Multimodal Pretraining for Music Audio

by

Andreas Sideras

Submitted
in partial fulfillment of the requirements for the degree of

Master of Artificial Intelligence

at the

UNIVERSITY OF PIRAEUS

July 2024

Author .....

Andreas Sideras

II-MSc “Artificial Intelligence”

July, 2024

Certified by.....

Theodoros Giannakopoulos
NCSR Demokritos, Researcher
Thesis Supervisor

Certified by.....

Ilias Maglogiannis
University of Piraeus, Professor
Member of Examination Committee

Certified by.....

Ilias Zavitsanos
NCSR Demokritos, Researcher
Member of Examination Committee

Multimodal Pretraining for Music Audio

By

Andreas Sideras

Submitted to the II-MSc "Artificial Intelligence" on Month July, 2024, in partial fulfillment of the requirements for the MSc degree

Abstract

Data can be expressed in various forms, each potentially encoded through diverse means. For instance, we might encounter audio data paired with descriptive texts about their lyrics. Modern systems leverage, if available, the different sources of information and outperform, under certain conditions, their single-modal counterparts. In such multimodal settings, each modality encapsulates a distinct aspect of the underlying semantics of the data and has a supplementary role. Data can also be limited and without annotations related to the task at hand. In such cases, transfer learning and pretraining could be two techniques that enhance the performance of the models. In this thesis, we explore various unsupervised pretraining techniques while evaluating them on a supervised downstream task. Our goal is to train a model that can extract meaningful features and be further finetuned to any new task. We use LLMs to create pseudo-captions that describe the sentiment and the theme of the lyrics, from a large pool of non-annotated audio. We then perform a pretraining step, where we learn a multimodal coordinated space between the audio signals and these pseudo-captions. Then, we finetune our model on an annotated dataset, where only the audio modality is available. We highlight the ability of such models to deliver adequate performance in few-shot learning settings, the incorporation of LLMs into the pretraining step, and the importance of learning a shared semantic space for information originating from different modalities.

Thesis Supervisor: Theodoros Giannakopoulos
Title: Principal Researcher at NCSR Demokritos

Code: https://github.com/asideras/MIR-Natural-Language-Supervision

Acknowledgments

I would like to express my gratitude to all the professors of the Master's course for their support and knowledge sharing over the past two years, and especially Mr. Theodoros Giannakopoulos for delivering his passion and expertise for multimodal machine learning. I am also deeply grateful to my family for their constant support.

Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the University of Piraeus and Inst. of Informatics and Telecom. of NCSR "Demokritos".

Contents

List of Figures

List of Tables

1 Introduction

Nowadays, the abundance of data and the advancement in computational resources have made training deep neural networks possible. These models have huge expressive power and outperform traditional machine learning models in several settings. However, the definition of a proper hypothesis in this space is not trivial. We often need both big amounts of data and an appropriate inductive bias to guide our model to regions that fit our problem. For example, computer vision tasks are a success story of deep learning because they successfully address these requirements. They combine the large pool of easily accessible visual content data with the translation invariance bias of CNNs.

The quality and quantity of data is of utmost importance regarding the performance of machine learning systems. Quality data ensures that the patterns and insights derived from it are accurate and reliable, thereby enhancing the effectiveness of the model's predictions. The quality of the available data samples depends on the features that are used to describe them. These features should be informative enough to give a solid, thorough description of the underlying semantics of data. On the other hand, quantity matters because it provides a broader spectrum of examples for the model to learn from, enabling it to capture more nuanced relationships and achieve greater generalization ability. In addition, data can be encoded in various means and contain multiple constituents that capture different dimensions of the underlying semantics. Modern systems combine multiple modalities (e.g. audio, text, image), if available, and achieve superior performance to models that use just one source of information. Nonetheless, the fusion of such information is not trivial and constitutes an open research field.

Still, even in modern times with such a plethora of information, certain tasks lack enough data or annotations, something that may be crucial to the task at hand. Addressing the challenges of limited data in machine learning

necessitates a strategic approach, with *pretraining* emerging as a pivotal technique. By initially training models on related tasks or larger datasets, pretraining provides a foundation of knowledge that can be *finetuned* with the available limited data, bolstering model performance and generalization. Complemented by strategies such as data augmentation, transfer learning, and data synthesis, pretraining amplifies the effective dataset size, fostering robust model development. In addition, it is quite common nowadays to utilize big pretrained models, like LLMs, to create artificial samples, summarize, pseudo-annotate, or augment the available data, and then incorporate these artifacts into the training process.

In this thesis, we explore how a *multimodal pretraining* step would affect the performance of a *downstream task*, where only one modality is available, and data are limited. The downstream task is genre classification. We carefully gathered an annotated set of audios and labels, that concern songs and genres which are quite difficult to distinguish. We believe that the theme and sentiment of the lyrics of these songs, even though not explicitly available, could guide the model to make more accurate predictions. Thus, we acquired more, unlabeled data, namely music audios (concerning any genre) along with their lyrics, and we asked a LLM to derive pseudo-captions from them. These captions, comment on the sentiment and general theme of the lyrics, and features we find relevant to our task. We then experiment with various techniques, primarily involving *metric learning*, that try to correlate the audio signals to these text captions. During this pretraining step, the audio encoding part of the model learns features from the audio signal that are aligned with the sentiment and theme of the tracks. We then tested this component on the downstream task and achieved a significant increase in performance. What's important to highlight, is that this pretraining step is not necessarily bound to the genre classification task, and thus both the audio and text encoder components could be used to address any task that implicitly involves the sentiment and theme of the lyrics.

2 Background

We initiate by exploring the representation of audio signals as inputs for machine learning models, followed by an examination of state-of-the-art techniques in Natural Language Processing (NLP). Progressing further, we delve into the challenges and advancements in handling multimodal data, where models are tasked with learning representations and making predictions from diverse information sources. Lastly, we reflect on the transformative impact of the pretraining process on the landscape of machine learning.

2.1 Audio Representation Learning

A sound signal is produced by variations in air pressure [1]. We can measure the intensity of the pressure variations and plot those measurements over time. Sound signals often repeat at regular intervals so that each wave has the same shape. The height shows the intensity of the sound and is known as the *amplitude*.

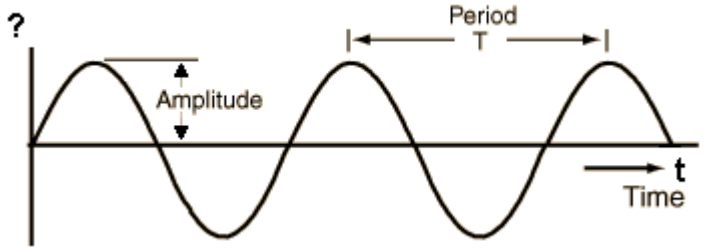

Figure 1 - Sound periodic signal [1].

The *period* of a signal refers to the duration it takes to complete a single wave, while the *frequency* denotes the number of waves generated by the signal within one second. Frequency is essentially the inverse of the period and is measured

in Hertz. Although many sounds we come across may not exhibit such straightforward periodic behavior, combining signals of various frequencies can produce composite signals with intricate repeating patterns. All audible sounds, including the human voice and musical instruments, are comprised of such waveforms.

To convert a sound wave into digital form, we employ a process where we quantify the amplitude of the sound at regular time intervals. This involves transforming the continuous analog signal into discrete numerical data points, enabling us to feed it into computational models. By sampling the amplitude at fixed intervals, we capture the essential characteristics of the sound wave, facilitating its representation and analysis within digital systems.

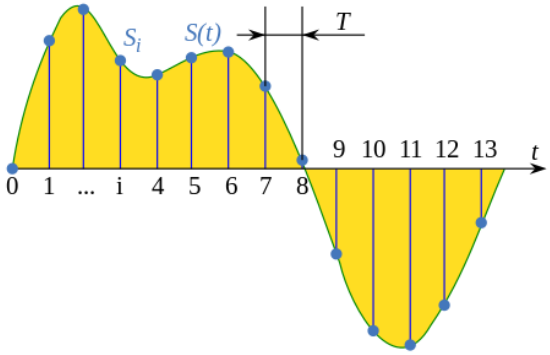

Figure 2 - Signal sampling (Wikipedia)

Every individual measurement taken in this process is referred to as a *sample*, and the *sample rate* denotes the quantity of samples captured within a single second.

Directly feeding the raw signal into a machine learning model can pose challenges due to the nature of the data. Sound signals are continuous and analog, containing a vast amount of information. Machine learning models typically operate on structured data, which necessitates converting the raw

signal into a format that the model can interpret effectively. Thus, deriving a set of features that effectively describe the audio signal is necessary [2-4]. In the realm of older machine learning methodologies, *domain knowledge* played a pivotal role in extracting audio features from sound signals. Traditional approaches often relied on handcrafted feature extraction techniques, where experts in signal processing and audio engineering meticulously designed algorithms to capture relevant characteristics of the audio data. These features encompassed various aspects such as frequency content, temporal dynamics, and spectral properties, each tailored to specific tasks like speech recognition or music classification. Engineers and researchers leveraged their understanding of signal theory and domain-specific insights to engineer features that encapsulated the discriminative information crucial for the task at hand. While effective, this approach required substantial domain expertise and manual effort to devise feature extraction pipelines tailored to individual applications. Despite its limitations in scalability and adaptability to diverse datasets, this domain-driven feature engineering paradigm laid the foundation for early successes in audio analysis and served as a precursor to more data-driven and automated approaches in modern machine learning. An example of such a feature is the *zero-crossing rate*, a measure of the number of times in a given time interval/frame that the amplitude of the speech signals passes through a value of zero.

Modern machine learning audio systems tend to avoid manually extracted features due to several reasons. Firstly, manually crafting features requires domain expertise and can be time-consuming, hindering scalability across diverse audio domains. Secondly, manually engineered features may not capture all relevant information present in the data, limiting the model's ability to generalize effectively. Moreover, with the advancements in deep learning, end-to-end learning approaches using raw audio inputs or image representations, have demonstrated superior performance by automatically

extracting discriminative features directly from the data, thereby reducing the reliance on handcrafted features.

Modern representations encapsulate how the signal changes over time and frequency. *Time domain* analysis deals with examining the behavior of a signal as a function of time. It provides information about how the signal changes over time. *Frequency domain* analysis involves examining the behavior of a signal in the frequency domain. It provides information about the frequency components present in the signal and their respective magnitudes and phases. Time domain analysis provides the transitory response of a system to be analyzed, and it permits a better understanding of the flow of both mechanical and electrical energies [5]. Whereas for the frequency domain, visualization tools such as a *spectrum analyzer* are commonly in use when visualizing electronic signals. Also, some specialized signal processing techniques make use of transforms, and this results in a joint time-frequency domain. Moreover, the instantaneous frequency is a critical link between the time domain and the frequency domain.

The *Fourier transform* is a mathematical formula that allows us to decompose a signal into its individual frequencies and the frequency's amplitude [7,8]. In other words, it converts the signal from the time domain into the frequency domain. The result is called a *spectrum*. This happens because any signal can be broken down into a bunch of sine and cosine waves that when you add them up, make the original signal.

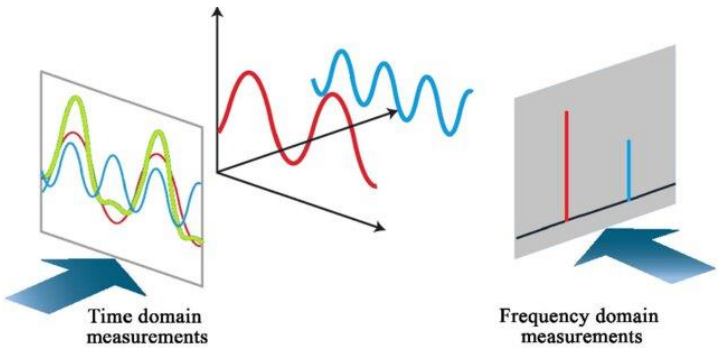

Figure 3 - Time vs Freq. domain [5]

Initially, the signal is divided into small *segments*, typically *overlapping [7-9]*. Then, each segment undergoes *windowing*, where a window function is applied to reduce spectral leakage and artifacts at the edges of the segment. Common window functions include Hanning, Hamming, and Gaussian. After windowing, Fast Fourier Transform (FFT) is applied to each segment, converting it from the time domain to the frequency domain. Finally, these FFT results are stacked together to form the *spectrogram*, where the intensity of each frequency component is represented by its color or brightness, with time along the x-axis and frequency along the y-axis. The spectrogram serves as a visual depiction of how a signal's volume, or intensity, changes over time across various frequencies. When generating a spectrogram, additional processes are involved behind the scenes. Notably, the y-axis is transformed into a logarithmic scale, and the color representation is converted to decibels, akin to amplifying the log scale of the intensity. This adjustment accounts for human perception, which is sensitive to only a narrow and specific range of frequencies and intensities. An example is depicted below.

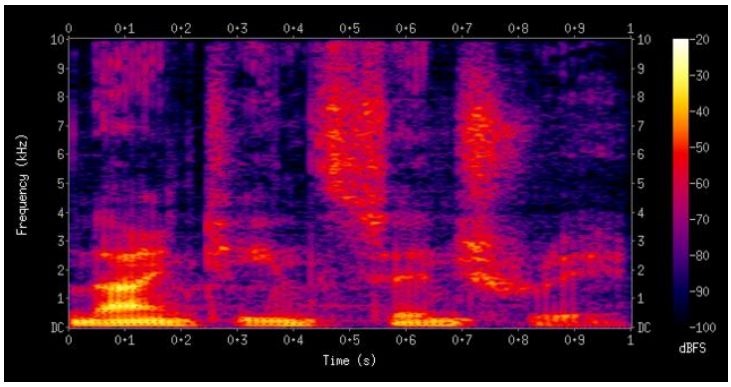

Figure 4 - Spectrogram [8]

The *Mel scale* is a perceptual scale of pitches that approximates the human ear's response to different frequencies. *Mel spectrograms* are spectrograms where

the frequencies are converted from the linear scale to the Mel scale, providing a more accurate representation of how humans perceive sound.

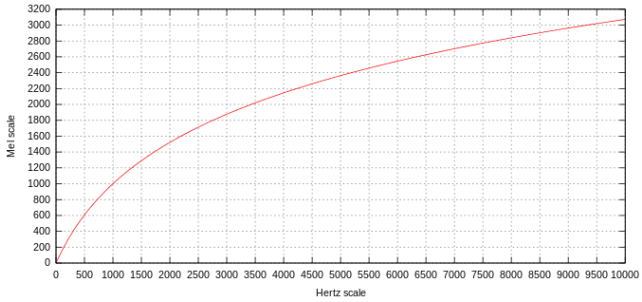

Figure 5 - Mel scale [9]

Mel spectrograms are versatile representations of audio data that can be effectively utilized as inputs for deep learning models [10]. By converting the frequency axis to the Mel scale, these spectrograms align more closely with human auditory perception. Conceptually, Mel spectrograms can be viewed as tensors or 2D images, where each element represents the intensity of a specific frequency band at a given time frame. This tensor-like structure allows them to be integrated into deep learning frameworks. Deep models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), can readily process Mel spectrograms as input data. CNNs are adept at extracting hierarchical spatial features from spectrograms, treating them as images, while RNNs excel in capturing temporal dependencies across successive frames. Tasks benefiting from Mel spectrogram inputs include but are not limited to speech recognition, music classification, sound event detection, and environmental sound analysis. Their compatibility with deep learning architectures and ability to encapsulate salient audio features make Mel spectrograms a preferred choice for numerous audio-related applications.

2.2 Text Representation Learning

Natural language processing (NLP) endeavors to construct language-specific algorithms enabling machines to comprehend and utilize human languages. Traditional NLP techniques heavily lean on feature engineering to construct semantic representations of text, demanding meticulous design and substantial expertise. In contrast, representation learning strives to autonomously generate insightful representations of raw data for subsequent utilization and has gained notable success recently.

Studying representation schemes in NLP often begins with words, as they are the fundamental units of natural languages. One common method for representing words in a computer-readable format is through *one-hot vectors*, where each word is assigned a binary value (denoting a word's existence in the sequence) in a vector equal to the size of the vocabulary. However, one-hot vectors lack semantic information beyond distinguishing words. These vectors can be extended to represent entire documents using *bag-of-words (BOW)* [11] models, treating documents as unordered collections of words. BOW models are effective in various applications such as spam filtering and text classification, relying on the distribution of words for representation. Another approach is through word representation learning, such as n-gram models [12], which utilize contextual information to predict the next word in a sentence. The concept of n-gram models aligns with the *distributional hypothesis* [11], which posits that language elements with similar contextual distributions also have similar meanings. In simpler terms, it emphasizes that a word's meaning is defined by the context in which it is used, often expressed as “a word is characterized by the company it keeps”.

Modern NLP approaches are based on the distributional hypothesis and try to learn word representations by projecting words into multidimensional spaces. Such projections are called *embeddings*. These are vectors that

incorporate the meaning of each word. Thus, words with similar meanings should be placed closely into the learned space.

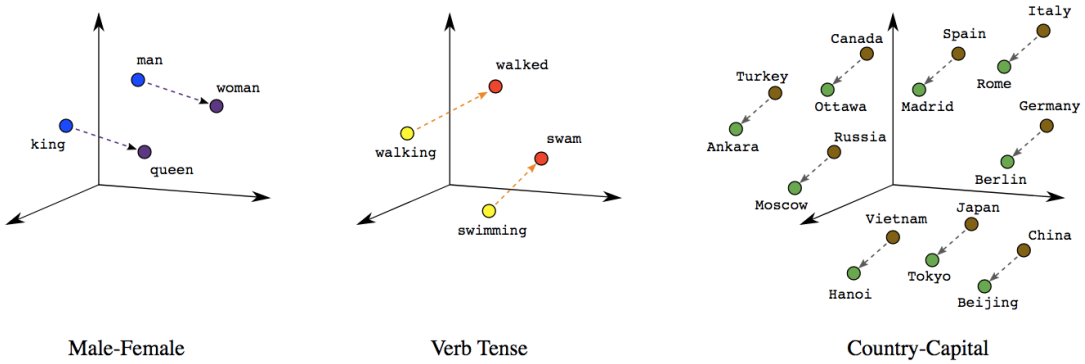

Figure 6 - Word Embeddings [13]

Word embeddings are created through techniques that map words from a high-dimensional space (such as a one-hot encoded vector space) into a continuous, lower-dimensional space where the geometric relationships between words represent their semantic similarities. One common method for creating word embeddings is through neural network-based models like Word2Vec [14] and GloVe [15]. These models typically utilize large corpora of text data to learn the embeddings. In Word2Vec, for example, the model learns to predict a target word based on its surrounding context words (skip-gram model) or predicts context words given a target word (continuous bag of words model - CBOW) as depicted in Figure 7 - CBOW & Skip Gram Objectives [16]. Through this training process, the model adjusts the word embeddings to capture semantic relationships between words. Word embeddings are versatile tools used across various NLP tasks. In RNNs, word embeddings serve as inputs, allowing the network to understand semantic relationships and context within sequential data like text. CNNs benefit from word embeddings by capturing local patterns and features in text data, enhancing their ability to perform tasks like sentiment

analysis or text classification with improved accuracy and efficiency. Additionally, word embeddings enable dimensionality reduction techniques, aiding in the visualization and clustering of text data for exploratory analysis or information retrieval systems.

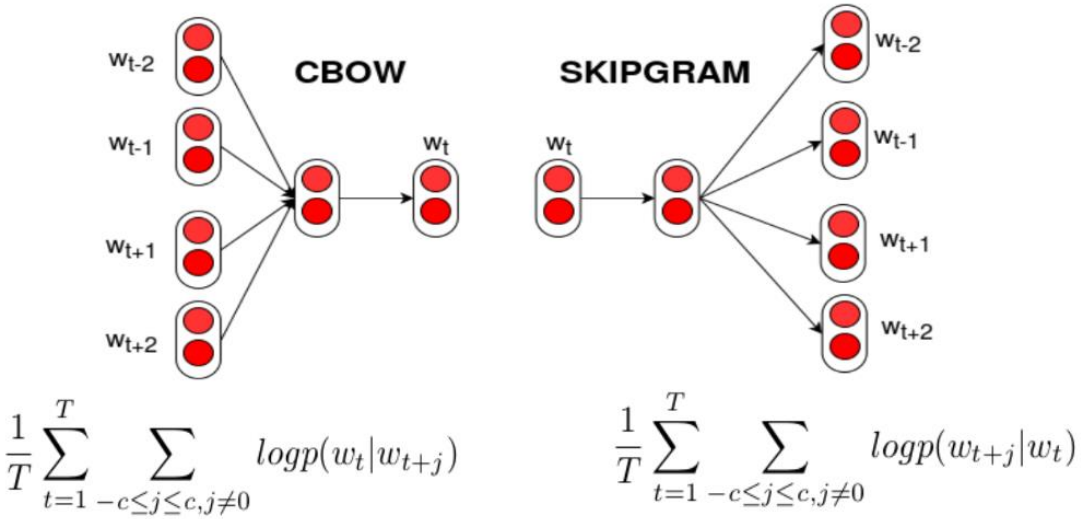

Figure 7 - CBOW & Skip Gram Objectives [16]

Traditional word embedding models like Word2Vec and GloVe generate a unique embedding for each word, regardless of its context in a sentence or document. This approach lacks awareness of the nuanced meaning that words can have in different contexts. For example, the word “play” can be used as a verb (do something for amusement) or noun (theatrical performance) and thus should have different representations in each of these contexts. *Contextual word embeddings* [17], on the other hand, are essential because they capture the variability in word semantics based on their surrounding context within a text. By considering the entire input sequence, contextual embeddings provide dynamic representations that adjust according to the context, enabling them to capture polysemy, syntactic structures, and other nuances in meaning. This contextual awareness makes them crucial for tasks such as sentiment analysis,

named entity recognition, and machine translation, where understanding the meaning of words within their specific contexts is vital for accurate language understanding and generation.

[18] Since 2017, a more powerful neural architecture, the Transformer [19] model, which is equipped with a self-attention mechanism, has received extensive attention from the NLP community. Compared to RNNs, Transformers could handle sequential data in parallel instead of processing a word at a timestep. They extract contextualized embeddings and make use of the *attention mechanism* [20], something that enables each word to focus on separate parts within the text sequence. [18] A Transformer is a nonrecurrent encoder-decoder architecture with a series of attention-based blocks. For the encoder, there are multiple layers, and each layer is composed of a multi-head attention sublayer and a position-wise feed-forward sublayer. And there is a residual connection and layer normalization of each sublayer. The decoder also contains multiple layers, and each layer is slightly different from the encoder. First, sublayers of multi-head attention and feed-forward with identical structures with the encoder are adopted. The input of the multi-head attention sublayer is from both the encoder and the previous sublayer, which is additionally developed. This sublayer is also a multi-head attention sublayer that performs self-attention over the outputs of the encoder. The sublayer adopts a masking operation to prevent the decoder from seeing subsequent tokens. The architecture of the Transformer is shown in Figure 8 - Transformer Architecture [19].

Inspired by Transformer architecture, BERT [21] (Bidirectional Encoder Representations from Transformers) is a model that uses the encoder part of the network. It is responsible for generating contextualized embeddings for the whole input sequence. BERT was trained through a process of pretraining followed by finetuning. In the pretraining phase, BERT was exposed to vast amounts of text data and learned to predict missing words in sentences (Masked Language Model task) and to determine whether pairs of sentences followed

each other (Next Sentence Prediction task). This pretraining endowed BERT with a deep understanding of language structure and context. Following pretraining, BERT can be finetuned on specific tasks by further training it on labeled data relevant to the task at hand.

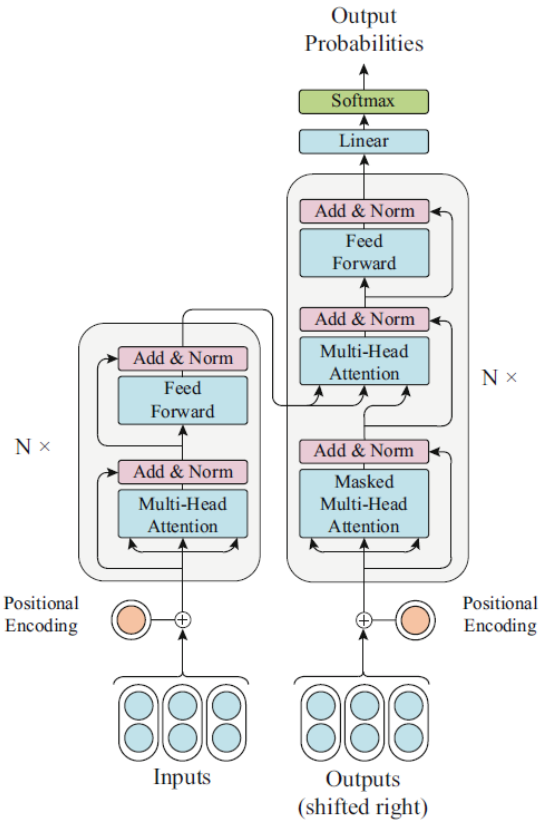

Figure 8 - Transformer Architecture [19]

BERT outputs one embedding per token. It also outputs two special tokens, the [CLS] and [SEP] tokens. The [CLS] token stands for “classification”. It is always inserted at the beginning of the input sequence. During finetuning for classification tasks, the output corresponding to this [CLS] token is used as a summary representation of the entire input sequence. This representation can then be passed to a classification layer (e.g., a softmax layer) to make predictions for the classification task. In essence, the [CLS] token serves as a

special marker indicating that the model should produce a single vector representation suitable for classification based on the input sequence. The [SEP] token stands for “separator”. It is used to separate two sequences in BERT input. When working with a single sequence (as in classification tasks), the [SEP] token is appended to the end of the input sequence. When working with two sequences (as in question answering or natural language inference tasks), the [SEP] token is used to separate the two sequences. The [SEP] token helps BERT distinguish between different segments of text and learn relationships between them.

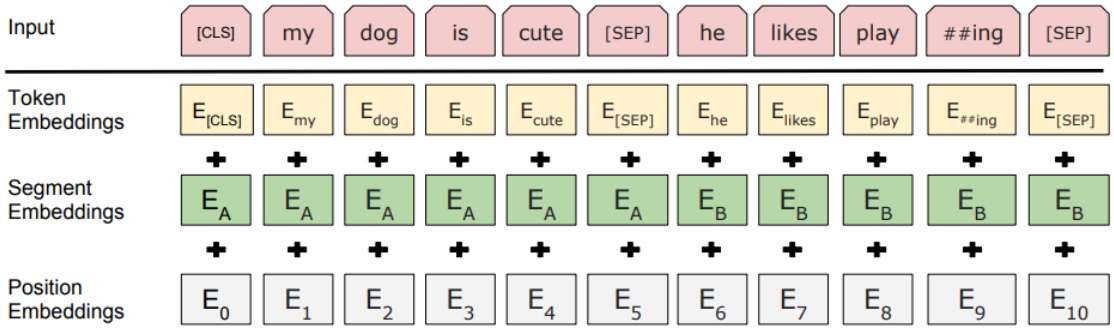

Figure 9 - Input/Output Tokens [19]

2.3 Multimodal Settings

Modern data are often represented in multiple formats, and so, modern machine learning systems aim to fully leverage them. To present thorough information about objects in the world, diverse cognitive signals detailing various aspects of the same object are captured across different media types, including text, images, video, sound, and graphs. In the domain of representation learning, the term *“modality”* pertains to a specific method or mechanism of encoding information. Consequently, the various media types

mentioned earlier (text, image, etc.) also fall under modalities, and representation learning tasks that involve multiple modalities are described as *multimodal*. Because multimodal data perceive an object from various perspectives, often with complementary or supplementary content, they offer more comprehensive information compared to unimodal data.

There are various ways to address multimodal settings, and the best choice depends on the task at hand. Although a variety of different multimodal representation learning models may share similar architectures, the essential components used for extracting *modality-specific features* could be quite different from each other. Each component adds a different inductive bias, something that facilitates them to encode the modality properly. RNNs [22] and CNNs [23] are successfully used to encode time series and images respectively. These two modalities, exhibit patterns that typically hold *translation invariance* in time and space. The *parameter-sharing* mechanism property of these models, matches the translation invariance property of these modalities, by constraining their *expressing power*. This constraint guides the derived hypothesis to spaces that sufficiently approximate the function we try to learn. So, in our multimodal scenario, the most common strategy is to give each modality as input to a model able to capture its special properties.

After encoding each modality with a modality-specific model, there should be an integration of their features. This is not trivial though, because there is something called *heterogeneity gap* [24]. Since feature vectors from different modalities are originally located in unequal subspaces, vectors associated with similar semantics would differ greatly. This discrepancy, known as the heterogeneity gap, impedes the comprehensive utilization of multimodal data by subsequent machine learning modules. A common approach to tackle this issue involves projecting the heterogeneous features into a common subspace, where multimodal data with *similar semantics* are represented by *similar vectors*. Thus, the main goal of multimodal representation learning is to reduce the

distribution gap in a shared semantic subspace while preserving modality-specific semantics.

There are various ways to incorporate features from different modalities. A natural extension of this strategy in a multimodal setting is the utilization of *fused heterogeneous features*. To overcome the heterogeneity gap between various modalities, *joint representation* endeavors to transform unimodal representations into a unified semantic subspace. This enables the fusion of multimodal features. This strategy involves, after encoding each modality with a separate neural network, mapping all of them into a shared subspace. Another type of method popular in multimodal learning is *coordinated representation*. Instead of learning a joint representation, a model learns a different space for each modality but keeps a correspondence between them. As the information within different modalities varies, acquiring separate representations proves advantageous for preserving the distinctive and valuable characteristics specific to each modality. Lastly, a popular framework also is the Encoder-Decoder architecture, where one modality is translated into another. All these setups are depicted in Figure 10 - (a) Joint Representation (b) Coordinated Representation (c) Encoder – Decoder [24].

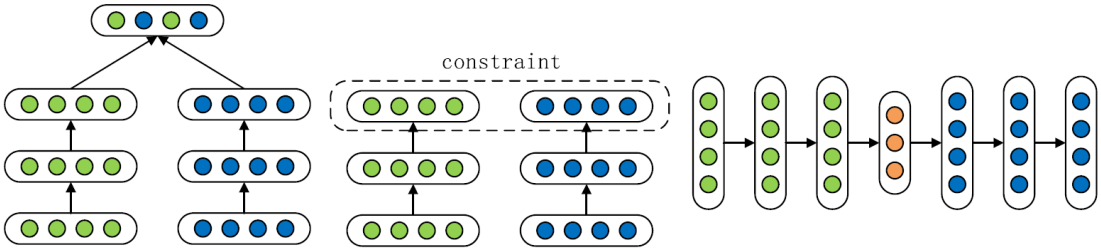

Figure 10 - (a) Joint Representation (b) Coordinated Representation (c) Encoder – Decoder [24]

Understanding the potential of multimodal machine learning opens the door to a wide array of innovative applications that leverage its capabilities. By

integrating and processing data from various sources such as text, images, audio, and more, multimodal models can achieve a more comprehensive understanding and deliver sophisticated solutions across different domains. These applications span diverse fields, revolutionizing areas such as medical care, entertainment, security, and beyond.

A family of profound models used in the multimodal settings is the *probabilistic graphical models*. These models primarily include deep belief networks (DBN) [25] and deep Boltzmann machines (DBM) [26] and their objective is to maximize the joint distribution over modalities. These models uncover the cross-modal correlations and can be trained in an unsupervised fashion. Several applications have arisen based on probabilistic graphical models such as audio-visual emotion recognition [27] and audio-visual speech recognition [28].

Autoencoders [29], renowned for their ability to learn data representations without labeled data, include two key components: an encoder that compresses input data into a latent representation, and a decoder that reconstructs the original input from this compressed form, minimizing reconstruction loss. Extending this concept to multimodal scenarios, Ngiam et al. [30] introduced a bimodal deep autoencoder capable of learning shared representations across audio and video modalities. In their model, separate autoencoders for each modality are combined at a common latent representation layer, allowing for robust cross-modal reconstruction even when one modality is absent. Silberer and Lapata [31] expanded on this by incorporating a classification loss alongside the reconstruction loss, enhancing the model's ability to distinguish between different objects based on the learned representations. Wang et al. [32] further introduced orthogonal regularization on weights to minimize redundancy in the learned representations. The primary advantage of autoencoders lies in their ability to preserve the dominant semantic information of input data, as the latent representation encodes critical factors for generating the input. Additionally, their unsupervised training approach negates the need for labeled

data. However, to enhance performance for specific tasks, autoencoders often require additional constraints or a supervised learning process. This adaptability makes them highly valuable for a wide range of applications in multimodal machine learning, where integrating diverse data types is crucial.

GANs, pioneered by Goodfellow et al. in 2014 [33], have achieved remarkable success in various unimodal applications, such as image synthesis, image-to-image translation, and image super-resolution, owing to their unique architecture comprising a generator and a discriminator in a competitive framework. Extending this framework to multimodal tasks has proven effective in addressing the challenges inherent in these domains. GANs have been extended to multimodal tasks to address the inherent challenges in such domains. For example, in text-to-image synthesis [34], GANs effectively encode textual descriptions into visual concepts, enabling the generation of coherent images from textual inputs. Similarly, GANs have been applied to visual captioning [35], where they generate descriptive text based on visual input, and cross-modal retrieval [36], facilitating the retrieval of relevant items across different modalities. The use of GANs in multimodal storytelling demonstrates their ability to integrate various forms of data, producing comprehensive narratives that combine multimodal elements. These applications underscore the versatility and potential of GANs in enhancing multimodal representation learning by generating high-quality data and bridging gaps between different data modalities.

2.4 Pretraining

Pretraining in machine learning is a technique where a model is first trained on a large dataset using unsupervised or self-supervised learning methods before finetuning it on a specific task with labeled data. The idea behind pretraining is to initialize the model with knowledge learned from a broad range of data,

which helps the model capture general patterns and features that can be useful for a variety of tasks. This initialization typically involves training the model on a large dataset, such as a corpus of text, a collection of images, or a combination of various data types. Once the model has been pretrained, it can be finetuned on a smaller dataset related to a specific task of interest, such as image classification, natural language processing, or speech recognition. Finetuning involves updating the parameters of the pretrained model using labeled data from the target task, allowing the model to adapt its representations to better suit the task at hand. Pretraining is particularly useful in scenarios where labeled data for a specific task is limited or expensive to acquire. By leveraging pretrained models, researchers and practitioners can benefit from the general knowledge encoded in the pretrained model and achieve better performance with less labeled data.

Pretraining can be achieved via unsupervised and self-supervised methods. In unsupervised pretraining, the model learns to represent the data without any explicit supervision signal. This can involve tasks such as autoencoding or generative modeling. Autoencoders, for example, learn to reconstruct their input data from a compressed representation learned by the model.

Self-supervised learning is a form of unsupervised learning where the supervision signal is generated from the data itself. In self-supervised pretraining, the model is trained to solve a task that is automatically generated from the input data. For example, in computer vision, self-supervised pretraining tasks might involve predicting the relative position of image patches or generating image rotations or distinguishing between images and their corresponding augmentation.

Deep metric learning is a subfield of machine learning that focuses on learning representations of data such that *similar instances* are embedded closer together in a high-dimensional space [37]. This approach is particularly useful for tasks like image retrieval, face recognition, and recommendation systems. In the context of pretraining, deep metric learning can serve as a

crucial step to initialize the model with meaningful representations before finetuning on a downstream task. Namely, we can define a pretraining objective by using a criterion from the metric learning domain. The core of metric learning is the notion of similarity. Especially in unsupervised settings, it is not clear which samples should be considered similar or not. Several engineering tricks have been applied to address this issue, like augmentation in computer vision tasks. A pair of one original image sample with its augmentation constitutes a *positive pair*, while a random pair a *negative pair*. By trying to learn a space where a model embeds positive pairs closely while pushing negative pairs apart, the model can learn quite informative, general features from the data.

One very popular criterion is called *triplet loss* [38]. Given one anchor input x, we select one positive sample x^+ and one negative x^-. Triplet loss learns to minimize the distance between the anchor and positive while maximizing the distance between the anchor and negative at the same time with the following equation:

$$\mathcal{L}(x, x^+, x^-) = \sum_{x \in \mathcal{X}} max\,(0, ||f(x) - f(x^+)||_2^2 - ||f(x) - f(x^-)||_2^2 + \epsilon)$$

where the margin parameter e is configured as the minimum offset between distances of similar and dissimilar pairs and $f(x)$ is the output of a neural network.

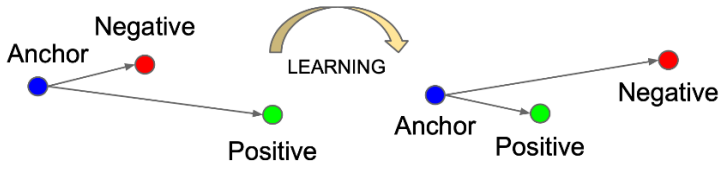

Figure 11 - Triplet Loss

A generalization of triplet loss is *Multi-class N-pair loss* [39]. This approach includes the comparison between one positive and multiple negative pairs. By including multiple negative pairs, the model encounters a broader range of negative examples, making the training more robust. This ensures that the model doesn't overfit to a limited set of negative examples and learns to discriminate between various instances more effectively.

$$\mathcal{L}_{\text{N-pair}}(x, x^+, \{x_i^-\}_{i=1}^{N-1}) == -log\frac{\exp\left(f(x)^\top f(x^+)\right)}{\exp\left(f(x)^\top f(x^+)\right) + \sum_{i=1}^{N-1} \exp\left(f(x)^\top f(x_i^-)\right)}$$

Multimodal systems can also leverage such pretraining techniques. Such models follow the same pretraining principles like unimodal models, but they may incorporate the multimodality factor into their pretraining procedure objectives. For instance, a popular pretraining method is the *modality matching* task, where the model tries to match the features of one modality to the other. For instance, Unicoder-VL [40] utilizes the Visual-linguistic Matching (VLM) for vision-language pre-training. They extract the positive and negative image-sentence pairs and train their model to predict whether the given sample pairs are aligned or not (in other words, to predict the matching scores). In addition, after pretraining, all the single-modality encoder components could be used together or separately, to address any downstream task. If a downstream model uses a subset of them, we could see the pretraining procedure to instill some external, modality-oriented knowledge. For example, WAV2CLIP's [41] pretraining objective was to learn a coordinated space between images and audio signals. After pretraining, they tested the Audio Encoder component on various downstream tasks requiring a single modality. Finally, learning such coordinated spaces between different modalities enables these systems to perform cross-modal retrieval, by using simple distance metrics. CLIP [42] model also exhibits outstanding performance on various tasks by using *zero-*

shot predictions. After learning a coordinated space between images and text, they encode the labels for a given classification task using the text encoder part of the network. Then, they make predictions by measuring the distance of the embeddings of the audio and the embeddings of the labels.

3 Related Work & Contribution

In this section, we provide an overview of the related work concerning our approach. We examine methods and systems utilizing metric learning to learn coordinated representations across modalities that aim to address specific tasks, like cross-modal retrieval, music caption generation and genre classification. We highlight the challenges encountered and outline open research directions.

Recently, plenty of research has focused on multimodal settings that use various metric learning approaches. Many of them, explore the cross-modal retrieval task, where a text (or audio) query must retrieve a set of relevant results of audio (or texts). Conventional music search engines typically use methods for finding music that involves matching natural language queries with music metadata. However, there's been a growing push to broaden these methods to also incorporate the audio features of music itself, using queries in different forms like text, video, and speech. While many of these approaches aim to align with the overall meaning of the music in response to the input queries, only a handful specifically target the emotional qualities. In [43] they used contrastive learning to learn an emotion-aligned joint embedding space between images and music and performed image-to-music and music-to-image retrieval. [44] Follows a similar approach, but they distill explicitly textual information by using the CLIP pretrained model as a frozen image encoder.

They also performed a thorough examination of various downstream tasks and a qualitative assessment of the learned space by performing image generation conditioned on music and text.

Textual representation can be an extremely effective way to describe the semantics of another modality. Information like genre, mood, lyrics, and general metadata could be written explicitly. Thus, various models use text as a supplementary modality to music. In [45] they propose a method called MUSER that performs a sort of 3-way contrastive learning between audio, spectrum, and text. They experimented with predefined text templates like “a song of {genre}, belongs to {tag}, whose style is {style}” and assessed their methods on genre classification and automatic tagging tasks. In [46] the paper explores design strategies for enhancing text-to-music retrieval systems, emphasizing the importance of accommodating diverse input queries such as pre-defined tags and sentence-level descriptions. It critiques previous works, which often focused on a singular query type, hindering generalizability across different input formats. Through a proposed benchmark, recent text-based music retrieval systems are evaluated based on input text representation and training objectives, resulting in a universal retrieval system capable of achieving comparable performances across tag and sentence inputs. Additionally, the proposed multimodal representation demonstrates effectiveness across nine distinct downstream music classification tasks, indicating broader applicability. The paper in [47] addresses Automated Audio Captioning (AAC) by proposing CLIP-AAC, a novel system that integrates both acoustic and textual information for improved cross-modal decoding. CLIP-AAC utilizes a pre-trained encoder with separate audio and text heads to extract relevant information, while also employing contrastive learning to bridge domain disparities between audio signals and captions.

Most of this research focuses on certain tasks and uses metric learning to directly address them. Some evaluate the learned space on different tasks to demonstrate their models' ability to extract general features. Their data contain

very specific text captions, such as comma-separated tags or predefined formats, without fully leveraging the expressiveness of natural human language. Some use datasets that contain human annotated captions, requiring huge human effort. Furthermore, the vast majority of them employ the Multi-class N-pair loss (or slight modifications), without exploring different ways to semantically correlate the two modalities. They also limit their approaches to losses that use only one positive example within each batch and lack any sort of supervision signal, something that may lead to suboptimal performances [48].

Based on the above, in our work we contribute in the following ways:

- Explore different ways to perform multimodal pretraining while evaluating them on an unimodal downstream task.
- Use a LLM to create multiple pseudo-captions from unstructured data that are semantically correlated to the downstream task.
- Add variability, reduce human effort, and leverage the expressiveness of human natural language.
- Experiment with different pretrained text encoders and highlight how they instill their modality-oriented knowledge.
- Demonstrate how the incorporation of multiple positive examples during pretraining can enhance the model's performance.
- Illustrate how the performance gap is affected when finetuning on different numbers of samples.

4 Proposed Method

In this section, we present an overview of our proposed method. We begin by clarifying the process of data collection and the utilization of the LLM to generate pseudo-captions. Subsequently, we overview the architecture of our model and its underlying objectives. Lastly, we provide a thorough presentation

encompassing the various pretraining objectives employed in our pretraining procedure.

4.1 Task overview

In our work, we aim to simulate the real-world scenario of operating within constraints similar to those encountered in practice: a restricted pool of labeled samples alongside access to a larger repository of unstructured, unlabeled data. To optimize the downstream task we pretrain on the unlabeled data and then finetune on the labeled dataset. We illustrate how incorporating knowledge derived from a different modality in a pretraining step could enhance the performance of a single-modal model operating in a low resource scenario. We then conduct research on various methods for performing this pretraining, evaluating them by monitoring their impact on the performance of the downstream task.

Given any annotated downstream task related to music, one can tap into the abundance of unlabeled music data available on the web and collect audio and their corresponding lyrics. If lyrics contain information that is semantically related to the downstream task, then a model could perform a pretraining step and learn to extract features that are aligned to the semantics of the lyrics. However, the lyrics alone wouldn't be enough for a model to accurately capture these nuanced semantics, given the vast and multifaceted nature of natural language within the realm of music art, even in big amounts of data. In addition, modern state-of-the-art NLP models have a limit in their input size and cannot ingest and encode all the lyrics of a song. In this work, we utilize Open AI's Chat GPT turbo 3.5 model [49] to extract text features that we believe to be correlated with our task. We asked the model to generate captions that comment on the sentiment and the theme of the lyrics. We will use these pairs to train an Audio Encoder capable of distinguishing audio features based on the available audio

signal, as well as on features originating from lyrics that are not available and have been implicitly derived from the pretraing process.

After acquiring more unlabeled data, and creating those pseudo-captions, we experimented with various ways to pretrain our network. Our model comprises two modality-oriented networks which we describe in detail later. One encodes the audio signal and the other the text signal. Since we used pretrained language models as Text Encoders, we explored which of them achieves the better performance. Then, we kept this Text Encoder fixed and continued our exploration on different ways to pretrain. We experiment with learning coordinated spaces between the two modalities, with a modality matching task and we highlight how incorporating many positive pairs in the loss could enhance the model's performance.

4.2 Data

We essentially have two different datasets. One that is used for the downstream task and one for the pretraining. The downstream task's dataset is a set of pairs of music audios and annotations about their labels. The pretraining task's dataset is a set of randomly selected music audios along with their lyrics. We started by just downloading a bunch of mp3 songs. We then divided this set into two disjoint sets and retrieved the lyrics for the one (pretraining dataset) and the genre label for the other (genre classification).

To create both datasets, we scrapped the YouTube platform using the yt-dlp [50] library and extracted the mp3 files from the songs from various playlists. We used playlists that could contain any genre and we didn't have a particular set of genres in our mind at the time. We downloaded also the songs listed in this dataset [51], which contains popular songs for a very wide range of music genres and a random subset of [52]. At this stage, we had nearly 60,000 MP3 files, accompanied only by their titles. Subsequently, we aimed to exclude

certain files for the genre classification task and utilize the remainder for pretraining. We used the Spotify API [53] and retrieved the genres of our downloaded songs. After some data exploration, we found out that genre classification task isn't as simple as we initially thought. Distinguishing between various music genres can often prove to be a challenging task, particularly when certain genres share striking similarities. For instance, distinguishing between hard rock and punk rock can be tricky. Hard rock features heavy guitar riffs and powerful vocals, while punk rock is known for its fast tempos and rebellious attitude. However, a song that combines hard rock's intensity with punk rock's speed blurs the lines, making it hard to classify as just one genre. Even for human experts, discerning the subtle nuances that define these genres can be far from trivial, highlighting the complexity inherent in genre classification tasks within the realm of music. This claim is verified by the fact that the API returned many labels for each song (≈ 1000 unique labels across all the dataset), some of them combining tags of different genres. For example, the following table illustrates that differentiating between alternative metal, alternative rock, hip hop, and rap metal can be particularly challenging due to their shared elements and thematic complexities. Similarly, identifying the nuances between rock, garage rock, power pop, and punk blues can be equally daunting, highlighting the intricate nature of genre classification.

Artist Name	Song Title	Labels
Rage Against the Machine	Kick Out the Jams	alternative **metal**, alternative **rock**, conscious **hip hop**, funk **metal**, hard rock, nu metal, political hip hop, post-grunge, rap metal, **rap** rock, rock
The Warning	Z	mexican **metal**, monterrey **indie**
B.B. King	You Shook Me	**blues**, blues **rock**, classic rock, electric blues, **jazz** blues, soul blues, traditional blues

Table 1 - Multiple Genre Labels

To deliberately challenge the downstream genre classification task, we identified the most intricately similar genres. We measured the cooccurrence frequencies of pairs of genres and sorted these tuples in descending order. After examining the top occurring pairs, we see that all of them are *specifications* (e.g. *country* **rock**, *alternative* **metal**, *Canadian* **indie**) of 7 **main** genres, namely of: **country**, **hip hop**, **indie**, **metal**, **pop**, **punk**, **rock**. We consider them to be either the most *indistinguishable* music genres or the music genre from which most songs tend to *combine acoustic elements*. We then exclude the songs that contain other genres (e.g. blues, electronic, reggae, r&b) and perform a final cleaning step. This step involved mapping all specifications to main genres by counting the occurrences of each main genre in these specifications and taking the majority. For instance:

Artist Name	Song Title	Labels	Assigned Label
Arctic Monkeys	505	garage **rock**, modern **rock**, permanent wave, **rock**, sheffield **indie**	Rock
Black Sabbath	Wicked World	album **rock**, alternative **metal**, birmingham **metal**, classic **rock**, hard **rock**, **metal**, **rock**, stoner **rock**, uk doom **metal**	Rock
50 Cent	Crazy	east coast **hip hop**, gangster rap, **hip hop**, pop rap, queens **hip hop**, rap	Hip hop
David Luning	Driftin	**country**	**country**

Table 2 - Main Genre Mapping

This preprocessing and filtering resulted in 5038 audio tracks concerning these 7 genres. Their distribution is displayed below:

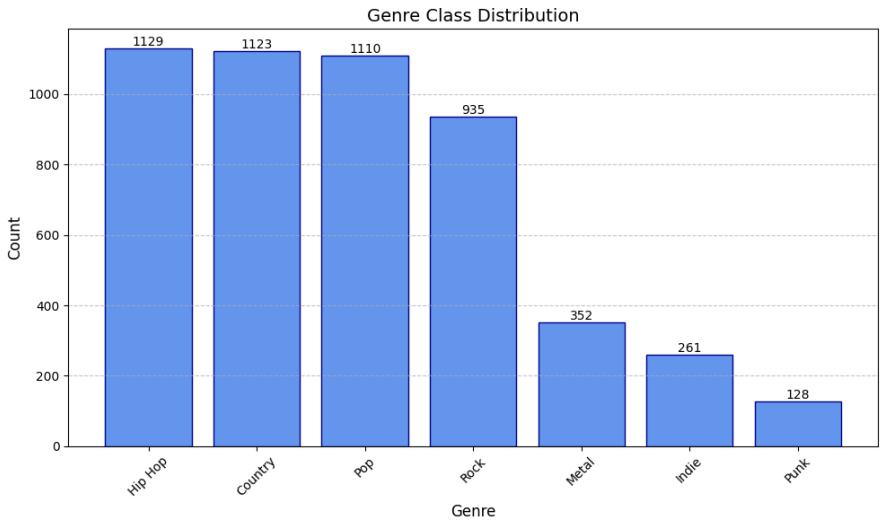

Figure 12 - Genre Label Distribution

From the remained music tracks, we sampled a proportion of them due to resource limitations, and these 34,843 samples will be used for the pretraining step. For each title, we used the LyricsGenius python client [54] and retrieved the song's lyrics. We then asked the ChatGPT LLM to produce the pseudo-captions by using the following prompt:

```
content = f"""Write one sentence summarizing the sentiment of the lyrics of the song below.
            Write a second sentence summarizing the theme of the song.
            And a third one summarizing sentiment and genre.
            Genre is [{genre}].
            The song's lyrics are: " {lyrics} " """
```

Figure 13 - LLM prompt

Note that, as described earlier, our remained data is a mix of manually downloaded songs, along with songs found in [51], where a genre label was available. When available, we include this information in the prompt, otherwise, we provide an empty list as depicted in Figure 13 - LLM prompt. The interesting part is that for many of the songs we didn't provide a genre, the LLM predicted

its genre class (falsely or not) and included it in the response. We didn't have ground truth labels for these instances, so we didn't measure its accuracy, but for some well-known songs we manually checked, it was correct. It is quite possible for the LLM to have seen some of these lyrics during training. Some examples are listed below:

Arctic Monkey's "Do me a favour":

The sentiment of the lyrics in this song is one of sadness and longing, as the speaker reflects on a deteriorating relationship and asks for a favor to alleviate the pain. The theme of the song revolves around the deterioration of a relationship and the emotional struggle that comes with it. The sentiment and genre of the song can be described as melancholic and garage rock with elements of ***permanent wave*** *and modern* ***rock****.*

An example where we included the genre and happened to encounter this genre also in the downstream task.

The Kingston Trio's ''Little Maggie":

The sentiment of the lyrics in this song is melancholic and bittersweet, conveying a sense of longing and heartbreak. The theme of the song revolves around a woman named Maggie who drinks to forget her troubles and engages in deceitful behavior in her relationships. This folk song blends elements of American ***folk*** *revival and traditional folk genres, evoking a nostalgic and authentic sound.*

This is an example, where we included the available genre, which was "folk" and is not included in the genre classification downstream task.

John Hartford's "Up on the hill where they do the boogie":

The lyrics convey a sense of discontent and frustration with the city and its inequality, with a focus on the contrast between the wealthy elite "up on the hill" and the struggles of those in the city.

That is an example where there is no genre information included.

4.3 Model Architecture

Our pretraining model comprises two modality specific encoders. Each of them has two extra linear layers (of dimensionality 1024 and 512) with a nonlinear activation function (tanh) between them. The outputs of the two components are used to directly optimize the different criteria we experiment with.

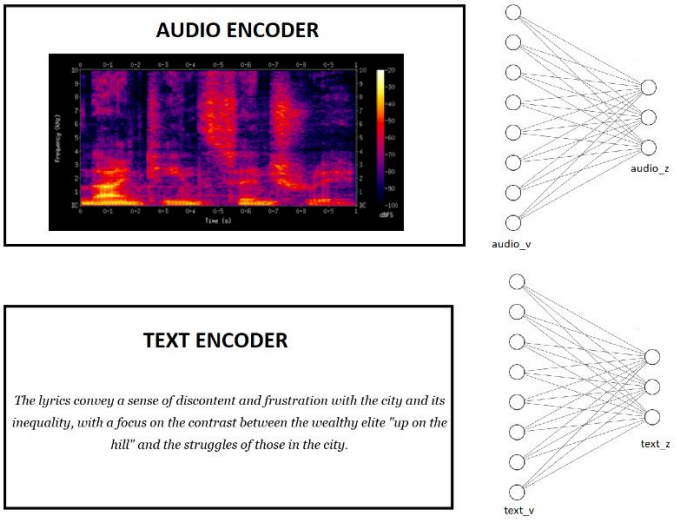

Figure 14 - Pretraining Model Architecture

As for the audio encoder, we implemented a simple CNN that takes as input the Mel spectrograms of the music songs. The parameters used in generating these Mel spectrograms are crucial for shaping the representation fed into the CNN. We used the following parameters:

```
NUM_SAMPLES = 4_320_000
SAMPLE_RATE = 16000
N_FFT = 2048
N_MELS = 64
HOP_LENGTH = 1024
```

Figure 15 - Mel Spectrogram Parameters

NUM_SAMPLES: This parameter defines the number of audio samples to be considered for processing. It's set to 4,320,000 samples (equivalent to 4.5 minutes of audio) given the following sample rate value. This value determines the duration of the audio segment to be analyzed.

SAMPLE_RATE: This parameter specifies the number of samples per second (in Hertz) taken from the audio signal. It is set to 16,000 Hz, indicating that 16,000 samples are captured every second. This value influences the resolution and fidelity of the resulting spectrogram.

N_FFT: The number of data points used in each discrete Fourier transform (DFT). It defines the size of the window function applied to the signal before the Fourier transformation. Here, it's set to 2048, meaning each FFT computation considers 2048 samples at a time. Higher values can provide better frequency resolution but require more computational resources.

N_MELS: This parameter determines the number of Mel frequency bands to be used in the Mel spectrogram computation.

HOP_LENGTH: It defines the number of samples between successive frames in the spectrogram. A smaller hop length results in more frames and potentially higher time resolution but requires more computational resources. Here, it's set to 1024, meaning each frame in the spectrogram is separated by 1024 samples.

The above transformation converted each audio music signal into a tensor of shape (64,4219). This tensor is given as input to the following Audio Encoder component:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 66, 4221]             160
              ReLU-2         [-1, 16, 66, 4221]               0
         MaxPool2d-3         [-1, 16, 33, 2110]               0
            Conv2d-4         [-1, 32, 18, 1056]           4,640
              ReLU-5         [-1, 32, 18, 1056]               0
         MaxPool2d-6          [-1, 32, 9, 528]               0
            Conv2d-7          [-1, 64, 5, 264]          51,264
       BatchNorm2d-8          [-1, 64, 5, 264]             128
              ReLU-9          [-1, 64, 5, 264]               0
        MaxPool2d-10          [-1, 64, 2, 132]               0
           Conv2d-11         [-1, 128, 2, 132]         204,928
      BatchNorm2d-12         [-1, 128, 2, 132]             256
             ReLU-13         [-1, 128, 2, 132]               0
        MaxPool2d-14          [-1, 128, 1, 66]               0
          Flatten-15                [-1, 8448]               0
================================================================
Total params: 261,376
Trainable params: 261,376
```

Figure 16 - Audio Encoder Architecture

As regards the Text Encoder, we experimented with three different Transformer Encoder models described in 2.2 (more details in 4.4.1).

During pretraining, the representation layer (v space) acts as a feature extractor, tasked with learning meaningful representations of the input data. This layer aims to capture relevant patterns and features that can be utilized for downstream tasks. On the other hand, the final layer (z space) directly optimizes the training criterion. It serves as the endpoint for the learning process, refining the representations learned by the preceding layers to enhance their discriminative power and utility for the intended applications. Thus, after pretraining our model, we keep the audio encoder part plus the audio_v layer, using them as a feature extractor module and append a classification head on top of it. The classification head consists of two linear layers of dimensions of 512 and 7 (number of available classes) with a ReLU applied between them. The output logits are used to optimize the cross-entropy loss for the downstream classification task.

4.4 Pretraining Methods

Considering our pretraining and finetuning pipeline, our objective is to assess various pretraining methods and their effects on optimizing the performance of the downstream task (macro F1 score). We initiate this process by establishing the baseline performance, which represents the performance of the audio encoder without any pretraining. This serves as a reference point for evaluating the efficacy of different pretraining approaches. Furthermore, each experiment is conducted using varying amounts of training data. This approach allows us to observe the performance improvement across a spectrum, ranging from scenarios resembling few-shot learning, to situations with a sufficient quantity of training samples. This spectrum enables us to comprehensively analyze the impact of training data volume on performance enhancement. Finally, it is crucial to note that the evaluation for every pair of pretraining methods and the

number of training samples is conducted on the same test set. We performed a stratified split and held out a test set of size 1038 that we used to evaluate every single model.

The un-pretrained supervised music genre classification model achieved the following macro F1 scores:

Num of training samples/macro F1	48	160	400	800	1600	3200
Audio Encoder	0.08	0.21	0.27	0.26	0.32	0.35

Table 3 - Baseline Audio Encoder Performance

4.4.1 Exploration of Various Text Encoders

Our initial focus in optimizing our model centered around identifying a suitable Text Encoder. We employed pretrained language models for this purpose. These models have undergone training on extensive corpora, enabling them to grasp the overall meaning conveyed in input captions. Since our Audio Encoder is untrained, these models serve as a form of knowledge signal, aiding in the processing of textual input. Although the three different Text Encoders we experimented with share the same Transformer Encoder architecture (with slight changes), they differ in terms of the data and objectives they utilized during their training. Thus, they perform a different semantic encoding of our text captions.

The initial criterion these models optimized was inspired by NTXent Loss (2.3):

$$\mathcal{L}\left(a, t^{+}, \{t_i^{-}\}_{i=1}^{N-1}\right) == -log\frac{\exp(A(a)^{\top}T(t^{+})/\tau)}{\exp(A(a)^{\top}T(t^{+})/\tau) + \sum_{i=1}^{N-1}\exp(A(a)^{\top}T(t_i^{-})/\tau)}$$

Where A and T are the Audio and Text encoders, a is an audio signal and t a text caption. We denote with t^{+} the correct caption pair for a given audio and with t_i^{-} a random caption from the current batch of size N. τ is a temperature parameter and is used to ensure more stable training.

We started by using the generic BERT base uncased model. We then switched to RoBERTa [55], which was trained on a larger corpus of text data compared to BERT. We also used a Sentence Transformer Encoder [56], a model that uses Siamese networks to learn to produce meaningful embeddings for whole text sequences. Finally, a simple trick we used and managed to improve slightly the performance was to unfreeze the Audio Encoder during the finetuning in the 10th epoch and reduce its learning rate.

Num of training samples/macro F1	48	160	400	800	1600	3200
BERT	0.23	0.26	0.3	0.28	0.28	0.36
Sent Transf	0.04	0.28	0.38	0.37	0.40	**0.43**
RoBERTa	0.27	0.37	0.37	0.38	0.39	0.40
RoBERTa & Unfreezed AE	**0.35**	**0.41**	**0.44**	**0.45**	**0.43**	0.41

Table 4 - Downstream Performance After Pretraining for Various TEs

Pretraining with RoBERTa as a Text Encoder achieved mostly the best performance and thus, we kept it fixed in the following experiments.

4.4.2 Multiple Positive Pairs

Up until now, we operated in the classical contrastive loss environment, where for each sample we had one positive pair and multiple negatives. As discussed in 2.4, the incorporation of multiple negatives in the loss can lead to more robust and discriminative models by leveraging the diverse patterns between positive and negative samples. That could be also the case in the positive class case. By incorporating multiple positives, the model would effectively capture more variations and nuances in the data, leading to more discriminative features. This is particularly beneficial in complex datasets where the relationship between data points can be intricate, thus enabling the model to form a more comprehensive understanding of the underlying data distribution.

In our case, we could create as many positive samples as we wanted by using the LLM. However, we decided to use the already available text by splitting each caption into the three sentences we specified in the prompt (3.2). The Text Encoder component will produce three different embeddings for these sentences, not just one as before. Each embedding provides the model with a distinct semantic encoding of the lyrics. The first embedding concerns sentiment, the second focuses on the theme, and the third is a more abstract one that also includes the genre (if available or predicted). These three embeddings constitute three different positive text pairs to a single audio signal.

Using a single output embedding to encode all the available information aggregates different dimensions of the meaning of the captions. We believe that, when data is quite limited, as in our case, the resulting representation may not fully capture these meanings. On the other hand, these 3 semantically different captions could let the Text Encoder generate 3 rather different embeddings each

of which encodes a different aspect of the underlying semantics. These vectors are placed in different regions in the learned space. The criteria used try to minimize the distance of the audio signal from these 3 vectors, something that could result in more robust representations.

The most straightforward way to do so is to just split the one-to-three tuples of audio-captions into three independent audio-caption pairs (abbreviated IS, which stands for 'Independent Samples').

Audio signal	Sentence 1. Sentence 2. Sentence 3.

Becomes:

Audio signal	Sentence 1.
Audio signal	Sentence 2.
Audio signal	Sentence 3.

The second way we experimented with, was to directly incorporate the multiple positive samples in the loss function (abbreviated SCL, which stands for "Supervised Contrastive Loss"). In [48] supervised contrastive loss, they leverage some signal of supervision, like an available class label and they perform positive and negative sampling based on this label. Then, they compute the NTXent loss for each positive pair and then just sum up. We use this loss and treat the origin of each sentence as a class label, so that 3 sentences

originating from the same audio signal to have the same label. Thus, the multiple positive contrastive loss becomes:

$$L_{pos} = \sum_{i \in I} \sum_{p \in P(i)} \left[-\log \frac{\exp\left(A(a_i)^T \cdot T(t_p)/\tau\right)}{\sum_{j \in I} \exp\left(A(a_i)^T \cdot T(t_j)/\tau\right)} \right]$$

Where I is the set of all indices in the batch, a_i is the audio embedding of index i , t_p $(or\ t_j)$ is the text embedding of index p (or j). $P(i)$ is the set of indices of positive examples of index i, with $|P(i)| = 3, \forall i$.

In all NTXent-inspired losses we have experimented with so far, we include multiple negatives in the denominator. However, that approach has a significant flaw. It is very probable, in the current batch, to encounter different audios with similar sentiment or thematic meanings. In such cases, NTXent loss does not differentiate between these semantically similar samples, treating them all as negatives relative to each other. This can lead to a scenario where the model fails to place semantically similar audio and text closely together in the embedding space, as they inadvertently contribute to the repelling forces in the loss calculation.

To address this issue, we experimented with triplet loss (abbreviated STL, which stands for Stochastic Triplet Loss). Triplet loss allows us to focus on one positive and one negative sample at a time. This approach reduces the likelihood of erroneously treating an audio-caption pair with similar meanings as negative, as we only sample one positive and one negative instance during each iteration. However, to introduce diversity and ensure robust learning, we incorporate multiple positive text pairs by uniformly sampling one positive from the three available options each time. By doing so, we aim to maintain a balance between providing sufficient positive examples for effective learning and avoiding the

weakening of semantically meaningful relationships by overly emphasizing negative instances. The table below summarizes these multiple positive models:

Num of training samples/macro F1	48	160	400	800	1600	3200
IS	0.25	**0.42**	**0.4**	**0.45**	0.45	0.45
SCL	**0.33**	0.32	0.37	0.44	**0.46**	**0.52**
STL	0.15	0.25	0.27	0.29	0.33	0.36

Table 5 - Multiple Positive Models Metrics

4.4.3 Stochastic Modality Matching

To fully disengage from this comparative setting and mitigate the effects of negative sampling, we conducted one final experiment (abbreviated SMM, which stands for 'Semantic Matching Model') that doesn't involve learning a metric space. We employed a classification task that considers both modalities and aims to establish a correspondence between them by utilizing multiple positive samples. We pretrained a model on pairs of audio signals and text, performing binary classification to predict whether these inputs constitute a true pair. Specifically, for each audio input, we randomly decide with a probability of 0.5 whether to select a true text pair or a random one. If we choose a true positive text pair, we uniformly sample one of the available captions. This model follows the architecture depicted in Figure 14 - Pretraining Model Architecture, where we concatenate the z-space embeddings and append a final classification layer with one output. We employ Binary Cross Entropy Loss as our loss function:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

Where $y = \hat{y}$ when the model got as input a correct audio-text pair. The results are depicted below:

Num of training samples/macro F1	48	160	400	800	1600	3200
SMM	0.31	0.37	0.4	0.4	0.44	0.44

4.4.4 Comparison

Num of training samples/macro F1	48	160	400	800	1600	3200
Audio Encoder	0.08	0.21	0.27	0.26	0.32	0.35
1-Text(*)	**0.35**	0.41	**0.44**	**0.45**	0.43	0.41
IS	0.25	**0.42**	0.4	**0.45**	0.45	0.45
SCL	0.33	0.32	0.37	0.44	**0.46**	**0.52**
STL	0.15	0.25	0.27	0.29	0.33	0.36
SMM	0.31	0.37	0.4	0.4	0.44	0.44

Table 6 - Main Models Comparison

(*) Mentioned as RoBERTa & Unfreezed AE in Exploration of Various Text Encoders.

Figure 17 - Models Comparison Across Number of Training Data (log scale)

All the models have significantly enhanced downstream task performance and show considerable variance when compared to each other. Figure 17 - Models Comparison Across Number of Training Data allows us to examine the performance curve from the few-shot learning scenario up to the case with an adequate amount of data (in a logarithmic scale).

First and foremost, we emphasize that there is no specific model that consistently outperforms the others across various amounts of training data. In the few-shot case, the best performing models was the 1-Text. In the full training dataset, the best performing one was SCL. In the intermediate cases, we see the IS consistently outperforming the SMM and SCL. In the last two pools of available data (1600, 3200) we see that the performance ordering remains unchanged even the amount of data doubles and we strongly believe

that was the final one as the data increases. On the contrary, we observe a fluctuation in the performances in the intermediate cases.

From the figure we observe that all the models that incorporate multiple negatives in their loss significantly outperformed STL. STL was the worst performing model and was consistently close to the unpretrained Audio Encoder. It seems that contrasting against only one negative example, even when providing multiple positives across the iterations, does not help the model to learn a sufficient distinguishable space.

Regarding the multiple positive models, they outperform their single-text counterparts as the available data increases. This suggests that a single encoding of the three semantic dimensions (sentiment, theme, mix, and genre, if available) provided the model with sufficient information during pretraining. This encoding is a fusion of the semantics of these three rather independent sentences. Taken individually, these sentences aren't always correlated to the downstream task we evaluate our models on. For example, one song's theme may not be representative of its genre category. When using multiple positives, the text encoder produces three quite different embeddings that are treated equally and try to be placed closely to the audio embeddings. This, combined with the fact that different audios could share one of the three dimensions (for example to have similar themes) may guide the model to push the embeddings of different things close. We believe that deriving a single text representation could address such cases by mixing relevant with non-relevant information. However, as we get more data, these models achieved the best performance. During finetuning, the Audio Encoder may identify regions of entangled data points and learn a non-linear mapping that classifies them effectively, while also leveraging a large proportion of the space where the three text signals led to distinguishable regions. Something difficult when limited data are available. Nevertheless, these are intuitive explanations and further examination of the learned space is required and we leave it as a future work.

Below the distribution of different Macro F1 scores is displayed using boxplots. This facilitates us to assess the robustness of our models to different amounts of data.

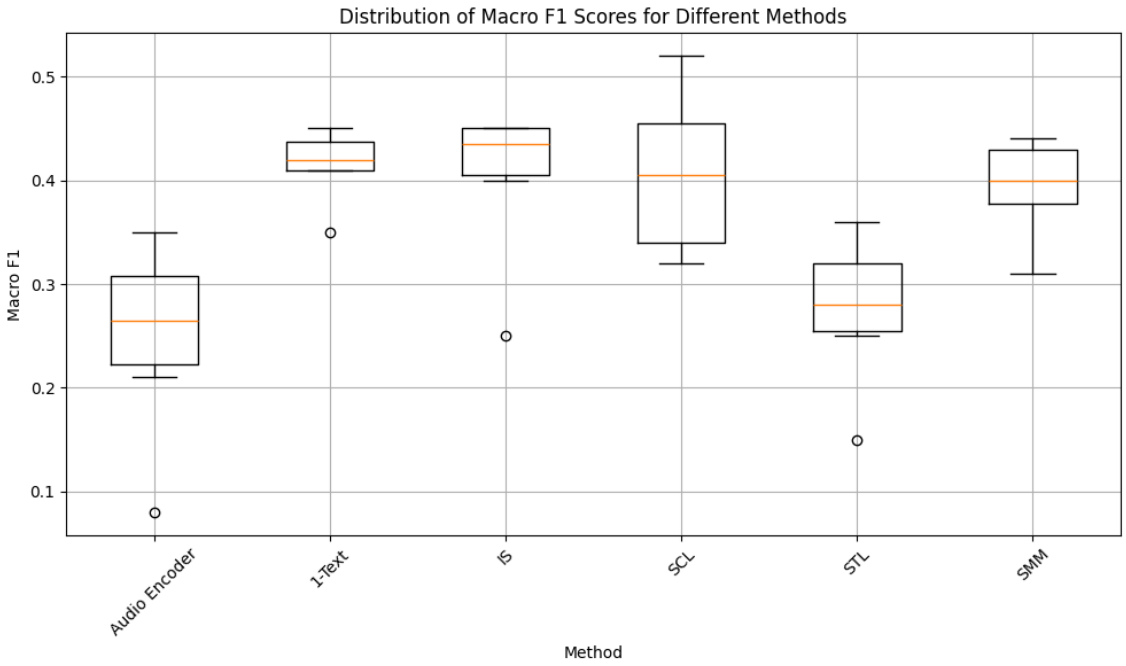

Figure 18 - Performances Distribution

The boxes with relatively small widths correspond to models that did not show improved performance as the training data increased. The median performance is indicated by a straight orange horizontal line, while a small circle denotes a performance outlier. The models 1-Text, IS, and SMM remained mostly unaffected when acquiring more data. However, their high median values indicate that they are good choices when the downstream task's data is limited. Conversely, the width and upper limit of SCL's box highlight this model as a very good option when sufficient downstream task data is available, as it seems to scale its performance accordingly.

5 Conclusions

In this thesis, we demonstrated that learning a coordinated space during a pretraining step between different modalities can significantly enhance the performance of unimodal models, especially in scenarios with limited annotated data. By utilizing a LLM to generate pseudo-captions from a large pool of non-annotated audio data, we were able to pretrain a model in an unsupervised manner before fine-tuning it on a low-resource annotated dataset. This approach highlighted the substantial performance improvements in few-shot learning settings, with a notable performance gap when comparing low-resource scenarios. As more data is used for training, this performance gap tends to decrease for most of our models, indicating the ability of supervised models to scale proportionally to the available resources and the need for more careful finetuning of the pretrained models. Additionally, the Text Encoder, pretrained to capture the semantics of captions, acted as an effective knowledge transfer mechanism during the pretraining procedure, suggesting that a Text Encoder trained on similar tasks (lyrics sentiment analysis, genre prediction etc.) could further enhance performance.

Moreover, we found that incorporating multiple positives into the loss function led to the highest performance on the full training dataset, illustrating the importance of this technique in multimodal learning. Stochastic Modality Matching showed promising results compared to Contrastive Learning, as it avoids pushing away samples with similar semantics that could act as negatives in the contrastive setting. Conversely, single-positive models performed better when tested on smaller amounts of data, suggesting their suitability for low-resource scenarios.

5.1 Future Work

In this section, we highlight some limitations of our methodology and propose questions for future research based on our experimentation. Firstly, the prompt we used for the Large Language Model (LLM) resulted in text descriptions that were too closely tied to the downstream task. We could improve this by using more general queries, such as “Please comment on the given lyrics in 3-4 sentences”. These broader captions could then be used to pretrain any downstream task, regardless of our intuition about their relation to sentiment or theme. Nonetheless, both our current model and any such alternative model should be evaluated across a variety of downstream tasks to assess their ability to incorporate general semantics. This would enable the development of a versatile audio embedder that can be fine-tuned for any specific task. Following that, we should perform an evaluation on the Text Encoder also, or even a multimodal variant with pretrained Audio and Text encoders, and check whether these components also leverage this pretraining step.

Regarding the contrastive losses and the variants we used, the most significant components were the positive and negative samples. We demonstrated that incorporating multiple positive pairs could benefit this method. However, we also noted that during negative sampling, text pairs that may be semantically correlated to the audio anchor, and even match their genre labels, are treated as negatives and are pushed apart. These observations suggest the need to explore methods for more efficient positive or negative sampling or even new forms of loss functions.

Lastly, we observed that the pretrained models didn’t scale much when acquiring more data for the downstream task. We should not only experiment more with the architecture and hyperparameters of our models but also perform an exploration of the learned space. For instance, an application that further lets us evaluate the learned space is cross-modal retrieval. Learning a coordinated

space using similarity measures enables us to perform modality oriented queries and use the same similarity metrics to retrieve samples from the other modality.

6 References

[1] Ketan Doshi, Towards Data Science, Audio Deep Learning Made Simple (Part 1): State-of-the-Art Techniques, Feb 11, 2021

[2] Pohle, T., Pampalk, E., & Widmer, G. (2005). Evaluation of frequently used audio features for classification of music into perceptual categories. In Proceedings of the Fourth International Workshop on Content-Based Multimedia Indexing (Vol. 162). Citeseer.

[3] Giannakopoulos, T. (2015). pyAudioAnalysis: An open-source Python library for audio signal analysis. PLoS One, 10(12), e0144610.

[4] Sharma, G., Umapathy, K., & Krishnan, S. (2020). Trends in audio signal feature extraction methods. Applied Acoustics, 158, 107020.

[5] Cadence PCB solutions, https://resources.pcb.cadence.com/blog/2020-time-domain-analysis-vs-frequency-domain-analysis-a-guide-and-comparison.

[6] Mastriani, M. (2018). Quantum-classical algorithm for an instantaneous spectral analysis of signals: a complement to Fourier Theory.

[7] https://en.wikipedia.org/wiki/Fourier_transform

[8] https://en.wikipedia.org/wiki/Spectrogram

[9] https://en.wikipedia.org/wiki/Mel_scale

[10] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S. -Y. Chang and T. Sainath, "Deep Learning for Audio Signal Processing," in IEEE Journal of Selected Topics in Signal Processing, vol. 13, no. 2, pp. 206-219, May 2019, doi: 10.1109/JSTSP.2019.2908700.

[11] Zellig S Harris. Distributional structure. Word, 10(2-3):146–162, 1954.

[12] Claude E Shannon. A mathematical theory of communication. Bell system technical journal, 27(3):379–423, 1948.

[13] Shraddha Anala. A Guide to Word Embedding. Towards Data Science. 26 October 2020. https://towardsdatascience.com/a-guide-to-word-embeddings-8a23817ab60f

[14] T Mikolov and J Dean. Distributed representations of words and phrases and their composi-tionality. In Proceedings of NeurIPS, 2013.

[15] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Proceedings of EMNLP, 2014.

[16]https://machinelearninginterview.com/topics/natural-language-processing/what-is-the-difference-between-word2vec-and-glove/

[17] Liu, Q., Kusner, M. J., & Blunsom, P. (2020). A Survey on Contextual Embeddings. arXiv preprint arXiv:2003.07278.

[18] Zhiyuan Liu, Yankai Lin, Maosong Sun. Representation Learning for Natural Language Processing. Springer, 2020.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Llion Jones, Jakob Uszkoreit, Aidan N Gomez, and Lukasz Kaiser. Attention is all you need. In Proceedings of NeurIPS, 2017.

[20] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[21] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. CoRR, abs/1810.04805. Retrieved from http://arxiv.org/abs/1810.04805

[22] Rumelhart, David E; Hinton, Geoffrey E, and Williams, Ronald J (Sept. 1985). Learning internal representations by error propagation. Tech. rep. ICS 8504. San Diego, California: Institute for Cognitive Science, University of California.

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

[24] Guo, W., Wang, J., & Wang, S. (2019). Deep multimodal representation learning: A survey. Ieee Access, 7, 63373-63394.

[25] G. E. Hinton, S. Osindero, and Y.-W. Teh, ``A fast learning algorithm for deep belief nets," Neural Comput., vol. 18, no. 7, pp. 1527-1554, 2006.

[26] R. Salakhutdinov and G. Hinton, ``Deep Boltzmann machines," in Proc. 29th Int. Conf. Artif. Intell. Statist., 2009, pp. 448-455.

[27] Y. Kim, H. Lee, and E. M. Provost, ``Deep learning for robust feature generation in audiovisual emotion recognition," in Proc. IEEE Int. Conf. Acoust., Speech Signal Process., May 2013, pp. 3687-3691

[28] L. Pang and C.-W. Ngo, ``Mutlimodal learning with deep Boltzmann machine for emotion prediction in user generated videos," in Proc. 5th ACM Int. Conf. Multimedia Retr., 2015, pp. 619-622.

[29] G. E. Hinton and R. S. Zemel, ``Autoencoders, minimum description length and Helmholtz free energy," in Proc. Adv. Neural Inf. Process. Syst., 1994, pp. 3-10.

[30] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, ``Multimodal deep learning," in Proc. 28th Int. Conf. Mach. Learn., 2011, pp. 689-696.

[31] C. Silberer and M. Lapata, “Learning grounded meaning representations with autoencoders,” in Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics, vol. 1, 2014, pp. 721-732.

[32] D. Wang, P. Cui, M. Ou, and W. Zhu, ``Deep multimodal hashing with orthogonal regularization," in Proc. 24th Int. Conf. Artif. Intell., 2015, pp. 2291-2297.

[33] I. J. Goodfellow et al., ``Generative adversarial nets," in Proc. 27th Int.

Conf. Neural Inf. Process. Syst. (NIPS), vol. 2. Cambridge, MA, USA:
MIT Press, 2014, pp. 2672-2680.

[34] H. Zhang et al., ``StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks,'' in Proc. IEEE Int. Conf. Comput. Vis., Jun. 2017, pp. 5907-5915.

[35] X. Liang, Z. Hu, H. Zhang, C. Gan, and E. P. Xing, ``Recurrent topic-transition GAN for visual paragraph generation,'' in Proc. IEEE Int. Conf. Comput. Vis., Jun. 2017, pp. 3362-3371.

[36] R. Socher, Q.V. L. A. Karpathy, C. D. Manning, and A.Y. Ng, ``Grounded compositional semantics for nding and describing images with sentences,'' Trans. Assoc. Comput. Linguistics, vol. 2, no. 1, pp. 207-218,
2014.

[37] Kaya, M., & Bilge, H. Ş. (2019). Deep metric learning: A survey. Symmetry, 11(9), 1066.

[38] 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).

[39] Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. Advances in neural information processing systems, 29.

[40] Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 11336–11344, 2020.

[41] Wu, H. H., Seetharaman, P., Kumar, K., & Bello, J. P. (2022, May). Wav2clip: Learning robust audio representations from clip. In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 4563-4567). IEEE.

[42] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... & Sutskever, I. (2021, July). Learning transferable visual models from natural language supervision. In International conference on machine learning (pp. 8748-8763). PMLR.

[43] Stewart, S., Avramidis, K., Feng, T., & Narayanan, S. (2024, April). Emotion-Aligned Contrastive Learning Between Images and Music. In ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 8135-8139). IEEE.

[44] Wu, H. H., Seetharaman, P., Kumar, K., & Bello, J. P. (2022, May). Wav2clip: Learning robust audio representations from clip. In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 4563-4567). IEEE.

[45] Chen, T., Xie, Y., Zhang, S., Huang, S., Zhou, H., & Li, J. (2022, May). Learning music sequence representation from text supervision. In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 4583-4587). IEEE.

[46] Doh, S., Won, M., Choi, K., & Nam, J. (2023, June). Toward universal text-to-music retrieval. In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1-5). IEEE.

[47] Chen, C., Hou, N., Hu, Y., Zou, H., Qi, X., & Chng, E. S. (2022). Interactive audio-text representation for automated audio captioning with contrastive learning. arXiv preprint arXiv:2203.15526.

[48] Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., ... & Krishnan, D. (2020). Supervised contrastive learning. Advances in neural information processing systems, 33, 18661-18673.

[49] https://platform.openai.com/docs/models/gpt-3-5-turbo

[50] https://pypi.org/project/yt-dlp/

[51] https://www.kaggle.com/datasets/mervedin/genius-lyrics

[52]https://www.kaggle.com/datasets/carlosgdcj/genius-song-lyrics-with-language-information

[53] https://developer.spotify.com/documentation/web-api

[54] https://lyricsgenius.readthedocs.io/en/master/index.html

[55] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

[56] Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084.