into the models' performance and effectiveness in capturing audio similarities.

2. Background
2.1. Audio Representations

Audio representations [10] play a fundamental role in the analysis, processing, and understanding of audio signals. These representations aim to capture and encode the intricate characteristics of sound waves, enabling effective manipulation and interpretation of audio data. In recent years, with the rapid advancement of deep learning techniques, audio representations have garnered significant attention, driving breakthroughs in various audio-related tasks such as speech recognition, music information retrieval, and sound event detection.

2.1.1. Time-domain representations

Time-domain representations [11] capture the audio signal in its original waveform form. They represent the variation of the audio signal over time. Common time-domain representations include the raw audio waveform and its variations, such as amplitude envelopes or temporal features extracted using windowing techniques.

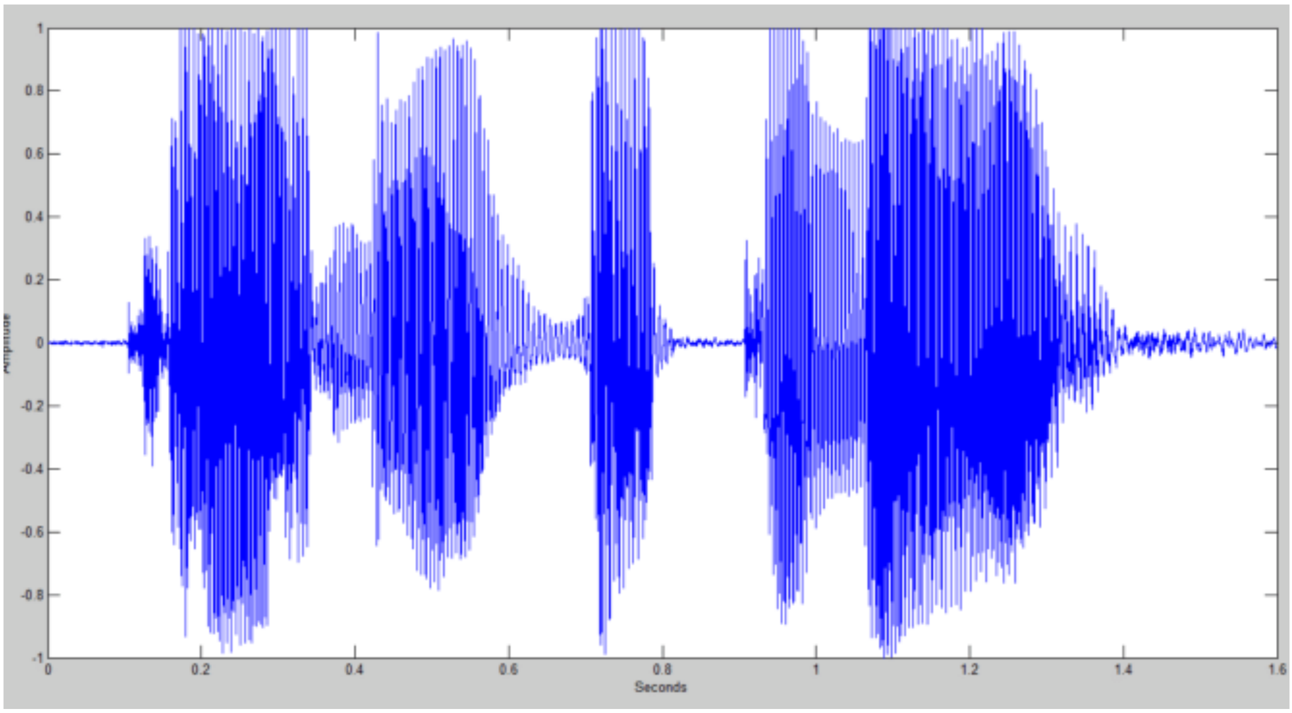

Figure 2.1 - Time domain representation of original .wav signal [12]

2.1.2. Frequency-domain representations

Frequency-domain representations [13] transform the audio signal from the time domain to the frequency domain. They provide information about the spectral content of the audio signal.

Examples of frequency-domain representations include spectrograms, which provide a visual representation of the frequencies present in the signal over time, and power spectral density (PSD) estimates, which represent the distribution of signal power across different frequencies.

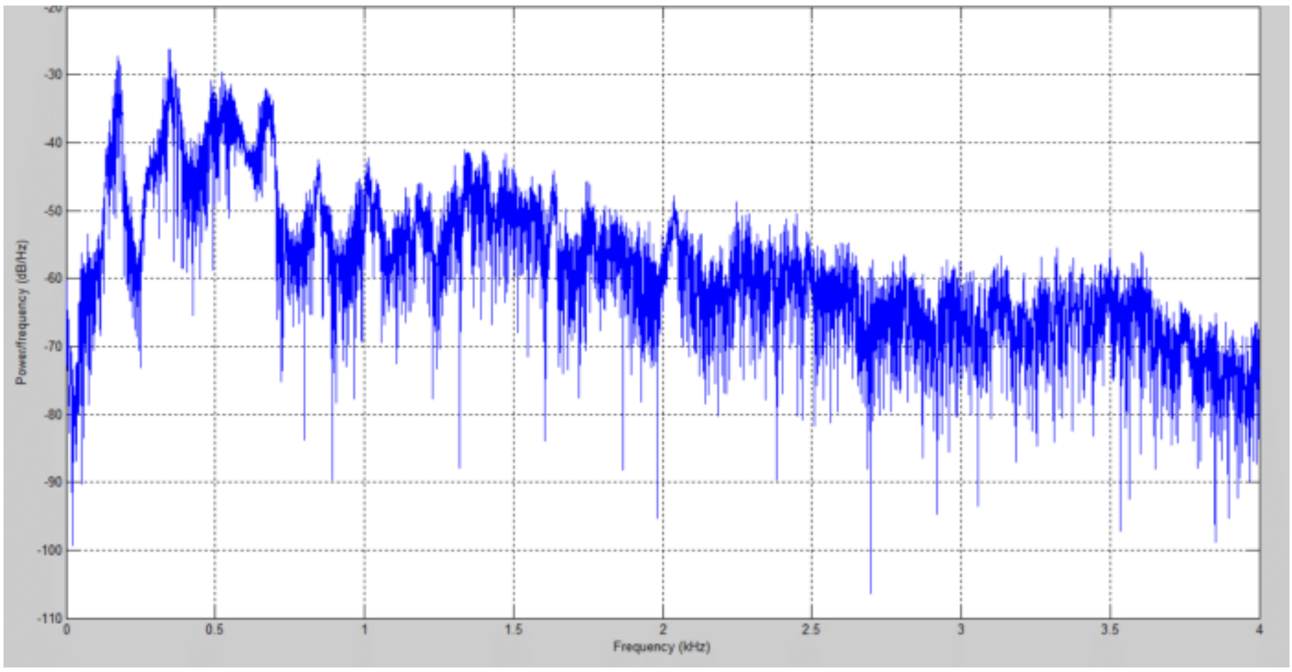

Figure 2.2 - Frequency domain representation of original .wav signal [12]

2.1.3. Mel-scale representations

Mel-scale representations [14] are based on the mel-frequency scale, which simulates the non-linear human perception of pitch. Mel-frequency cepstral coefficients (MFCCs) are a popular example of mel-scale representations. MFCCs capture the perceptual characteristics of audio by applying a series of transformations, including a Mel-scale filterbank and the Discrete Cosine Transform (DCT), to obtain a compact representation of the audio signal.

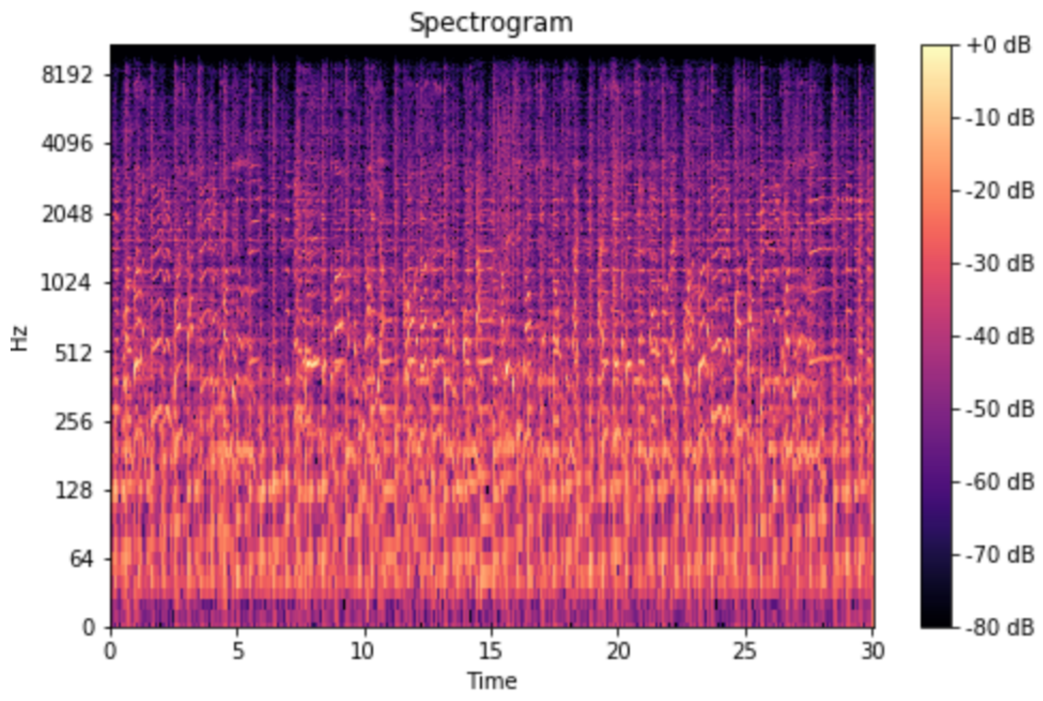

Figure 2.3 - Mel Spectrogram [15]

2.1.4. Transform-based representations

Transform-based representations [16] involve applying mathematical transforms to the audio signal to extract specific features. The Fourier transform, such as the Short-Time Fourier Transform (STFT), provides a frequency-domain representation. Other transform-based representations include the wavelet transform, which captures both time and frequency information, and the constant-Q transform (CQT), which provides a logarithmic frequency resolution.

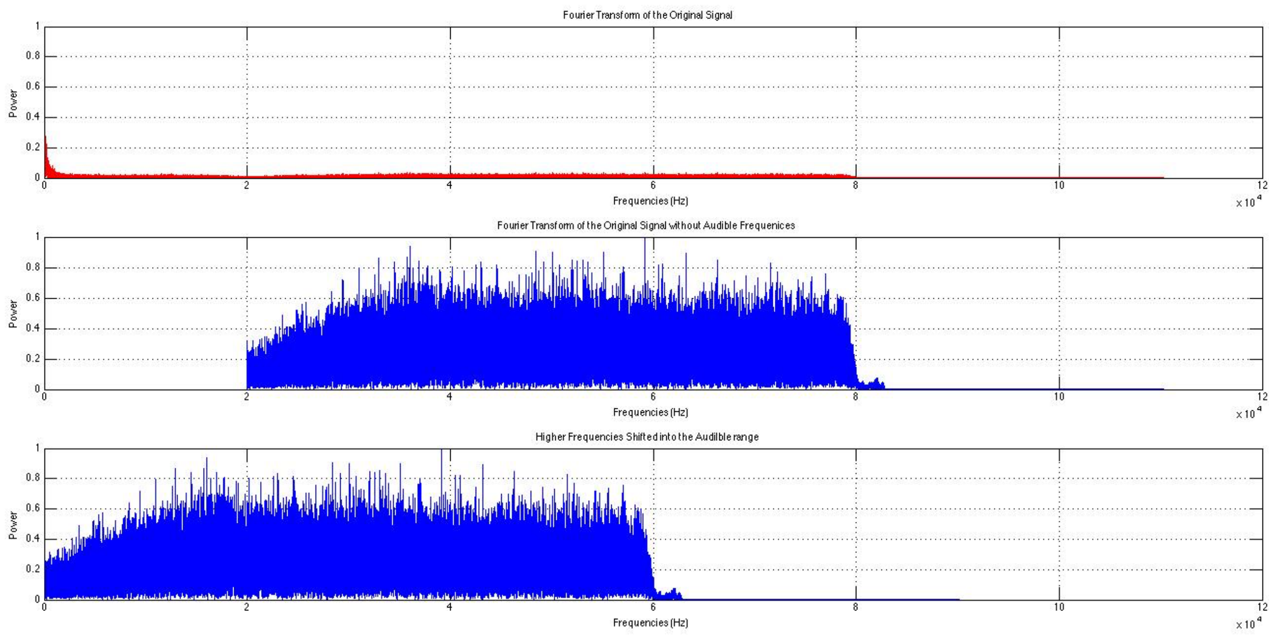

Figure 2.4 - Operations during Fourier Transformation [17]

2.1.5. Statistical representations

Statistical representations [18] involve characterizing the audio signal using statistical measures. For instance, statistical features like mean, variance, skewness, or kurtosis can be computed on the audio signal or its transformed representations to capture different aspects of its statistical properties.

2.1.6. Deep learning representations

With the advent of deep learning, representations learned by deep neural networks have gained prominence. Deep learning representations [16] can be derived from raw audio waveforms or transformed representations, such as spectrograms or MFCCs. Deep architectures, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), learn hierarchical representations from audio data, enabling automatic feature extraction and capturing complex patterns. Specifically, Wave2vec [19] and Trillson [20] are two state-of-the-art deep learning models for audio representation learning. Wave2vec is a self-supervised learning model that learns to represent speech signals in a way that is useful for automatic speech recognition (ASR). Trillson is a deep metric learning framework for music information retrieval (MIR) tasks. Both Wave2vec and Trillson can be used to learn powerful audio representations that can be used for a variety of downstream tasks.

2.1.7. Hybrid representations

Hybrid representations [16] combine multiple types of audio representations to capture diverse aspects of the audio signal. For example, a hybrid representation might include a combination of spectrograms and MFCCs to capture both the spectral content and perceptual characteristics of the audio signal.

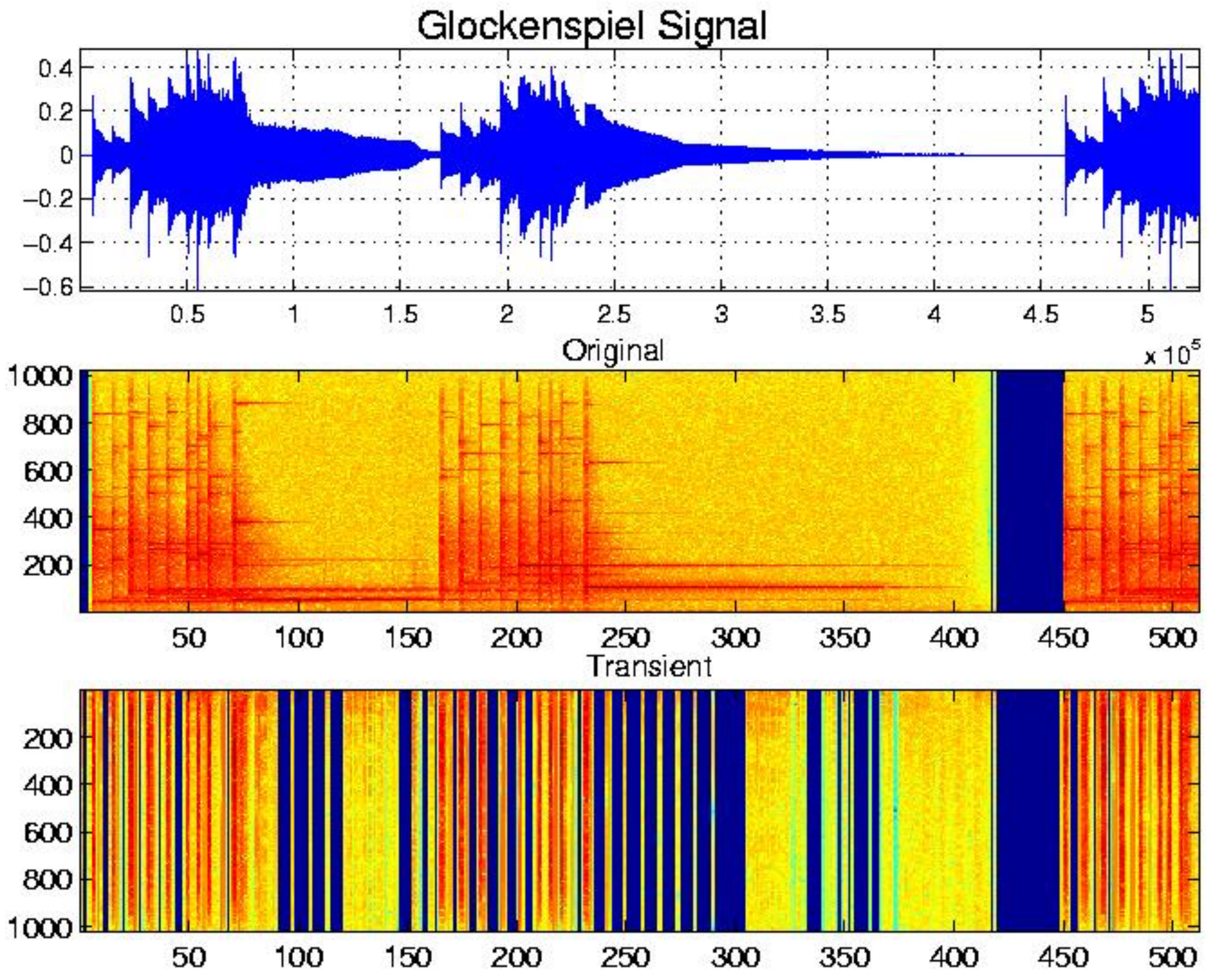

Figure 2.5 - Audio representation using time and frequency transformations. [21]

2.2. Deep Learning

2.2.1. Introduction

Deep learning is a subset of machine learning that has revolutionised various fields by enabling the development of highly complex and sophisticated models. It involves training neural networks with multiple layers to learn hierarchical representations from raw data. By leveraging the power of deep neural networks, deep learning models can automatically extract intricate patterns and features from data, allowing them to capture nuanced relationships and make accurate predictions. Deep learning algorithms, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have demonstrated exceptional performance in tasks such as image and speech recognition, natural language processing, and recommender systems. These models have pushed the boundaries of what is possible in these domains, achieving human-level or even superhuman performance in some cases.
Deep learning has benefited from significant advancements in hardware and the availability of large-scale datasets. The development of powerful Graphics Processing Units (GPUs) and specialised hardware accelerators, along with the rise of distributed computing, has enabled the training of deeper and more complex models. Additionally, the proliferation of diverse and abundant data sources has facilitated the training of deep learning models on massive datasets. This combination of computational resources and data availability has played a crucial role in unlocking the potential of deep learning and driving its widespread adoption across industries.
The impact of deep learning spans across various domains. In healthcare, deep learning models have been employed for medical imaging analysis, disease diagnosis, and drug discovery. In autonomous driving, deep learning algorithms have revolutionised perception systems and enabled significant progress in autonomous navigation. Deep learning has also revolutionised natural language processing, leading to advancements in machine translation, sentiment analysis, and voice recognition. In finance, deep learning models have been used for fraud detection, algorithmic trading, and risk assessment. These are just a few examples of the extensive applications of deep learning, highlighting its transformative potential and the continuous exploration and improvement of deep learning techniques to address increasingly complex problems.

2.2.2. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [22] have revolutionised the field of deep learning by enabling effective analysis of visual and sequential data. CNNs are particularly well-suited for tasks such as image classification, object detection, and natural language processing. Their ability to automatically learn hierarchical representations from raw data, capturing local and global dependencies, has made them indispensable in various applications. By leveraging convolutional layers, pooling layers, and non-linear activations, CNNs excel at extracting and recognizing meaningful patterns, making them a powerful tool for audio, image, and video analysis.

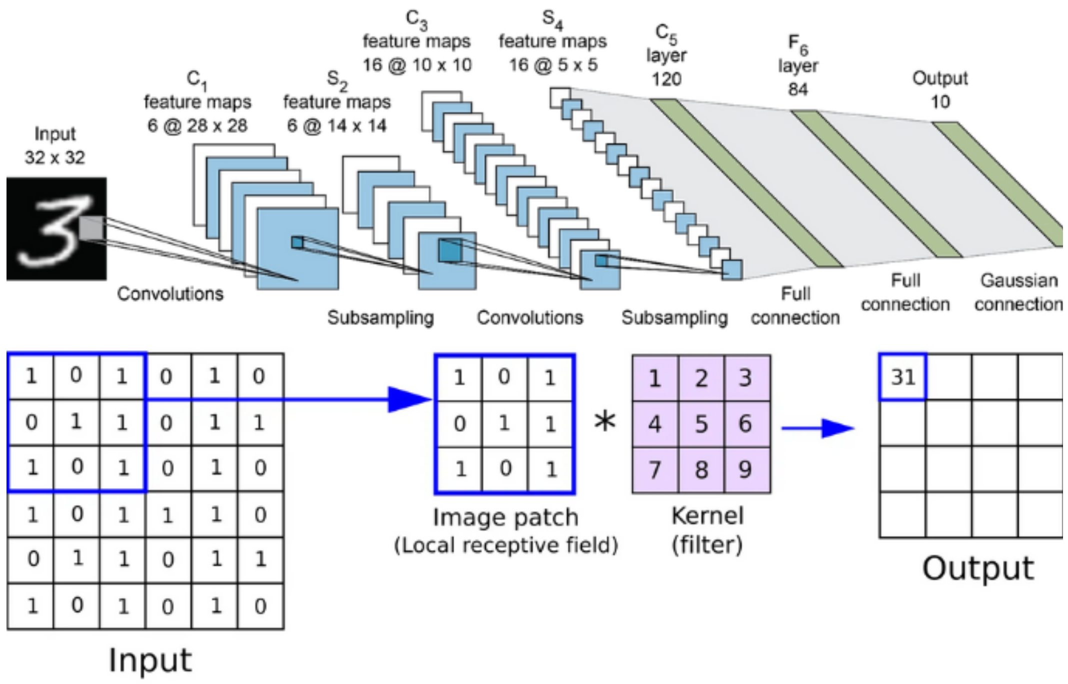

Figure 2.6 - Convolutional Neural Network Architecture

2.2.3. Common Architectures

2.2.3.1. RESNET

ResNet (Residual Neural Network) [23] models have significantly advanced the training of deep neural networks. By introducing skip connections that allow the flow of information directly from earlier layers to subsequent layers, ResNet models enable the training of much deeper architectures. This addresses the vanishing gradient problem and facilitates the optimisation of extremely deep networks with hundreds of layers. ResNet models have achieved remarkable success in image classification, object detection, and other visual tasks, surpassing the performance of shallower networks. Their ability to effectively learn hierarchical representations and handle complex data distributions has made ResNet models a crucial component of deep learning applications.

2.2.3.2. LENET

LeNet-5 [24] is one of the pioneering CNN architectures, introduced by Yann LeCun in 1998. It consists of multiple convolutional layers followed by fully connected layers. LeNet-5 was specifically designed for handwritten digit recognition and played a crucial role in establishing the effectiveness of CNNs in image classification tasks.

2.2.3.3. ALEXNET

AlexNet [25], proposed by Alex Krizhevsky et al. in 2012, gained significant attention for its breakthrough performance in the ImageNet Large-Scale Visual Recognition Challenge. It comprises multiple convolutional layers, pooling layers, and fully connected layers. AlexNet introduced the use of rectified linear units (ReLU) as activation functions and demonstrated the effectiveness of deep CNNs in image classification tasks.

2.2.3.4. VGGNET

VGGNet [26], developed by the Visual Geometry Group (VGG) at the University of Oxford in 2014, is known for its simplicity and depth. It consists of a series of stacked convolutional layers with small 3x3 filters and pooling layers. VGGNet explores the impact of increasing network depth and demonstrates that deeper networks can improve performance on various image recognition tasks.

2.2.3.5. GOOGLENET

GoogLeNet [27], introduced by Szegedy et al. from Google Research in 2014, introduced the concept of inception modules. These modules consist of parallel convolutional layers with different filter sizes, allowing the network to capture information at multiple scales. GoogLeNet addresses the challenge of computational efficiency and demonstrates the effectiveness of "network-in-network" architectures.

2.2.3.6. DENSENET

DenseNet [28], presented by Huang et al. in 2016, rethinks the connectivity pattern in CNNs. It employs dense blocks, where each layer is directly connected to all subsequent layers within the block. This dense connectivity facilitates feature reuse, reduces the number of parameters, and enhances gradient flow throughout the network. DenseNet exhibits strong performance and parameter efficiency.

2.2.3.7. MOBILENET

MobileNet [29], introduced by Howard et al. in 2017, focuses on efficiency for deployment on mobile and embedded devices. It utilises depth-wise separable convolutions, which split the standard convolution into separate depth-wise and point-wise convolutions. This technique significantly reduces the computational complexity while maintaining good accuracy, making it suitable for resource-constrained environments.

2.2.3.8. EFFICIENTNET

EfficientNet [30], proposed by Tan et al. in 2019, addresses the challenge of scaling CNN models effectively. It employs a compound scaling method that balances network depth, width, and resolution. By systematically scaling these dimensions, EfficientNet achieves state-of-the-art performance while maintaining computational efficiency.

2.2.4. Transformers

Transformer [31] models have emerged as a groundbreaking architecture in natural language processing and have found success in other domains as well. Unlike traditional recurrent neural networks (RNNs), Transformers employ a self-attention mechanism to capture dependencies between elements in a sequence simultaneously, allowing for efficient parallelisation and handling of long-term dependencies. This attention-based approach enables Transformers to capture contextual relationships and achieve state-of-the-art performance in tasks such as machine translation, text generation, and sentiment analysis. Their ability to model global interactions has also led to their adoption in other domains, including audio and image processing.

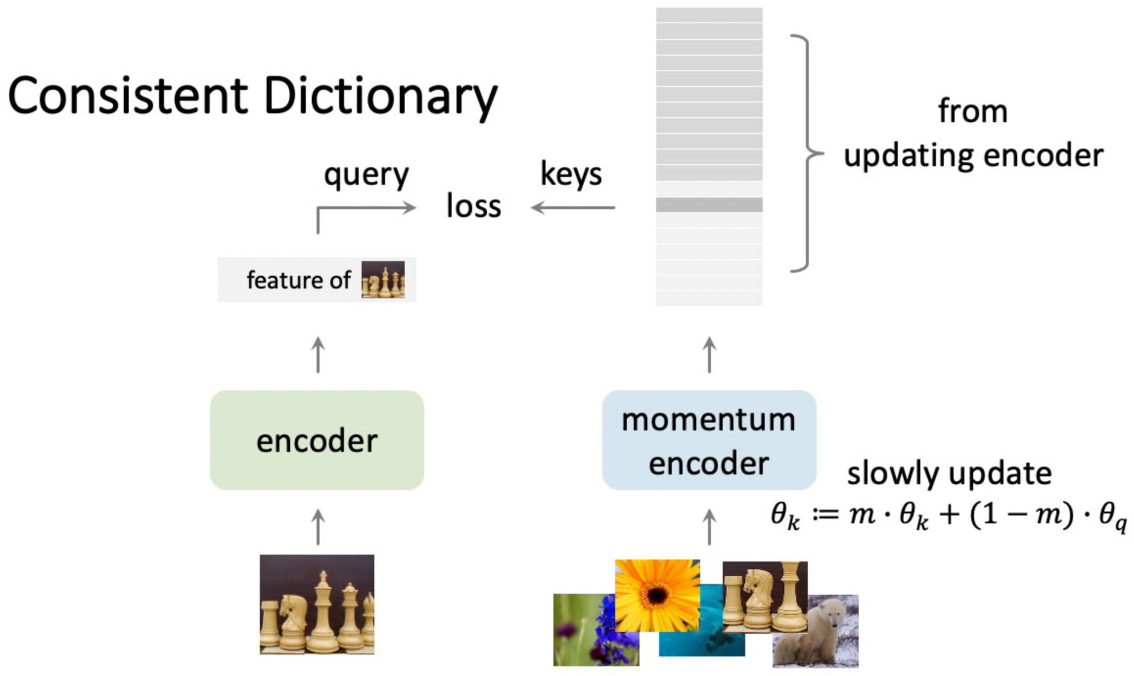

Figure 2.7 - Transformers Architecture

2.2.5. Contrastive Learning

Contrastive learning is a prominent paradigm in the field of machine learning, particularly within the domain of computer vision, that has garnered significant attention and made substantial contributions to the advancement of representation learning. This approach has proven to be instrumental in various applications, such as image recognition, object detection, and natural language processing. The fundamental principle underlying contrastive learning revolves around the notion of enhancing the discriminative power of feature representations by leveraging the relationships between positive and negative pairs of data points. Through this process, contrastive learning seeks to imbue representations with semantic information that facilitates downstream tasks.

One of the seminal frameworks in the realm of contrastive learning is the SimCLR (SimCLR: A Simple Framework for Contrastive Learning of Visual Representations) [5], which has been instrumental in elucidating the principles and techniques involved in this paradigm. SimCLR,, introduces a novel perspective on contrastive learning by emphasizing the importance of data augmentation and the construction of positive and negative pairs. It employs a siamese network architecture to learn feature representations that maximize similarity among positive pairs and minimize it among negative pairs. SimCLR's architecture encapsulates the idea of self-supervised learning, where positive pairs are derived from augmentations of the same image, fostering a rich understanding of the underlying data distribution. This self-supervised paradigm has gained considerable traction, largely due to its ability to capitalize on large-scale unlabeled datasets and its compatibility with transfer learning tasks.

Another noteworthy contrastive learning framework is the Momentum Contrast (MoCo) [6]. MoCo innovatively addresses the issue of constructing negative pairs by introducing a dynamic dictionary queue, which circumvents the need for maintaining a fixed set of negative samples. This architecture relies on a momentum encoder and a query encoder, where the momentum encoder lags behind the query encoder, thus promoting the accumulation of informative features over time. The MoCo framework not only enhances the efficiency of contrastive learning but also showcases the significance of a momentum update mechanism in the context of training deep neural networks. It has demonstrated its efficacy in various computer vision benchmarks, underscoring its capacity to yield state-of-the-art performance in a multitude of tasks.

In summary, contrastive learning is an influential paradigm within the field of machine learning, with SimCLR and MoCo standing as prominent exemplars of its application. These frameworks have significantly advanced our understanding of how to learn powerful feature representations through the judicious construction of positive and negative pairs and the employment of data augmentation techniques. Their contributions extend beyond the realm of computer vision, with implications for a wide array of domains, making them pivotal in contemporary research efforts aimed at harnessing the full potential of contrastive learning for enhancing the performance of machine learning models.

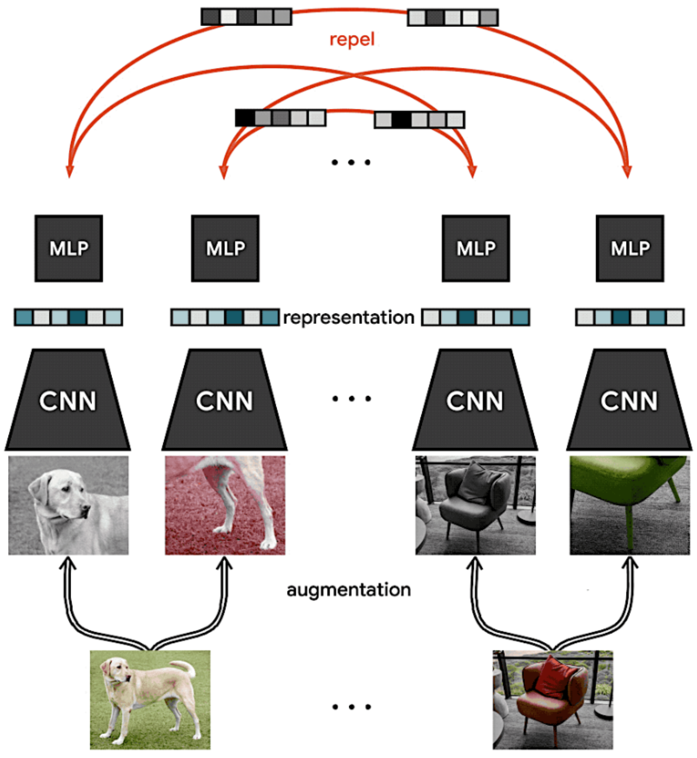

Figure 2.8 - Contrastive Learning

2.3. Distances

Distances play a pivotal role in deep learning and hold particular significance in music information retrieval (MIR) when employing deep metric learning on audio representations. In the context of MIR, audio data is often transformed into high-dimensional feature spaces where the choice of distance metric directly impacts the model's ability to capture meaningful similarities and differences between audio samples. Deep metric learning leverages these distances to learn embeddings that optimize the similarity between similar audio instances while maximizing the dissimilarity between dissimilar ones. This is crucial for tasks like music recommendation and similarity-based search, where understanding nuanced audio relationships is paramount. Additionally, distances enable the model to generalize effectively, making it adaptable to diverse audio content. Therefore, careful consideration and customization of distance metrics in deep metric learning are essential for enhancing the performance and relevance of MIR systems, aligning them more closely with human perception and preferences.

2.3.1. Introduction

Distances play a pivotal role in machine learning and deep learning, serving as fundamental measures for quantifying the similarity or dissimilarity between data points. These metrics enable the assessment of proximity or separation of samples within the feature space, crucial for various tasks in these domains. In the context of machine and deep learning, a range of distance metrics are utilised, each with its own characteristics and applicability.

2.3.2. Euclidean Distance

The Euclidean distance, one of the most widely utilised metrics, measures the straight-line distance between two points in a multidimensional space. By computing the square root of the sum of squared differences between corresponding coordinates, it provides a reliable measure of dissimilarity. Euclidean distance finds extensive application in tasks such as clustering, classification, and dimensionality reduction.

$$d = \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2}$$

2.3.3. Manhattan Distance (Cityblock Distance)

The Manhattan distance, also known as the Cityblock distance or L1 distance, evaluates dissimilarity by summing the absolute differences between corresponding coordinates. This distance metric proves particularly valuable when dealing with data that follows a grid-like structure or when movement is constrained to vertical and horizontal paths. Manhattan distance is applied in diverse domains, including image recognition, time series analysis, and recommendation systems.

$$d = |x_2 - x_1| + |y_2 - y_1|$$

2.3.4. Chebyshev Distance

The Chebyshev distance, often referred to as maximum norm or L∞ norm, determines dissimilarity by considering the maximum absolute difference between corresponding coordinates. This metric captures the maximum shift required along any dimension to align two points. Chebyshev distance finds applications in image processing, anomaly detection, and clustering algorithms.

$$d = max(|x_2 - x_1|, |y_2 - y_1|)$$

2.3.5. Minkowski Distance

The Minkowski distance represents a generalised distance metric that encompasses both Euclidean and Manhattan distances as special cases. It introduces a parameter, p, allowing the adjustment of distance calculation based on specific requirements. When p=2, Minkowski distance is equivalent to the Euclidean distance, while p=1 corresponds to the Manhattan distance. Minkowski distance offers flexibility and is used in various domains, including pattern recognition, feature selection, and clustering.

$$d = \left(\sum_{i=1}^{n} |x_2^i - x_1^i|^{\rho} \right)^{1/\rho}$$

2.3.6. Hamming Distance

Hamming distance specialises in comparing binary data, evaluating dissimilarity by counting the number of positions at which two binary strings differ. This metric is extensively employed in error detection and correction, DNA sequence analysis, and data clustering tasks.

$$d = \sum_{i=1}^{n} |x_2^i - x_1^i|$$

2.3.7. Cosine Similarity

Cosine similarity quantifies the similarity between two vectors in a high-dimensional space. By calculating the cosine of the angle between the vectors, it captures the cosine of their similarity. Cosine similarity is widely used in natural language processing, recommendation systems, and information retrieval.

$$\cos(\theta) = \frac{x \cdot y}{||x|| \, ||y||}$$

2.3.8. Correlation Distance

The correlation distance measures dissimilarity by considering the correlation coefficient between two vectors. It quantifies the linear relationship between variables and determines dissimilarity based on the lack of correlation. This distance metric finds application in various domains, including data analysis, image processing, and feature selection.

$$d = 1 - \frac{(x \cdot y)}{\|x\| \|y\|}$$

2.3.9. Seuclidean Distance

The Seuclidean distance, also known as standardized Euclidean distance, adjusts the Euclidean distance by scaling each dimension based on its standard deviation. It accounts for the different scales of features and emphasizes the impact of features with higher variability. Seuclidean distance is valuable in scenarios where feature scaling plays a significant role, such as in bioinformatics, medical imaging, and sensor data analysis.

$$d = \sqrt{\sum_{i=1}^{n} (x_2^i - x_1^i)^2}$$

2.3.10. Kulsinski Distance

The Kulsinski distance is a statistical distance metric that evaluates dissimilarity between two binary vectors. It takes into account the number of matching elements and non-matching elements, considering the distribution of ones and zeros in the vectors. This distance is commonly employed in clustering, data mining, and pattern recognition tasks.

$$d = \frac{1}{n} \sum_{i=1}^{n} \frac{|x_i - y_i|}{x_i + y_i}$$

In machine and deep learning, the selection of an appropriate distance metric depends on the specific characteristics of the data and the objectives of the task at hand. These diverse distance measures, including Euclidean, Manhattan, Chebyshev, Minkowski, Hamming, Cosine, Cityblock, Seuclidean, Correlation, and Kulsinski, offer versatile tools for assessing similarity and dissimilarity between data points, enabling accurate and effective analyses in various applications.

2.4. Audio Augmentation

2.4.1. Introduction

Audio augmentations play a crucial role in enhancing the performance and generalization capabilities of deep learning models for audio-related tasks. These techniques involve applying various transformations and modifications to the audio data during the training process. By introducing controlled variations, such as additive noise, random masking, time stretching, pitch shifting, and other augmentations, the models can learn to capture robust and invariant representations that are more resilient to real-world challenges. Augmentations help address data limitations, alleviate overfitting, and improve the model's ability to handle variations in audio content, duration, pitch, background noise, and other acoustic factors. By incorporating audio augmentations into the training pipeline, deep learning models can learn more effectively from diverse and realistic audio datasets, resulting in improved performance and greater adaptability in real-world audio applications.

2.4.2. Additive Noise

Additive noise augmentation introduces random background noise to the audio signal. It involves superimposing a low-level noise signal onto the original audio waveform, emulating environmental or recording conditions. By adding controlled levels of noise, the model can learn to be robust against varying noise levels and improve its ability to extract relevant features in noisy audio environments.

2.4.3. Random Masking

Random masking augmentation randomly masks segments of the audio, either in the time or frequency domain. This technique involves selectively zeroing out specific temporal or spectral regions of the audio signal. By masking certain parts of the audio, the model is encouraged to focus on other unmasked segments, forcing it to learn from different parts of the audio spectrogram and enhancing its ability to handle partial or missing information.

2.4.4. Time Stretching

Time stretching augmentation alters the playback speed of the audio while preserving its pitch. It can compress or expand the temporal duration of the audio, thereby introducing variations in the rhythm and tempo. This augmentation helps the model learn to recognize and handle different audio durations, making it more resilient to speed variations in real-world audio recordings.

2.4.5. Pitch Shifting

Pitch shifting augmentation modifies the pitch of the audio while maintaining its duration. It involves changing the fundamental frequency of the audio, introducing variations in the tonal characteristics. This technique enables the model to learn representations that are invariant to pitch variations and improves its ability to generalize across different musical tones and voices.

2.4.6. Time and Frequency Masking

Time and frequency masking augmentation involves selectively masking temporal or spectral regions of the audio spectrogram. Temporal masking zeros out specific time segments, while frequency masking masks certain frequency bins. By removing segments in the time or frequency domain, this augmentation encourages the model to focus on different temporal or spectral components, promoting robustness to missing or noisy segments.

2.4.7. SpecAugment

Inspired by computer vision, SpecAugment randomly masks segments of the audio spectrogram. It introduces horizontal and vertical masks that cover consecutive time steps or frequency bins, encouraging the model to learn from various parts of the spectrogram. This technique helps the model generalize to variations in acoustic features and improves its robustness against small perturbations in the input spectrogram.

2.4.8. Resampling

Resampling augmentation involves changing the sample rate of the audio signal. It can upsample or downsample the audio, altering its frequency content and temporal resolution. Resampling introduces variations in the audio quality and can help the model adapt to different sample rates encountered in real-world scenarios.

3. Methodology

3.1. Problem Definition and Solution Approach

This thesis addresses the critical challenge of developing robust and discriminative audio representations for a range of applications, including audio classification, music retrieval, and speech recognition. In an era marked by the exponential growth of digital audio content, the need for effective methods to navigate and harness this wealth of information is paramount. Deep Metric Learning (DML) emerges as a promising avenue to meet this challenge by harnessing the capabilities of machine learning models to decipher intricate relationships within audio data. The primary objective of this research is to delve into the realm of DML and its application in creating meaningful audio representations. To achieve this goal, the study will explore a variety of deep neural network architectures and loss functions, including the widely-used triplet loss and contrastive loss. These architectural and loss function variations will be rigorously evaluated, offering insights into their effectiveness in producing audio embeddings capable of capturing the intrinsic characteristics of the data.
One of the central tenets of this research involves the selection and assessment of appropriate deep neural network architectures. Convolutional Neural Networks (CNNs) are renowned for their ability to capture both local and global dependencies within audio signals. By experimenting with various architectural configurations, this study seeks to identify the most suitable architecture that optimally encodes audio data for downstream tasks. This investigation is pivotal to understanding the nuances of different neural network structures and their role in achieving effective audio representations.
In tandem with architectural exploration, this thesis will thoroughly investigate the choice of loss functions, a critical component in DML for audio. Triplet loss and contrastive loss, among others, will be evaluated to discern their impact on the quality of learned audio embeddings. The objective is to uncover which loss function(s) yield embeddings that most accurately reflect the semantic relationships among audio samples. Subsequently, these learned representations will be subjected to rigorous testing using various distance metrics and normalization techniques. The effectiveness of these embeddings, as determined by distance-based evaluation using established benchmark datasets, will serve as a pivotal measure of their utility for diverse audio-related tasks. Ultimately, this research aims to provide invaluable insights into the design and optimization of audio representations, with a keen focus on their relevance to music information retrieval and content-based similarity tasks.

3.2. Loss Functions

3.2.1. Triplet Loss

Triplet Loss [32] has emerged as a prominent technique in the field of Deep Metric Learning (DML), offering a powerful means of learning effective representations for similarity-based tasks. With the increasing availability of large-scale music collections, the need for robust models capable of capturing intricate relationships between songs has become imperative. In this context, Triplet Loss serves as a cornerstone for training machine learning models that can accurately measure the similarity between musical inputs.

Triplet Loss operates on the principle of learning embeddings, which map input instances into a high-dimensional space, where distances reflect their inherent similarity or dissimilarity. By employing triplets of instances—comprising an anchor, a positive, and a negative—the objective is to ensure that the anchor is closer to the positive instance compared to the negative one. The triplet formulation provides a fine-grained supervisory signal for the model to optimize the embedding space, aligning it with the desired similarity relationships.

The efficacy of Triplet Loss lies in its ability to facilitate discriminative learning, where embeddings of similar instances are brought closer together while pushing dissimilar instances apart. This process enables the model to capture subtle nuances in the audio features of songs, transcending conventional categorical labels and providing a more nuanced representation of similarity.

Moreover, Triplet Loss has shown notable success in addressing challenges associated with the curse of dimensionality. By leveraging the triplet structure, the model is encouraged to learn compact representations, where semantically similar instances are densely clustered, enhancing retrieval performance and computational efficiency.

In the realm of music, the application of Triplet Loss holds tremendous potential. By exploiting the rich audio content of songs, DML models trained with Triplet Loss can offer personalized music recommendations, facilitate content-based music retrieval, and contribute to various music-related tasks such as playlist generation and music similarity analysis.

This thesis aims to delve into the realm of Deep Metric Learning for music retrieval, focusing specifically on the application of Triplet Loss. By investigating novel techniques and strategies for leveraging Triplet Loss in the context of music similarity, we strive to advance the state-of-the-art in music recommendation systems and enhance the overall user experience in navigating and discovering music in the digital age.

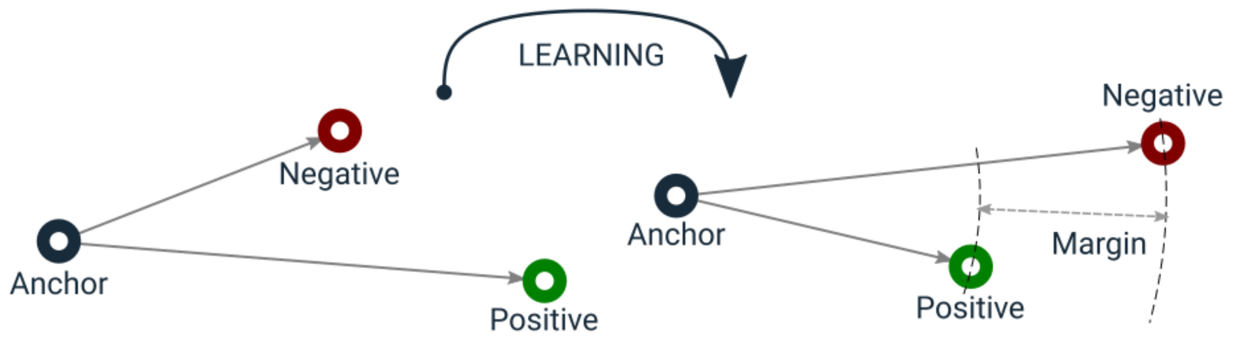

Figure 3.1 - Triplet Loss

3.2.2. Contrastive Loss

Contrastive Loss [33] has emerged as a key component in Deep Metric Learning (DML) algorithms, offering a powerful framework for training models that can effectively capture similarity relationships between instances. In the context of music retrieval, where the goal is to identify and recommend similar songs based on their audio content, Contrastive Loss provides a valuable mechanism to learn discriminative embeddings that preserve the inherent structure of the music.

The underlying principle of Contrastive Loss is to encourage similarity for pairs of instances belonging to the same class, while enforcing dissimilarity for pairs of instances from different classes. By formulating a loss function that maximizes the similarity between positive pairs and minimizes the similarity between negative pairs, the model is incentivized to learn embeddings that effectively separate distinct musical instances while bringing similar instances closer together.

One of the key advantages of Contrastive Loss lies in its ability to mitigate the challenges posed by high-dimensional spaces. By learning embeddings that are optimized for pairwise comparisons, Contrastive Loss helps to reduce the computational complexity of similarity-based tasks. Furthermore, the learned embeddings can capture subtle variations in audio features and preserve the semantic relationships between songs, facilitating accurate music retrieval and recommendation.

In the field of music, Contrastive Loss holds significant promise for a wide range of applications. By leveraging its ability to learn meaningful representations, models trained with Contrastive Loss can provide personalized music recommendations, enable content-based music retrieval, and support tasks such as music similarity analysis and genre classification.

The primary objective of this thesis is to explore and investigate the effectiveness of Contrastive Loss in the context of music retrieval and recommendation systems. By employing innovative strategies and techniques, we aim to enhance the performance and efficiency of existing approaches, ultimately contributing to the advancement of music recommendation technology and improving the overall user experience in discovering and enjoying music in the digital era.

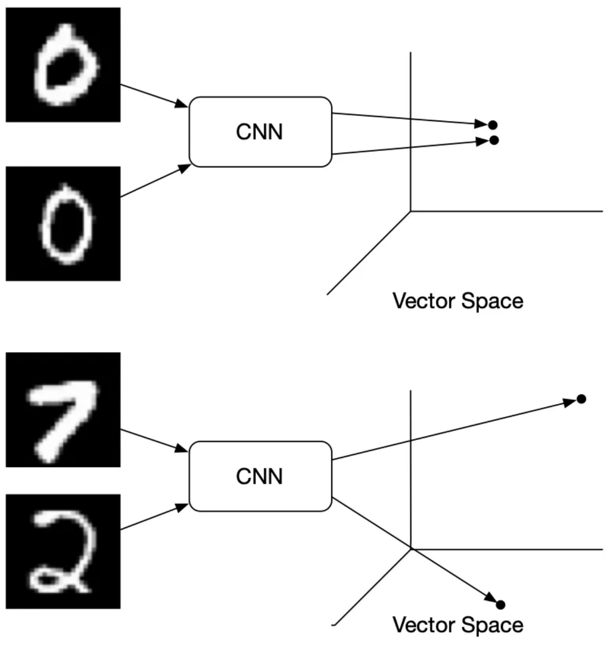

Figure 2.3 - Contrastive Learning

3.3. Data Preparation

Here, we will describe in detail the process of creating the training dataset for our experiment.Our initial dataset consists of 1139 songs in the WAV format. To extract meaningful representations from each song, we employ two libraries: pyaudio and deep-audio-features. These libraries provide us with the necessary tools to extract high-level features and representations from audio data.

To create our ground truth files, we construct quartets of songs. Each quartet comprises three different songs, and the fourth song within the quartet is one of the first three. However, in this case, the fourth song is intentionally chosen to be the least similar among the three. This quartet structure enables us to establish a clear contrast between similar and dissimilar songs within our training data.

For the implementation of triplet loss, we form triplets. Each triplet consists of representations from three songs: an anchor, a positive, and a negative. The anchor serves as the reference point, while the positive represents a song similar to the anchor. Conversely, the negative represents a song that is dissimilar to the anchor. By using these triplets, we train our model to learn a metric space where the distance between the anchor and positive songs is minimized, while the distance between the anchor and negative songs is maximized.

In addition to triplet loss, we also utilize contrastive loss during our training process. Each training batch comprises representations of N songs randomly selected from the entire database. This means that each batch includes a diverse set of songs, allowing the model to learn to distinguish between various instances and develop robust representations that capture the inherent similarities and differences among the songs.

By employing these approaches, we aim to train our machine learning models to effectively learn the underlying structure of the audio data, enabling them to identify and rank the most similar songs given an input.

3.4. Models

In this master thesis, various convolutional neural network (CNN) architectures, in combination with linear layers, were employed to tackle the task of music similarity and retrieval. Specifically, a total of five distinct CNN models were utilized, each incorporating a different number of CNN layers and linear layers.
Each of these five models featured Dropout regularization applied after the last CNN layer. Dropout is a technique used to prevent overfitting by randomly deactivating a certain proportion of neurons during training. By applying Dropout, the models were able to enhance their generalization capability and reduce the risk of overfitting.
One of the models implemented in this study was a CNN architecture comprising six CNN layers. Each CNN layer consisted of a convolutional layer, a Batch Normalization layer (BatchNorm2d), a LeakyReLU activation function, and a MaxPooling layer (MaxPool2d). This architecture was designed to capture intricate audio features through multiple levels of convolutional and pooling operations.
In addition to the CNN layers, this particular model was equipped with three linear layers. Each linear layer included a Dropout layer, a linear transformation (Linear), and a LeakyReLU activation function, except for the last linear layer, which solely utilized a linear transformation and a LeakyReLU activation. The incorporation of linear layers allowed the model to learn complex relationships and patterns within the learned features.
Furthermore, two ResNet models, namely ResNet-18 and ResNet-50, were employed in this study. These models are renowned for their deep architectures and residual connections, which enable effective feature extraction and learning. In order to accommodate the specific requirements of the music dataset, the first convolutional layer of both ResNet models was modified. The modified layer was defined as 'nn.Conv2d(1, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)', facilitating the handling of the input music data.
The ResNet-18 and ResNet-50 models were employed to train models using both Contrastive Loss and Triplet Loss. Contrastive Loss, a technique used in Deep Metric Learning (DML), encourages similar instances to be closer together and dissimilar instances to be farther apart. Triplet Loss, another DML technique, focuses on optimizing the relative distances between triplets of instances, ensuring that positive instances are closer to an anchor instance compared to negative instances. By utilizing these loss functions, the ResNet models aimed to learn discriminative embeddings that effectively capture the similarity relationships between songs.
In contrast, the other models in this study were solely trained using Triplet Loss, emphasizing the exploration of this loss function's efficacy in the context of music retrieval and similarity analysis.
The utilization of various CNN architectures, including the aforementioned models with different layer configurations, alongside the application of Contrastive Loss and Triplet Loss, showcases a comprehensive approach to address the challenges of music similarity and retrieval. Through this research, we seek to advance the field by exploring novel combinations of network architectures and loss functions, thereby enhancing the performance and accuracy of music recommendation

systems and enabling users to discover and engage with music in a more personalized and satisfying manner.

3.4.1. Architectures:

3.4.1.1 TRIPLET LOSS

For training with Triplet Loss, the following models were utilized:

Model1

Layer no.	**Type**	**Output Channels**	**Kernel Size**	**Stride**	**Activation**
conv1	Conv2d	32	1x1	1	ReLU
conv2	Conv2d	64	1x1	1	ReLU
max_pool1	Max_Pool2d	-	1	1	-
dropout1	Dropout2d	-	-	-	-
fc1	Linear	128	-	-	-
Number of parameters: 7.4M					

Model2

Layer no.	**Type**	**Output Channels**	**Kernel Size**	**Stride**	**Activation**
conv1	Conv2d	32	1x1	1	ReLU
conv2	Conv2d	64	1x1	1	ReLU
conv3	Conv2d	128	1x1	1	ReLU
max_pool1	Max_Pool2d	-	1	1	-
dropout1	Dropout2d	-	-	-	-
fc1	Linear	512	-	-	-
fc2	Linear	256	-	-	-
fc3	Linear	128	-	-	-
Number of parameters: 59.5M					

Model3

Layer no.	Type	Output Channels	Kernel Size	Stride	Activation
conv1	Conv2d	32	1x1	1	ReLU
conv2	Conv2d	64	1x1	1	ReLU
conv3	Conv2d	128	1x1	1	ReLU
conv4	Conv2d	128	1x1	1	ReLU
max_pool1	Max_Pool2d	-	1	1	-
dropout1	Dropout2d	-	-	-	-
fc1	Linear	512	-	-	-
fc2	Linear	256	-	-	-
fc3	Linear	128	-	-	-
Number of parameters: 59.6M					

Model4

Layer no.	Type	Output Channels	Kernel Size	Stride	Activation
conv1	Conv2d	32	1x1	1	ReLU
conv2	Conv2d	64	1x1	1	ReLU
conv3	Conv2d	128	1x1	1	ReLU
conv4	Conv2d	128	1x1	1	ReLU
max_pool1	Max_Pool2d	-	1	1	-
dropout1	Dropoutd2d	-	-	-	-
fc1	Linear	1024	-	-	-
fc2	Linear	512	-	-	-
fc3	Linear	256	-	-	-
Fc4	Linear	128	-	-	-
Number of parameters: 119.5M					

Model5

Layer no.	Type	Output Channels	Kernel Size	Stride	Activation
conv1	Conv2d	32	1x1	1	ReLU
conv2	Conv2d	64	1x1	1	ReLU
conv3	Conv2d	128	1x1	1	ReLU
conv4	Conv2d	128	1x1	1	ReLU
conv5	Conv2d	64	1x1	1	ReLU
max_pool1	Max_Pool2d	-	1	1	-
dropout1	Dropoutd2d	-	-	-	-
fc1	Linear	2048	-	-	-
fc2	Linear	1024	-	-	-
fc3	Linear	512	-	-	-
fc4	Linear	256	-	-	-
fc5	Linear	128	-	-	-
Number of parameters: 121.6M					

Model6

Layer no.	Type	Output Channels	Kernel Size	Stride	Activation
conv_layer1: Sequential					
conv2d	Conv2d	32	1x1	1	LeakyReLU
batchnorm2d	BatchNorm2d	-	-	-	-
maxpool2d	MaxPool2d	-	1	1	-
conv_layer2: Sequential					
conv2d	Conv2d	64	1x1	1	LeakyReLU
batchnorm2d	BatchNorm2d	-	-	-	-
maxpool2d	MaxPool2d	-	1	1	-
conv_layer3: Sequential					
conv2d	Conv2d	128	1x1	1	LeakyReLU
batchnorm2d	BatchNorm2d	-	-	-	-

maxpool2d	MaxPool2d	-	1	1	-
conv_layer4: Sequential					
conv2d	Conv2d	64	1x1	1	LeakyReLU
batchnorm2d	BatchNorm2d	-	-	-	-
maxpool2d	MaxPool2d	-	1	1	-
linear_layer1: Sequential					
dropout	Dropout2d	-	-	-	-
linear	Linear	1024	-	-	LeakyReLU
linear_layer2: Sequential					
dropout	Dropout2d	-	-	-	-
Linear	Linear	256	-	-	LeakyReLU
linear_layer3: Sequential					
linear	Linear	126	-	-	LeakyReLU
Number of parameters: 59.7M					

Model7

ResNet18
Number of parameters: 11.7M

Model8

ResNet50
Number of parameters: 25.6M

In the implementation of ResNet-18 and ResNet-50, the default architectural configuration was maintained with the exception of the initial convolutional layer. Specifically, the original first convolutional layer was replaced with a customized layer defined as:

Layer no.	**Type**	**Output Channels**	**Kernel Size**	**Stride**	**Padding**
conv2d	Conv2d	64	7x7	2x2	3x3

3.4.1.2. CONTRASTIVE LOSS

For training with Contrastive Loss, the following models were utilized:

Model1

ResNet50
Number of parameters: 11.7M

Model2
ResNet50
Number of parameters: 25.6M

In the implementation of ResNet-18 and ResNet-50, the default architectural configuration was maintained with the exception of the initial convolutional layer. Specifically, the original first convolutional layer was replaced with a customized layer defined as:

Layer no.	Type	Output Channels	Kernel Size	Stride	Padding
conv2d	Conv2d	64	7x7	2x2	3x3

3.4.2. Training

The training process for this experiment involved training machine learning models using triplet loss and contrastive loss techniques. The models were optimized using the Adam optimizer, which is a widely used algorithm in deep learning. The Adam optimizer combines techniques such as momentum and adaptive learning rates to efficiently update the model's parameters during training. A learning rate of 1e^-5 was chosen for the training process. The learning rate determines the step size taken during parameter updates and can significantly impact the convergence and performance of the model. By carefully selecting a suitable learning rate and leveraging the benefits of the Adam optimizer, the training process aimed to find an effective balance between rapid convergence and accurate learning of the audio representations.

3.4.2.1. TRIPLET LOSS

This section outlines the training methodology for models utilizing the Triplet Loss function, a fundamental component in various applications of similarity learning. Triplet loss facilitates the learning of semantically meaningful representations by constraining the distances between anchor and positive samples while maximizing those between anchor and negative samples.

1. Batch Splitting
In the initial phase, each batch of training data is meticulously partitioned into three constituent subsets: anchor, positive, and negative samples. These subsets are pivotal for the subsequent steps in the training procedure.
Anchor Sample: The anchor sample is considered the point of reference for comparative analysis throughout the training process. It serves as the baseline against which similarity and dissimilarity are assessed.
Positive Sample: The positive sample represents an audio data point that is closely related or similar to the anchor sample. This similarity aids in reinforcing the desired representations.
Negative Sample: In contrast, the negative sample is deliberately chosen to be dissimilar or distinct from the anchor sample. This disparity is instrumental in guiding the model towards learning discriminative representations.

2. Model Inference
Following batch splitting, the three sample subsets - anchor, positive, and negative - are propagated through the trained model to derive their respective output representations. This step harnesses the model's ability to transform input data into meaningful feature representations.

3. Triplet Loss Calculation
The Triplet Loss function is the cornerstone of this training procedure. It takes as input the output representations of the anchor, positive, and negative samples. These representations are fed into the loss function in a predefined order: anchor, positive, negative.
The Triplet Loss function is designed to compute a scalar loss value based on the distances or similarities between the anchor and positive representations, in conjunction with the anchor and negative representations. The overarching objective is to minimize the distance between anchor and positive representations while concurrently maximizing the distance between anchor and negative representations. This dual constraint fosters the development of highly discriminative embeddings.

4. Backpropagation and Parameter Update
To enable learning, gradients of the Triplet Loss with respect to the model's parameters are meticulously computed. Subsequently, the backpropagation algorithm is employed to efficiently propagate these gradients through the neural network. The model's parameters are iteratively updated to minimize the Triplet Loss.
This parameter update is achieved through an optimization algorithm, such as the widely-used Adam optimizer, which leverages the calculated gradients and the designated learning rate to fine-tune the model parameters. This iterative optimization process progressively refines the model's ability to generate representations that conform to the desired similarity constraints.

5. Iteration and Continuation
The entire training procedure is repeated for each batch of training data in an iterative fashion. This cyclic process entails the successive passage of batches through the model, computation of the Triplet Loss, and the consequential parameter updates. Training continues until convergence or until a predefined stopping criterion is met.
The training procedure utilizing the Triplet Loss function is a robust and versatile method for cultivating semantically meaningful representations in various applications. It is characterized by the careful selection and manipulation of anchor, positive, and negative samples, which, when coupled with gradient-based optimization, empowers models to learn highly discriminative embeddings.

By following this training loop, the model gradually learns to generate discriminative audio representations that effectively capture the similarity relationships between the anchor and positive samples. The triplet loss drives the model to minimize the distances between similar pairs and maximize the distances between dissimilar pairs, facilitating the development of embeddings suitable for audio similarity and retrieval tasks.

3.4.2.2. CONTRASTIVE LOSS

This section elucidates the training methodology for models employing the Contrastive Loss within the SimCLR framework. The objective is to foster the acquisition of robust audio representations through data augmentation, model inference, and the optimization of contrastive loss. This process yields embeddings that facilitate effective discrimination among audio samples.

1. Data Augmentation
The foundation of this training procedure lies in data augmentation. Each sample in the training dataset undergoes a twofold augmentation process, enhancing the dataset's diversity and enabling the model to generalize more effectively. Two specific augmentation techniques are applied:
Additive Noise: A controlled level of additive noise is injected into each sample, with a prescribed strength parameter of 0.5. This augments the samples by introducing controlled variations, enhancing the model's robustness.

Random Masking: A random masking strategy is employed with a probability of 0.2. This technique randomly masks portions of the audio, further diversifying the dataset.

2. Model Inference
The augmented versions of each sample are then provided as inputs to the trained model. The model, which has been crafted within the SimCLR framework, is adept at extracting intricate audio representations capable of capturing high-level features.
Outputs Generation: The model processes each augmented sample and generates corresponding output representations. These representations encapsulate the extracted audio characteristics, and their quality is improved through the use of the SimCLR architecture.

3. Contrastive Loss Calculation
The output representations obtained from the model serve as inputs to the contrastive loss function. This critical step assesses the similarity or dissimilarity between pairs of augmented samples.
Contrastive Loss Function: The contrastive loss function, an essential element of SimCLR, quantifies the relationships between pairs of representations. It encourages the model to minimize the distance between positive pairs (representations from the same sample) while simultaneously maximizing the distance between negative pairs (representations from different samples). This loss function facilitates the creation of embeddings that inherently encode the distinctiveness of audio samples.

4. Backpropagation and Parameter Update
The gradients of the contrastive loss concerning the model's parameters are meticulously computed. These gradients fuel the backpropagation process, allowing the efficient propagation of information through the network.
Parameter Updates: The model's parameters are updated through an optimization algorithm, typically employing the Adam optimizer. This optimization process aims to minimize the contrastive loss iteratively, enabling the model to learn discriminating audio representations.

5. Iteration and Continuation
The training procedure unfolds through an iterative process. Augmentation, model inference, contrastive loss calculation, and parameter updates are conducted repeatedly for multiple training iterations or epochs.
Epoch Progression: Each iteration advances the model's capacity to produce effective audio representations. The procedure continues until convergence is achieved, or predefined convergence criteria are met.

The training procedure for Contrastive Loss-based models, as orchestrated within the SimCLR framework, is a robust strategy for enhancing audio representations. It leverages data augmentation to diversify the training dataset and employs the contrastive loss function to enforce representations that effectively discriminate between audio samples. This iterative process culminates in the acquisition of embeddings capable of capturing the intricate nuances of audio data.

This iterative process allows the model to gradually learn discriminative audio representations that capture the underlying similarities and differences in the augmented samples.
By following this training loop with contrastive loss and utilizing the SimCLR framework, the model is trained to generate audio representations that effectively capture the similarity relationships between augmented samples. The combination of data augmentation and contrastive loss optimization enhances the model's ability to learn robust and discriminative representations, facilitating accurate audio retrieval and similarity ranking.

3.4.3. Evaluation

3.4.3.1. DISTANCES

In this master thesis, we comprehensively evaluated the trained models using a diverse range of distance metrics and scalers to assess their performance in various audio similarity scenarios.
The goal was to understand how the models trained with Triplet Loss and Contrastive Loss are performing across different combinations of distance metrics and scalers.
The distances utilized for evaluation are as follows:

- Chebyshev Distance
- Euclidean Distance
- Minkowski Distance
- Hamming Distance
- Cosine Similarity
- Cityblock/Manhattan Distance
- Standardized Euclidean Distance
- Correlation Distance
- Kulsinski Distance

Furthermore, each distance metric was combined with three different scalers to observe the models' behavior under various normalization conditions:

- MinMaxScaler
- StandardScaler
- Normalizer

To effectively evaluate the models, we conducted a systematic analysis, assessing their performance using each distance-scaler pair on the testing dataset comprising 3223 audio representations. By doing so, we could obtain a comprehensive understanding of how the models generalize to unseen data and how the choice of distance metric and scaler impacts their performance.

3.4.3.2. SCORE CALCULATION

To quantitatively evaluate the performance of the trained machine learning models, we developed a rigorous scoring methodology centered around the notion of audio similarity. Our evaluation process hinged on the utilization of carefully curated triplets of audio representations, each comprising two similar audio samples and one dissimilar sample. These triplets were designed with known ground truth, specifying which two audio representations should be considered as similar and which one as dissimilar.
For each triplet, we passed the audio representations through the trained models to obtain their embeddings. Subsequently, we employed various distance metrics and scalers. To measure the similarity between the embeddings. The pivotal criterion for scoring was as follows: if the distance between the two audio representations designated as similar was smaller than the distance between the dissimilar pair, the model received a correct classification score for that triplet. This process was systematically repeated for every triplet in our test dataset.
By summing the correct classifications and normalizing the count with respect to the total number of triplets, we calculated a percentage score, providing a comprehensive assessment of each model's capability to capture intricate audio similarity relationships. This scoring mechanism offered valuable insights into the models' proficiency in distinguishing between similar and dissimilar audio representations, facilitating a quantitative evaluation of their performance across diverse distance metrics and normalization techniques.

3.4.4. Implementation Details

The codebase employed in this thesis is part of the mir (Multimedia Information Retrieval) [34] repository, specifically within the similarity section. The mir repository is maintained by the Multimedia Analysis Group of the Computational Intelligence Lab (MagCIL) at the Institute of Informatics and Telecommunications, part of the National Center for Scientific Research “Demokritos."
The core functionality of the similarity section of the repository focuses on the extraction of audio representations from audio files and the execution of similarity queries. These queries are designed to identify the most similar audio files based on a given input audio file.
The exact codebase can be found here.

The proposed methodology was implemented in Python using the following libraries:
- Models were implemented and trained using PyTorch
- The SimCLR framework was utilised in training with Contrastive Learning
- Models trained on Apple M1 Pro, Total Number of GPU Cores: 14

4. Experiments

4.1. Dataset

The dataset used for this master thesis consists of a total of 4362 audio representations of songs. These audio representations have been split into two distinct sets for training and testing purposes. The training set comprises 1139 audio representations, while the testing set consists of 3223 audio representations.
The primary objective of the thesis is to train and evaluate models using two prominent loss functions, Triplet Loss and Contrastive Loss, with the aim of learning meaningful embeddings from the audio representations. During the training phase, the models were exposed to the 1139 audio representations in the training set, leveraging their inherent pairwise similarities and dissimilarities to optimize the embeddings. Subsequently, the models were tested using the larger testing set containing 3223 audio representations, which were unseen during the training phase.

4.2. Results

To assess the performance of each model in our study, for both Triplet Loss training and Contrastive Loss training, we have employed the computed distances utilizing the various distance metrics, as well as the corresponding scaling techniques as outlined previously. In our evaluation, a higher score signifies a superior performance of the model.
The scores achieved by each model are presented as follows:

4.2.1. Triplet Loss

Model1 - batch size = 32, learning rate = 0.00001

Distance	**Score**
chebyshev-Normalizer	50.9%
cosine-Normalizer	49.7%
minkowski-Normalizer	49.7%
cityblock-StandardScaler	49.6%
correlation-MinMaxScaler	48.3%

In the evaluation of the first model, the top-performing combinations, ranked by higher similarity scores, are as follows:
- chebyshev-Normalizer: Demonstrating a prominent performance with a similarity score of 50.9%.
- cosine-Normalizer: Achieving a score of 49.7%.
- minkowski-Normalizer: Earning a score of 49.7%.

These outcomes highlight the effectiveness of utilizing the "Normalizer" scaling technique.
For more information, see Table 1 in Appendix A.

Model2 - batch size = 32, learning rate = 0.00001

Distance	**Score**
seuclidean-MinMaxScaler	39.8%
seuclidean-StandardScaler	39.8%

euclidean-Normalizer	39.5%
cosine-Normalizer	39.5%
cityblock-MinMaxScaler	39.4%

In the evaluation of the second model, the top-performing combinations, ranked by higher similarity scores, are as follows:
• seuclidean-MinMaxScaler: Demonstrating the highest performance with a similarity score of 39.8%.
• seuclidean-StandardScaler: Achieving a score of 39.8%.
• euclidean-Normalizer: Earning a score of 39.5%.
For more information, see Table 2 in Appendix A.

Model3 - batch size = 128, learning rate = 0.00001

Distance	**Score**
cosine-Normalizer	64.1%
euclidean-Normalizer	64.1%
minkowski-Normalizer	64.1%
cityblock-Normalizer	64.0%
chebyshev-Normalizer	63.8%

In the evaluation of the third model, the top-performing combinations, ranked by higher similarity scores, are as follows:
• cosine-Normalizer: Demonstrating the highest performance with an impressive similarity score of 64.1%.
• euclidean-Normalizer: Achieving an outstanding score of 64.1%, indicating a substantial level of audio resemblance.
• minkowski-Normalizer: Earning a commendable score of 64.1%, reflecting strong capabilities in audio similarity assessment.
These results underscore, again, the notable effectiveness of the "Normalizer" scaling technique. For more information, see Table 3 in Appendix.

Model4 - batch size = 128, learning rate = 0.00001

Distance	**Score**
cityblock-Normalizer	63.2%
seuclidean-Normalizer	63.0%
cosine-Normalizer	63.0%
minkowski-Normalizer	63.0%
chebyshev-Normalizer	62.5%

In the evaluation of the fourth model, the top-performing combinations, ranked by higher similarity scores, are as follows:
• cityblock-Normalizer: Demonstrating the highest performance with a notable similarity score of 63.2%.

- seuclidean-Normalizer: Achieving a strong score of 63.0%, indicating a substantial level of audio resemblance.
- cosine-Normalizer: Earning a commendable score of 63.0%, reflecting robust capabilities in audio similarity assessment.

These results emphasize the efficacy of the "Normalizer" scaling technique in conjunction with specific distance metrics within the fourth model, facilitating effective audio similarity identification.
For more information, see Table 4 in Appendix A.

Model5 - batch size = 128, learning rate = 0.00001

Distance	Score
euclidean-MinMaxScaler	62.2%
minkowski-MinMaxScaler	62.2%
cosine-Normalizer	62.1%
euclidean-Normalizer	62.1%
chebyshev-Normalizer	61.9%

In the evaluation of the fifth model, the top-performing combinations, ranked by higher similarity scores, are as follows:
- euclidean-MinMaxScaler: Demonstrating the highest performance with a notable similarity score of 62.2%.
- minkowski-MinMaxScaler: Achieving a strong score of 62.2%, indicating a substantial level of audio resemblance.
- cosine-Normalizer: Earning a commendable score of 62.1%, reflecting robust capabilities in audio similarity assessment.

For more information, see Table 5 in Appendix A.

Model6 - batch size = 64, learning rate = 0.00001

Distance	Score
chebyshev-Normalizer	62.6%
cosine-Normalizer	61.5%
euclidean-Normalizer	61.5%
minkowski-Normalizer	61.4%
seuclidean-Normalizer	61.4%

In the evaluation of the sixth model, the top-performing combinations, ranked by higher similarity scores, are as follows:
- chebyshev-Normalizer: Demonstrating the highest performance with an impressive similarity score of 62.6%.
- cosine-Normalizer: Earning a commendable score of 61.5%, reflecting robust capabilities in audio similarity assessment.
- euclidean-Normalizer: Achieving a strong score of 61.5%, indicating a substantial level of audio resemblance.

These results highlight the efficacy of the "Normalizer" scaling technique in conjunction with specific distance metrics within the sixth model, contributing to effective audio similarity identification.
For more information, see Table 6 in Appendix A.

Model7 - batch size = 128, learning rate = 0.00001

Distance	Score
correlation-MinMaxScaler	55.9%
cosine-MinMaxScaler	55.7%
euclidean-MinMaxScaler	55.4%
minkowski-MinMaxScaler	55.4%
chebyshev-MinMaxScaler	55.3%

In the evaluation of the seventh model, the top-performing combinations, ranked by higher similarity scores, are as follows:
- correlation-MinMaxScaler: Demonstrating the highest performance with a notable similarity score of 55.9%.
- cosine-MinMaxScaler: Earning a commendable score of 55.7%, reflecting robust capabilities in audio similarity assessment.
- euclidean-MinMaxScaler: Achieving a solid score of 55.4%, indicative of a substantial level of audio resemblance.

These results highlight the efficacy of the "MinMaxScaler" scaling technique in conjunction with specific distance metrics within the seventh model, contributing to effective audio similarity identification.
For more information, see Table 7 in Appendix A.

Model8 - batch size = 128, learning rate = 0.00001

Distance	Score
euclidean-StandardScaler	48.1%
minkowski-StandardScaler	48.1%
cityblock-MinMaxScaler	47.3%
cosine-MinMaxScaler	47.3%
chebyshev-StandardScaler	46.9%

In the evaluation of the eighth model, the top-performing combinations, ranked by higher similarity scores, are as follows:
- euclidean-StandardScaler: Demonstrating the highest performance with a notable similarity score of 48.1%.
- minkowski-StandardScaler: Achieving a score of 48.1%.
- cityblock-MinMaxScaler: Earning a score of 47.3%.

For more information, see Table 8 in Appendix A.

4.2.2. Contrastive Loss

Model1 - batch size = 32, learning rate = 0.00001

Distance	**Score**
euclidean-Normalizer	48.7%
cosine-Normalizer	48.7%
cityblock-MinMaxScaler	48.3%
minkowski-Normalizer	48.7%
chebyshev-StandardScaler	48.3%

In the evaluation of the first model, the top-performing combinations, ranked by higher similarity scores, are as follows:
- euclidean-Normalizer: Demonstrating the highest performance with an impressive similarity score of 48.7%.
- cosine-Normalizer: Achieving a strong score of 48.7%.
- cityblock-MinMaxScaler: Earning a commendable score of 48.3%.

For more information, see Table 9 in Appendix A.

Model2 - batch size = 32, learning rate = 0.00001

Distance	**Score**
correlation-MinMaxScaler	51.6%
euclidean-MinMaxScaler	51.5%
minkowski-MinMaxScaler	51.4%
cityblock-MinMaxScaler	51.4%
cosine-MinMaxScaler	51.4%

In the evaluation of the second model, the top-performing combinations, ranked by higher similarity scores, are as follows:
- correlation-MinMaxScaler: Demonstrating the highest performance with an impressive similarity score of 51.6%.
- euclidean-MinMaxScaler: Achieving a strong score of 51.5%, indicating a substantial level of audio resemblance.
- minkowski-MinMaxScaler: Earning a commendable score of 51.4%, reflecting robust capabilities in audio similarity assessment.

These results highlight the efficacy of the "MinMaxScaler" scaling technique in conjunction with specific distance metrics within the second model, contributing to effective audio similarity identification.
For more information, see Table 10 in Appendix A.

4.2.3. Comparison of Best-Performing Loss Functions.

Triplet Loss (Model 3 - CNN)		**Contrastive Loss (Model 2 - ResNet50)**	
Distance	**Score**	**Distance**	**Score**
cosine-Normalizer	**64.1%**	correlation-MinMaxScaler	51.6%

In our evaluation, we observed that the Contrastive loss yielded the highest performance with a score of 51.6% when applied to our top-performing model (Resnet50). Conversely, the Triplet loss, implemented with our leading CNN-based model, achieved an impressive score of 64.1%, highlighting its efficacy in capturing intricate audio relationships.

4.3. Observations

In this master thesis, we investigated the performance of two popular loss functions, Triplet Loss and Contrastive Loss, and their effectiveness in learning meaningful embeddings for image similarity tasks. The primary objective was to compare the models trained using these loss functions and evaluate their performance using various distance metrics and scalers.
For the Triplet Loss experiments, a total of eight different models were trained using Convolutional Neural Networks (CNNs) architecture. The models were trained with the goal of optimizing the embeddings such that the anchor points are closer to their respective positive samples while being distant from negative samples. During the experimentation phase, we assessed the impact of different distance metrics and scalers on the model's performance.
Among the trained Triplet Loss models, the third model emerged as the most successful, exhibiting superior performance across several distance metrics when combined with the Normalizer scaler. Specifically, the third model achieved remarkable results with the euclidean distance-Normalizer, minkowski distance-Normalizer, and cosine distance-Normalizer. This outcome highlights the effectiveness of the Triplet Loss function when used in conjunction with CNNs for deep metric learning tasks, particularly when normalized embeddings are employed.
Turning our attention to the Contrastive Loss experiments, two distinct models were trained, leveraging the powerful ResNet50 architecture. Contrastive Loss aims to minimize the distance between similar samples and maximize the distance between dissimilar samples. Similar to the Triplet Loss experiments, we explored the impact of different distance metrics and scalers on the model's performance.
The results revealed that the second model trained with Contrastive Loss outperformed the other variant, achieving remarkable scores with the correlation distance when paired with the MinMaxScaler. The successful performance of the ResNet50 model demonstrates the ability of Contrastive Loss to yield highly discriminative embeddings for image similarity tasks.
Overall, the findings of this master thesis underscore the importance of selecting appropriate loss functions and architectures for deep metric learning tasks. While both Triplet Loss and Contrastive Loss have demonstrated their effectiveness in learning meaningful embeddings, the choice of the best model depends on the specific requirements of the application and the nature of the data. Furthermore, the use of different distance metrics and scalers during evaluation further highlights the sensitivity of the models to the choice of these parameters.
The outcomes presented in this research contribute valuable insights to the field of deep metric learning and provide a foundation for further exploration and refinement of loss functions and architectures in image similarity tasks. Additionally, the evaluation of multiple distance metrics and scalers enriches the understanding of their influence on model performance and guides researchers in making informed decisions when designing and evaluating deep metric learning models in real-world applications.

5. Conclusion

In this master thesis, we delved into the domain of deep metric learning, specifically focusing on its application to audio data in the form of audio representations of songs. The primary objective of our research was to compare the performance of two prominent loss functions, Triplet Loss and Contrastive Loss, in learning meaningful embeddings that capture the underlying similarities and dissimilarities between songs.
Throughout our investigation, we trained and evaluated multiple models using Triplet Loss and Contrastive Loss in an effort to obtain embeddings that preserve the pairwise similarities present in the audio data representations. Understanding the importance of selecting appropriate loss functions and architectures for deep metric learning tasks, we designed experiments with audio data to explore the impact of different distance metrics and scalers on model performance.
Our results shed light on the effectiveness of Triplet Loss when paired with Convolutional Neural Networks in generating discriminative audio embeddings. Notably, the third Triplet Loss model emerged as the most successful among the trained models, achieving superior scores on distance metrics like euclidean, minkowski, and cosine, especially when combined with the Normalizer scaler. This finding showcases the capability of Triplet Loss in learning audio embeddings that preserve the inherent similarities between songs, and the normalization step proved to be a crucial factor in enhancing the model's performance.
In parallel, the Contrastive Loss models, implemented with the robust ResNet50 architecture, demonstrated their potential to learn meaningful embeddings by emphasizing pairwise comparisons between audio samples. The second Contrastive Loss model excelled in preserving song similarities, particularly when evaluated with the correlation distance metric, and benefited significantly from the application of the MinMaxScaler. This outcome highlights the ability of Contrastive Loss to effectively capture the pairwise relationships present in audio data representations, offering discriminative embeddings suitable for audio similarity tasks.
The comparisons drawn between Triplet Loss and Contrastive Loss underscore the significance of selecting the most appropriate loss function depending on the nature of the audio data and the specific requirements of the task. Triplet Loss, with its emphasis on relative comparisons between samples, proved to be a potent choice, particularly when normalized audio embeddings were utilized. On the other hand, Contrastive Loss, focusing on pairwise comparisons, demonstrated its effectiveness in capturing the inherent pairwise similarities in audio data representations.
Furthermore, the evaluation of various distance metrics and scalers revealed their substantial impact on the performance of the models. The choice of distance metric should align with the nature of the audio data and the specific similarity task at hand. At the same time, selecting an appropriate scaler played a vital role in optimizing the models to learn meaningful embeddings from audio data representations.
In conclusion, our research contributes valuable insights into the domain of deep metric learning applied to audio data, particularly audio representations of songs. The findings demonstrate that both Triplet Loss and Contrastive Loss can be valuable tools for capturing song similarities in their respective ways. The knowledge gained from this study will serve as a foundation for further advancements in deep metric learning for audio similarity tasks, offering researchers and practitioners guidance in selecting appropriate loss functions, architectures, and evaluation methodologies to achieve optimal performance in various audio-related applications, such as music recommendation, audio retrieval, and content-based audio search. As the field continues to evolve, we anticipate further refinements in loss functions, network architectures, and evaluation techniques that will continue to push the boundaries of deep metric learning and its applications in the realm of audio data analysis.

6. Bibliography

[1] Chen, Q., Zhang, Y., He, X., & Sun, J. (2017). Deep metric learning with angular loss. arXiv preprint arXiv:1708.01682.

[2] Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research, 10(5), 1707-1749.

[3] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 815-823).

[4] Lerche, M. J., Malmström, J., & Winther, O. (2019). Deep infomax: Learning useful representations by maximizing mutual information. In International Conference on Learning Representations (ICLR).

[5] Liu, X., Su, H., Tan, M., Le, Q. V., & Ng, A. Y. (2020). Simclr: A simple framework for contrastive learning of visual representations. In Advances in Neural Information Processing Systems (pp. 9757-9770).

[6] Chen, X., Kornblith, S., Norouzi, M., & Le, Q. V. (2019). Moco: Momentum contrast for unsupervised visual representation learning. In Advances in Neural Information Processing Systems (pp. 9729-9742).

[7] S learns. (2016). N-pair loss: Towards better understanding of siamese networks. arXiv preprint arXiv:1606.06558.

[8] Sung, Y., Wang, X., Yang, M., & Yuille, A. L. (2018). Learning to compare: Relation network for few-shot learning. In IEEE Conference on Computer Vision and Pattern Recognition (pp. 1199-1208).

[9] Zheng, L., Shen, L., Wang, S., Tian, J., & Liu, Z. (2019). Contrastive multiview coding for person re-identification. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(11), 2625-2639.

[10] LeCun, Y., Jaitly, N., & Sabour, S. (2015). Learning audio representations with convolutional neural networks. arXiv preprint arXiv:1506.05356.

[11] Luo, Y., & Mesgarani, N. (2018, April). Tasnet: time-domain audio separation network for real-time, single-channel speech separation. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 696-700). IEEE.

[12] Gosalia, S., Shetty, S., & Revathi, A. S. (2016, March). Embedding audio inside a digital video using LSB steganography. In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom) (pp. 2650-2653). IEEE.

[13] Umapathy, K., Ghoraani, B., & Krishnan, S. (2010). Audio signal processing using time-frequency approaches: Coding, classification, fingerprinting, and watermarking. EURASIP J. Adv. Signal Process., 2010(451695). https://doi.org/10.1155/2010/451695

[14] Waldekar, S., & Saha, G. (2018, September). Wavelet transform based mel-scaled features for acoustic scene classification. In INTERSPEECH (Vol. 2018, pp. 3323-3327).

[15] Roberts, L. (2022, August 17). Understanding the MEL Spectrogram - Analytics Vidhya - Medium. *Medium*. https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53 (accessed Sep. 13, 2023)

[16] Natsiou, A., & O'Leary, S. (2021, November). Audio representations for deep learning in sound synthesis: A review. In 2021 IEEE/ACS 18th International Conference on Computer Systems and Applications (AICCSA) (pp. 1-8). IEEE.

[17] Juvasquez. (2015, May 13). Audio Signal Processing – Conclusion | Modeling and Experimental Tools with Prof. Magnes. https://pages.vassar.edu/magnes/2015/05/13/audio-signal-processing-conclusion/ (accessed Sep. 13, 2023)

[18] Casey, M. A. (1998). Auditory group theory with applications to statistical basis methods for structured audio (Doctoral dissertation, Massachusetts Institute of Technology).

[19] Yuan, Y., Xun, G., Suo, Q., Jia, K., & Zhang, A. (2017). Wave2vec: Learning deep representations for biosignals. In 2017 IEEE International Conference on Data Mining (ICDM) (pp. 1159-1164). IEEE. doi: 10.1109/ICDM.2017.155

[20] Shor, J., Venugopalan, S. (2022) TRILLsson: Distilled Universal Paralinguistic Speech Representations. Proc. Interspeech 2022, 356-360, doi: 10.21437/Interspeech.2022-118

[21] Molla, S., & Torrésani, B. (2005). A hybrid scheme for encoding audio signal using hidden Markov models of waveforms. *Applied and Computational Harmonic Analysis*, *18*(2), 137–166. https://doi.org/10.1016/j.acha.2004.11.001

[22] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[23] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770-778).

[24] LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 78(10), 2278-2324.

[25] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[26] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[27] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-9).

[28] Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4700-4708).

[29] Zhu, M., Adam, H., Kalenichenko, D., Wang, W., Andreetto, M., Howard, A. G., Weyand, T., & Chen, B. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[30] Tan, M., & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946.

[31] A. Vaswani *et al.*, "Attention Is All You Need," *arXiv.org*, Jun. 12, 2017. https:// arxiv.org/abs/ 1706.03762

[32] Hoffer, E., & Ailon, N. (2015). Deep metric learning using triplet network. arXiv preprint arXiv:1412.6622.

[33] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (2006). Dimensionality reduction by learning an invariant mapping. In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (pp. 86-94).

[34] Giannakopoulos, T. (2023). mir [Source code]. Version main. GitHub repository URL: https://github.com/magcil/mir/tree/main. (accessed Sep. 13, 2023)

7. Appendix A

Table 1

Distance	**Score**
chebyshev-MinMaxScaler	47.714%
chebyshev-StandardScaler	46.971%
chebyshev-Normalizer	50.857%
euclidean-MinMaxScaler	47.657%
euclidean-StandardScaler	48.000%
euclidean-Normalizer	49.657%
minkowski-MinMaxScaler	47.657%
minkowski-StandardScaler	48.000%
minkowski-Normalizer	49.657%
hamming-MinMaxScaler	35.257%
hamming-StandardScaler	35.257%
hamming-Normalizer	35.257%
cosine-MinMaxScaler	47.600%
cosine-StandardScaler	47.086%
cosine-Normalizer	49.657%
cityblock-MinMaxScaler	47.543%
cityblock-StandardScaler	48.000%
cityblock-Normalizer	49.600%
seuclidean-MinMaxScaler	46.343%
seuclidean-StandardScaler	46.343%
seuclidean-Normalizer	48.114%
correlation-MinMaxScaler	48.286%
correlation-StandardScaler	46.914%
correlation-Normalizer	49.371%
kulsinski-MinMaxScaler	35.771%
kulsinski-StandardScaler	35.257%
kulsinski-Normalizer	35.257%

Table 2

Distance	Score
chebyshev-MinMaxScaler	38.229%
chebyshev-StandardScaler	37.771%
chebyshev-Normalizer	38.171%
euclidean-MinMaxScaler	39.143%
euclidean-StandardScaler	38.400%
euclidean-Normalizer	39.486%
minkowski-MinMaxScaler	39.143%
minkowski-StandardScaler	38.400%
minkowski-Normalizer	39.486%
hamming-MinMaxScaler	35.200%
hamming-StandardScaler	35.200%
hamming-Normalizer	35.314%
cosine-MinMaxScaler	39.029%
cosine-StandardScaler	38.057%
cosine-Normalizer	39.486%
cityblock-MinMaxScaler	39.143%
cityblock-StandardScaler	38.914%
cityblock-Normalizer	39.429%
seuclidean-MinMaxScaler	39.771%
seuclidean-StandardScaler	39.771%
seuclidean-Normalizer	39.543%
correlation-MinMaxScaler	39.714%
correlation-StandardScaler	38.000%
correlation-Normalizer	39.371%
kulsinski-MinMaxScaler	34.686%
kulsinski-StandardScaler	35.257%
kulsinski-Normalizer	35.257%

Table 3

Distance	**Score**
chebyshev-MinMaxScaler	58.686%
chebyshev-StandardScaler	58.400%
chebyshev-Normalizer	63.829%
euclidean-MinMaxScaler	62.686%
euclidean-StandardScaler	61.657%
euclidean-Normalizer	64.057%
minkowski-MinMaxScaler	62.686%
minkowski-StandardScaler	61.657%
minkowski-Normalizer	64.057%
hamming-MinMaxScaler	35.257%
hamming-StandardScaler	35.257%
hamming-Normalizer	35.257%
cosine-MinMaxScaler	62.800%
cosine-StandardScaler	59.886%
cosine-Normalizer	64.057%
cityblock-MinMaxScaler	63.086%
cityblock-StandardScaler	62.400%
cityblock-Normalizer	64.000%
seuclidean-MinMaxScaler	62.286%
seuclidean-StandardScaler	62.286%
seuclidean-Normalizer	63.657%
correlation-MinMaxScaler	62.629%
correlation-StandardScaler	60.000%
correlation-Normalizer	63.943%
kulsinski-MinMaxScaler	35.771%
kulsinski-StandardScaler	35.257%
kulsinski-Normalizer	35.257%

Table 4

Distance	**Score**
chebyshev-MinMaxScaler	59.314%
chebyshev-StandardScaler	57.714%
chebyshev-Normalizer	62.514%
euclidean-MinMaxScaler	61.543%
euclidean-StandardScaler	61.257%
euclidean-Normalizer	62.971%
minkowski-MinMaxScaler	61.543%
minkowski-StandardScaler	61.257%
minkowski-Normalizer	62.971%
hamming-MinMaxScaler	35.257%
hamming-StandardScaler	35.257%
hamming-Normalizer	35.257%
cosine-MinMaxScaler	61.371%
cosine-StandardScaler	58.914%
cosine-Normalizer	62.971%
cityblock-MinMaxScaler	61.600%
cityblock-StandardScaler	61.086%
cityblock-Normalizer	63.200%
seuclidean-MinMaxScaler	61.771%
seuclidean-StandardScaler	61.771%
seuclidean-Normalizer	63.029%
correlation-MinMaxScaler	61.371%
correlation-StandardScaler	58.857%
correlation-Normalizer	63.029%
kulsinski-MinMaxScaler	35.714%
kulsinski-StandardScaler	35.257%
kulsinski-Normalizer	35.257%

Table 5

Distance	Score
chebyshev-MinMaxScaler	60.343%
chebyshev-StandardScaler	59.714%
chebyshev-Normalizer	61.886%
euclidean-MinMaxScaler	62.171%
euclidean-StandardScaler	61.771%
euclidean-Normalizer	62.057%
minkowski-MinMaxScaler	62.171%
minkowski-StandardScaler	61.771%
minkowski-Normalizer	62.057%
hamming-MinMaxScaler	35.257%
hamming-StandardScaler	35.257%
hamming-Normalizer	35.257%
cosine-MinMaxScaler	61.886%
cosine-StandardScaler	59.543%
cosine-Normalizer	62.057%
cityblock-MinMaxScaler	61.943%
cityblock-StandardScaler	61.771%
cityblock-Normalizer	61.943%
seuclidean-MinMaxScaler	62.286%
seuclidean-StandardScaler	62.286%
seuclidean-Normalizer	62.286%
correlation-MinMaxScaler	61.600%
correlation-StandardScaler	59.543%
correlation-Normalizer	61.943%
kulsinski-MinMaxScaler	34.971%
kulsinski-StandardScaler	35.257%
kulsinski-Normalizer	35.257%

Table 6

Distance	Score
chebyshev-MinMaxScaler	54.400%
chebyshev-StandardScaler	47.029%
chebyshev-Normalizer	62.571%
euclidean-MinMaxScaler	60.286%
euclidean-StandardScaler	58.743%
euclidean-Normalizer	61.486%
minkowski-MinMaxScaler	60.286%
minkowski-StandardScaler	58.743%
minkowski-Normalizer	61.486%
hamming-MinMaxScaler	35.257%
hamming-StandardScaler	35.257%
hamming-Normalizer	35.257%
cosine-MinMaxScaler	60.400%
cosine-StandardScaler	59.600%
cosine-Normalizer	61.486%
cityblock-MinMaxScaler	60.800%
cityblock-StandardScaler	60.171%
cityblock-Normalizer	61.029%
seuclidean-MinMaxScaler	61.314%
seuclidean-StandardScaler	61.314%
seuclidean-Normalizer	61.371%
correlation-MinMaxScaler	60.171%
correlation-StandardScaler	59.429%
correlation-Normalizer	61.429%
kulsinski-MinMaxScaler	36.000%
kulsinski-StandardScaler	35.257%
kulsinski-Normalizer	35.257%

Table 7

Distance	Score
chebyshev-MinMaxScaler	55.257%
chebyshev-StandardScaler	52.629%
chebyshev-Normalizer	50.286%
euclidean-MinMaxScaler	55.371%
euclidean-StandardScaler	53.429%
euclidean-Normalizer	51.657%
minkowski-MinMaxScaler	55.371%
minkowski-StandardScaler	53.429%
minkowski-Normalizer	51.657%
hamming-MinMaxScaler	35.200%
hamming-StandardScaler	35.200%
hamming-Normalizer	35.257%
cosine-MinMaxScaler	55.714%
cosine-StandardScaler	50.000%
cosine-Normalizer	51.657%
cityblock-MinMaxScaler	53.200%
cityblock-StandardScaler	51.543%
cityblock-Normalizer	51.486%
seuclidean-MinMaxScaler	49.486%
seuclidean-StandardScaler	49.486%
seuclidean-Normalizer	51.200%
correlation-MinMaxScaler	55.886%
correlation-StandardScaler	49.943%
correlation-Normalizer	51.657%
kulsinski-MinMaxScaler	37.200%
kulsinski-StandardScaler	35.257%
kulsinski-Normalizer	35.257%

Table 8

Distance	Score
chebyshev-MinMaxScaler	44.800%
chebyshev-StandardScaler	46.857%
chebyshev-Normalizer	43.371%
euclidean-MinMaxScaler	47.314%
euclidean-StandardScaler	48.057%
euclidean-Normalizer	45.086%
minkowski-MinMaxScaler	47.314%
minkowski-StandardScaler	48.057%
minkowski-Normalizer	45.086%
hamming-MinMaxScaler	35.257%
hamming-StandardScaler	35.257%
hamming-Normalizer	35.257%
cosine-MinMaxScaler	47.314%
cosine-StandardScaler	45.257%
cosine-Normalizer	45.086%
cityblock-MinMaxScaler	47.314%
cityblock-StandardScaler	47.314%
cityblock-Normalizer	44.800%
seuclidean-MinMaxScaler	44.914%
seuclidean-StandardScaler	44.914%
seuclidean-Normalizer	44.571%
correlation-MinMaxScaler	47.143%
correlation-StandardScaler	45.314%
correlation-Normalizer	45.086%
kulsinski-MinMaxScaler	36.229%
kulsinski-StandardScaler	35.257%
kulsinski-Normalizer	35.257%

Table 9

Distance	Score
chebyshev-MinMaxScaler	44.800%
chebyshev-StandardScaler	48.286%
chebyshev-Normalizer	47.371%
euclidean-MinMaxScaler	48.171%
euclidean-StandardScaler	48.057%
euclidean-Normalizer	48.686%
minkowski-MinMaxScaler	48.171%
minkowski-StandardScaler	48.057%
minkowski-Normalizer	48.686%
hamming-MinMaxScaler	35.257%
hamming-StandardScaler	35.257%
hamming-Normalizer	35.257%
cosine-MinMaxScaler	47.943%
cosine-StandardScaler	46.171%
cosine-Normalizer	48.686%
cityblock-MinMaxScaler	48.343%
cityblock-StandardScaler	48.114%
cityblock-Normalizer	47.657%
seuclidean-MinMaxScaler	47.829%
seuclidean-StandardScaler	47.829%
seuclidean-Normalizer	46.914%
correlation-MinMaxScaler	47.657%
correlation-StandardScaler	46.057%
correlation-Normalizer	48.571%
kulsinski-MinMaxScaler	35.543%
kulsinski-StandardScaler	35.257%
kulsinski-Normalizer	35.257%

Table 10

Distance	Score
chebyshev-MinMaxScaler	48.514%
chebyshev-StandardScaler	48.571%
chebyshev-Normalizer	45.886%
euclidean-MinMaxScaler	51.543%
euclidean-StandardScaler	50.914%
euclidean-Normalizer	50.400%
minkowski-MinMaxScaler	51.543%
minkowski-StandardScaler	50.914%
minkowski-Normalizer	50.400%
hamming-MinMaxScaler	35.257%
hamming-StandardScaler	35.257%
hamming-Normalizer	35.257%
cosine-MinMaxScaler	51.371%
cosine-StandardScaler	46.686%
cosine-Normalizer	50.400%
cityblock-MinMaxScaler	51.371%
cityblock-StandardScaler	50.629%
cityblock-Normalizer	49.771%
seuclidean-MinMaxScaler	50.400%
seuclidean-StandardScaler	50.400%
seuclidean-Normalizer	47.429%
correlation-MinMaxScaler	51.600%
correlation-StandardScaler	46.743%
correlation-Normalizer	50.000%
kulsinski-MinMaxScaler	34.971%
kulsinski-StandardScaler	35.257%
kulsinski-Normalizer	35.257%