Master in Sound and Music Computing

Universitat Pompeu Fabra

Beyond Benchmarks: A Toolkit for Music Audio Representation Evaluation

Christos Plachouras

Supervisor: Dmitry Bogdanov

Co-Supervisor: Pablo Alonso-Jiménez

August 2023

Master in Sound and Music Computing

Universitat Pompeu Fabra

Beyond Benchmarks: A Toolkit for Music Audio Representation Evaluation

Christos Plachouras

Supervisor: Dmitry Bogdanov

Co-Supervisor: Pablo Alonso-Jiménez

August 2023

Universitat Pompeu Fabra
Barcelona

Contents

Acknowledgement

I would like to express my sincere gratitude to:

- my mother, my sister, and my partner for their unconditional support during all of my endeavors;
- Pablo and Dmitry for their genuine interest and support of this project, and our fruitful discussions;
- Marius for the inspiration to delve into the practical aspects of deep learning, and for our eye-opening collaborations; and,
- the people of the MTG for these wonderful, uplifting, and memorable two years.

Abstract

Numerous cutting-edge approaches employed in Music Information Retrieval (MIR) tasks are now leveraging representation learning. This technique entails learning meaningful representations of the desired data through a source task, which can act as compact, efficient inputs to separate downstream tasks. With the growing interest in developing general audio representations that are useful for multiple tasks, the need for thorough, consistent, and fair evaluation is more pertinent than ever. However, evaluation efforts so far are often fragmented, owing to differences in data availability and computational resources, missing implementation details, or lack of agreed-upon design choices. Public benchmarks often opt for a fixed evaluation setup that provides consistency in exchange for a narrower-scoped investigation of MIR systems.

In this master's thesis project, we present a toolkit for reproducible music audio representation evaluation. The toolkit provides an easy and configurable way to run evaluation experiments for MIR systems utilizing representation learning. It provides a variety of MIR datasets and tasks for evaluating performance given different input representations, embedding extraction frequency, downstream models, and audio perturbations. It also includes tools for exploring and visualizing evaluation results under different experimental setups. The toolkit is primarily focused on aiding the development of music audio representations while ensuring every evaluation experiment is transparent and can be faithfully reproduced. We use the toolkit to conduct an extensive evaluation of multiple representations from widely used music embedding models for a variety of MIR tasks, datasets, and deformation scenarios.

Keywords: Music Audio Representation Learning, Embedding Evaluation, Evaluation Toolkit, Representation Robustness, Reproducible Research

Chapter 1

Introduction

This thesis project aims to contribute methods and software for the advancement of reproducible evaluation of music audio representation learning systems. In the following sections of the introduction, the concept of representation learning and its history will be briefly explained (see Sec. 1.1.1), followed by an overview of the current approaches to music audio representation evaluation and their limitations (see Sec. 1.1.2), as well as how they motivate this thesis project (see Sec. 1.1.3). Subsequently, a summary of music audio representation learning techniques will be presented (see Sec. 1.2.1), followed by the current tools and frameworks for evaluating them (see Sec. 1.2.2). Finally, the contributions of this thesis project will be presented (see Sec. 1.3).

1.1 Background

1.1.1 Music Representations

By the early 2000s, statistical and machine learning methods such as Support Vector Machines (SVMs), Decision Trees, k-Nearest Neighbors (kNN), and even some early architectures of small neural networks had started being used for tasks in a variety of computational fields. Similarly, in the field of Music Information Retrieval (MIR), the interdisciplinary domain concerned with extraction, analysis, and organization of information from music data, researchers adapted and extended those promising paradigms to deal with music data, both audio-based, like music recordings, and symbolic, such as music notation. Up to the end of the decade, however, limitations in computational resources, data, and annotations, as well as, often, methodological constraints, prevented the use of raw data representations or even ones that closely represent most aspects of the data with high fidelity. For music audio, particularly, the raw digital waveform and even frequency-based audio representations were too information-dense to be used as inputs to statistical and machine learning systems.

Researchers in MIR, therefore, elected to employ algorithmically derived representations, otherwise denoted as handcrafted features or descriptors, which were often designed or adapted in an attempt to closely describe particular characteristics of the audio that could be relevant to the task at hand. Some especially popular examples of those features are Mel-Frequency Cepstral Coefficients (MFCC) [1, 2], descriptors of the power spectrum of a sound, which were often used as timbre descriptors for instrument recognition [3, 4, 5], and chroma-based features [6], descriptors of spectrum energy per pitch class of a music scale, which were often used for chord detection [7, 8, 9] and music alignment [10, 11]. Combinations of those features were also used, particularly for tasks that required descriptions of various musical aspects of the audio, like genre classification. Feature engineering, the process of creating a set of relevant features for a task, also gained traction in MIR [12, 13, 14], while some software provided configurable or preset feature extractors useful for MIR tasks, like the Essentia [15] music extractor [16] and Wekinator [17]. While these features were less information-dense than the raw data, many were still too high-dimensional and, particularly, too dense temporally to be practically useful. In many cases, their mean or variance over less granular intervals was instead computed and used, leading to some information loss.

Computational power and hardware acceleration advances leading up to the early 2010s coupled with the evolution of model architectures and training methodologies allowed researchers, particularly in the field of computer vision and speech, to start demonstrating the viability of end-to-end workflows [18, 19]. These holistic pipelines, commencing from general data representations and culminating in predictive outcomes, often outperformed shallower networks reliant on handcrafted features, particularly in scenarios where substantial data volumes were available. However, this paradigm, often referred to as end-to-end deep learning due to the large model size required, was certainly not accessible to all researchers due to the cost of the computational resources required to train deep models. Another significant barrier to end-to-end deep learning particularly for MIR was the scarcity of annotated data [20, 21], an issue that is still faced to some degree today partially due to the particularities of music copyright laws. In spite of this, many researchers demonstrated the potential of end-to-end deep learning for MIR on one task in particular: music auto-tagging. The relative availability of tags without a strict vocabulary compared to other types of annotation coupled with work from Creative-Commons-licensed and public domain music sources led to the creation of two large music datasets: MagnaTagATune [22] in 2009 and the Million Song Dataset [23] in 2011. Researchers demonstrated that large models using spectral audio representations [24, 25] or even the waveform [26, 27] could be used to improve auto-tagging performance compared to approaches using handcrafted features and combinations of them with shallower classifiers.

Even with these advancements, it remained impractical to use large models for other MIR tasks that required harder-to-obtain annotations and data. In the early 2010s, however, the paradigm of representation learning started attracting interest in MIR. Representation learning refers to techniques for learning data representations usually from raw or close-to-raw data representations through one or more source tasks. Most often, this is done by utilizing the representation of an intermediate layer, often called an embedding, of the model used for the source task or tasks, rather than the prediction layer. The trained representation model can then be used for inference of representations, which in turn can be used as inputs to shallower models solving separate so-called downstream tasks. Initially, approaches utilizing Deep Belief Networks (DBN) [28, 29, 30], described in more detail in Sec. 1.2.1, attempted to learn representations without labels with mixed but promising results. In the mid-2010s, however, the popularity of DBNs declined in favor of newer architectures that didn't suffer from the same training difficulty and limited scalability of DBNs, which was partially attributable to the vanishing gradient problem [31]. Instead, successful representation learning attempts using auto-tagging on the Million Song Dataset as the source tasks were published [32, 33, 34], reigniting interest in representation learning.

Representation learning promises many theoretical benefits. Among others, by training a single, appropriate large model on appropriate data, one can use the learned representation to solve separate downstream tasks with a lot less computational power. Researchers with limited computational resources often take advantage of this fact by using large, pre-trained models, often from different but neighboring domains such as speech and vision, released by corporations and research centers with leading computing capabilities to extract representations that can improve performance on several, separate MIR downstream tasks [35, 36]. Additionally, again depending on good source model and data selection, one can theoretically learn relevant representations to deal with limited-data scenarios, such as when dealing with a new instrument sound or music genre. Representation learning also provides control over the size and fidelity of representation; one can attempt to learn a high-dimensional representation appropriate for use in a variety of downstream tasks, or they can instead attempt to create a compact representation capturing a specific aspect of the data. The practical benefits of dealing with learned representations have also interested the MIR field. Embeddings can be compact, which makes dealing with and transferring datasets a lot easier compared to the information-dense digital audio recordings. They also provide an avenue for obtaining data without violating copyright law: corporations such as record labels, who often own the rights of music produced by their artists, are, compared to providing audio, less reluctant to provide extracted embedding representations of audio. This is because, apart for the case of embeddings from generative models like autoencoders, embeddings generally cannot be used to recover the original audio.

1.1.2 Music Representation Evaluation

In the current landscape of machine learning, especially within the realm of deep representations, a prevailing trend is the reliance on complex, deep models, whose layer-to-layer transformations are often elusive to decipher in terms comprehensible to humans. The result is that we don't immediately know what information these compact, dense representations contain, or how easy it is to extract. This sets them apart from certain handcrafted features which offer visualizability and intuitive interpretability. It is therefore all the more important to adequately evaluate and, ideally, also attempt to interpret these deep representations to some degree.

In the field of Music Information Retrieval, evaluating deep representations has traditionally implied the following setup: researchers, on a paper-to-paper basis, would construct a set of experiments where their newly developed representations are used as input to models aimed to solve downstream MIR tasks. Tedious effort would be put into implementing experiments on at least a few downstream tasks and datasets, but also to compute results for other relevant representations, potentially both deep and handcrafted ones. Even still, there are often slight or moderate differences across evaluations, such as the exact data splits, the data quality and preprocessing, the representation sampling frequency, the downstream model setup, and many others. Thus, even though significant effort is spent to design each evaluation, results across them are often not directly comparable [37].

The other primary means for deep representation evaluation have been public benchmarks and challenges. The MIR community has shown interest in benchmarks almost since its first official conference-symposium in the year 2000 with the inception of the first Annual Music Information Retrieval Evaluation eXchange (MIREX) just five years later [38, 39]. The main evaluation format of MIREX has been submission-based, with results computed on predetermined tasks and data setups by the organizing committee. The fixed evaluation setup ensured fairness and consistent, directly comparable evaluation between submissions. Ever since, a series of benchmark-challenges have been created, many of which are aimed specifically at representation learning for audio (see Sec. 1.2.2. Their core principle is the same: a submission-based setup with evaluation on a fixed set of tasks and datasets to ensure fairness and comparability.

In reality, however, benchmarks have limitations that often prevent them from being useful indicators of the real-world performance of systems. In the case of representation learning systems in particular, almost every aspect of the downstream evaluation pipeline is often frozen, including, but not limited to, the downstream model setup like its type, number and size of layers, optimizer, and training parameters, and the evaluation metrics used. In practice, a badly performing representation for a specific task might have greatly benefited from an extra layer in the downstream model, or, generally, might have performed differently under different downstream setups. Freezing a particular downstream pipeline provides consistent results for that particular setup, but ignores other optimizations in the complete representation learning pipeline. Moreover, benchmarks provide an exciting but, often, too simplistic view of system performance. In practice, in a deployed audio machine learning system, we might care about its resilience to various audio perturbations, its behavior in unseen data, its computational requirements, its interpretability, and its fairness. For representation learning, for example, one might consider: will a genre classification system based on deep representations perform inadequately if slight background noise was present in the audio, or if the audio was slightly time-stretched, or are there underlying assumptions during the learning of the representation, such as it being pitch-invariant, that might render it unusable on new data or subtasks like Turkish Makam recognition? Since evaluation in benchmarks is usually narrow-scoped, they don't eliminate the need for extensive, individual evaluations, and, coupled with their submission-based framework, aren't great tools for the development process of a deep representation.

1.1.3 Motivation

The aforementioned limitations of benchmarks and the difficulty of building extensive, consistent, and reproducible evaluations for representation learning systems are the factors that motivated the creation of this thesis project. In thinking beyond benchmarks for deep representation evaluation, we arrived at the requirements that follow.

Embedding development

Given the tediousness of building evaluation experiments and the narrow-scoped and delayed evaluation results public benchmarks often compute, we need a local-first, easy-to-set-up, and extensive evaluation tool. This tool would facilitate embedding development by making it easier to extensively evaluate a deep representation at every development iteration.

Embedding understanding

While performance in the target downstream tasks is usually the primary indicator of the representation's suitability, it's not always clear how the downstream task design might be limited compared to scenarios faced in the system's deployment. Having some understanding of the information that is contained in a representation, something that isn't trivial for deep representations, could help us expand our understanding of the representation's suitability for certain tasks. For example, it would be useful to be able to investigate whether the learned representation is the result of learning shortcuts, meaning it might not generalize to out-of-dataset scenarios, whether it's invariant or equivariant to specific audio transformations such as gain adjustment or pitch-shifting, or whether it correlates with a particular characteristic of the data such as the timbre, which could indicate tasks it might be suitable for.

Pipeline evaluation

Different deep representations, given differences in their size, in their generalizability, and in how disentangled or easy to extract relevant information from them is, might perform differently under different downstream pipeline setups. We need control over those setups in order to understand how to better leverage a representation and what performance compromises we might have to make.

Reproducibility

In order to ensure the consistency and comparability of evaluation results for different deep representations, the entire representation learning pipeline needs to be transparent and easily reproducible. We need an easy way to configure as many aspects of the pipeline as possible, while being able to share our configuration and have other researchers reproduce our results without worrying about missing implementation details, data and software versioning differences, and different metric definitions.

Modularity and extendability

While a rigid, consistent evaluation configuration is important for reproducibility, we also need to be able to easily expand various aspects of the evaluation, like, for example, the embedding model and its version used, the downstream classifiers, the tasks, the datasets, and the evaluation metrics. To do this, we need a framework decoupled enough to allow contributions from multiple parties. At the same time, users should be able to easily interface the toolkit with their own datasets and tasks, a use case that is particularly relevant for industry applications.

With these requirements in mind, we set to build a framework for music audio representation evaluation.

1.2 Related work

In this section, we present existing approaches for learning music audio representations for MIR tasks (see Sec. 1.2.1) and frameworks for evaluating them 1.2.2. Representation learning approaches are categorized based on the learning principle and type of data they employ. Given that current representation learning approaches rely on representations from intermediate model layers called embeddings, the terms "deep representation" and "embedding" are used interchangeably in this context.

1.2.1 Deep Audio Representations

In the early days of machine learning, researchers in MIR were concerned with designing handcrafted features, descriptors of music information extracted from the audio signal. After the success of the representation learning paradigm in neighboring fields such as computer vision and speech processing, however, approaches attempting to automatically learn useful representations that could be transferred to other tasks started appearing in MIR.

Unsupervised learning

The first category of representation learning approaches applied to music tasks can be said to be those employing Deep Belief Networks (DBN). DBNs [40] are generative probabilistic models with a greedy layer-wise unsupervised pretraining phase. They were seen as an attractive alternative to Deep Neural Networks (DNN), which were considered difficult to train using gradient descent [41].

In 2009, Lee et al. [28] used Convolutional DBNs (CDBN), a variant of DBNs with convolutional layers, a type of layer where the same weights and biases are applied to groups of contiguous inputs, typically using small kernels. They demonstrate how representation learning can be a powerful alternative to handcrafted features in various audio-related tasks. For music specifically, they decide to train a CDBN on an unlabelled music collection, using a short-window spectrogram that is then PCA-whitened as the input. The representations from the CDBN are then evaluated on music genre and music artist classification by using their $L1$ and $L2$ distances, outperforming experiments using the raw spectrogram and Mel Frequency Cepstral Coefficients (MFFCs), a popular handcrafted feature often used as a timbre descriptor.

Hamel et al. [29] further experiment with DBNs for music domain tasks. They pretrain a DBN in an unsupervised manner with the training data from the GTZAN dataset [42], and then fine-tune the model with the same training data using gradient descent to predict the genre annotations in the dataset. As an input feature, they instead use the Discrete Fourier Transforms (DFT) of 46.44ms audio chunks, and they run an extensive hyper-parameter optimization. Using the activations from different combinations of layers as features fed to a Support Vector Machine (SVM), they achieve state-of-the-art performance in genre classification and music auto-tagging.

Lastly, Dieleman et al. [30] attempt to leverage the large Million Song Dataset (MSD) by training a DCBL in an unsupervised fashion with beat-synchronized chroma and timbre components as input. They then initialize a Convolutional Multilayer Perceptron (CMLP) with the same architecture and the learned weights and biases from the DCBL, and fine-tune it in a supervised manner on artist recognition, genre recognition, and key detection. The authors observe that unsupervised pretraining helped the model converge faster, but it only modestly improved the accuracy in the tested tasks.

Classification

Soon after these self-supervised pretraining approaches, supervised pretraining ones based on descriptive tags and, later, editorial metadata started gaining interest.

Tagging Van den Oord et al. [32] demonstrate that the representations learned from an auto-tagging model using MSD and the associated last.fm tags outperform handcrafted features in genre classification in the GTZAN, Unique, and 1517-Artists datasets, as well as tag prediction in the MagnaTagATune (MTAT) dataset. To do this, they utilize the spherical K-means algorithm [43] on the audio's mel-spectrogram to extract low-level features in an unsupervised manner, and they reduce the dimensionality of the tags using Weighted Matrix Factorization (WMF) to handle the large, overlapping label set. They use this feature-label correspondence to train a linear regression model and two MLPs, one with a single and one with two hidden layers. The linear regression model generally performs the best, always outperforming an SVM using the handcrafted features and only falling behind the MLPs in tagging. Liang et al. [33] take a similar approach, utilizing auto-tagging on MSD for representation learning, but, instead, for improving collaborative filtering music recommendation models. They vector-quantize timbre features and train a 3-hidden-layer MLP, using the output of its last hidden layer to create a content-aware collaborative filtering latent space.

For the same dataset and tags, Choi et al. [34] propose a 5-hidden-layer Convolutional Neural Network (CNN) inspired by VGGNet [44] and using a mel-spectrogram as the input feature. They extend the suite of downstream tasks to Ballroom dance genre classification in the Extended ballroom dataset, genre classification and speech vs. music classification in GTZAN, emotion prediction in EmoMusic, vocal vs. non-vocal classification in a dataset from Jamendo, and audio event classification in Urbansound8K. Using a hyperparameter-optimised SVM and various combinations of the CNN activations, they show it significantly outperforms MFFCs and achieves performance close to the state-of-the-art. Further exploring CNNs for representation learning, Pons and Serra [25] release a set of CNNs trained in music auto-tagging in MSD and MTAT in a library entitled *MusiCNN* that also contains modules to facilitate transfer learning using the models' embeddings. The models at the time achieved state-of-the-art performance in music auto-tagging and competitive performance in many downstream tasks, with the musically-motivated CNN kernel design seemingly facilitating learning in limited-data scenarios [37].

In the direction of using domain knowledge to improve music representation learning, Won et al. [45] introduce a trainable frontend of triangular band-pass filters operating on the spectrogram exploiting the inherent harmonic structure. Combined with a simple CNN backend, they show improved performance in auto-tagging on MTAT, keyword spotting on the Speech Commands dataset, and sound event tagging on the DCASE 2017 subset of AudioSet.

Editorial Metadata Park et al. [46] use artist metadata with the premise that editorial metadata is subjective, unlike descriptive tags, and can relate to stylistic content in the music track. They use a deep, 1-dimensional CNN, and, to deal with the high dimensionality of the output layer and to avoid retraining every time a new artist is added, they use a Siamese network, a neural architecture that compares two inputs for similarity by processing them through shared weights. The models are trained on the MSD dataset and 30-second preview clips from the 7digital platform. They use the k-Nearest Neighbors classifier and a linear softmax classifier for genre classification on GTZAN, FMA, and tracks from NAVER, a Korean music service, and find that both the representations from the CNN and the Siamese network can learn useful representations, particularly for similarity-based retrieval. Lee et al. [47] later use a very similar approach to extend the metadata used to include album and track information. They train separate models for each type of metadata, but find that a joint model that uses all metadata generally works best in genre classification in the same datasets.

Correspondence

Tags Favory et al. [48] propose the use of text labels to enhance representations with audio semantics. They use an autoencoder (AE) with convolutional layers, a bottleneck, and a reconstruction objective to learn a representation of audio, and an AE with linear layers instead to learn a representation of the text labels. Contrastive loss is then used to align the two representations, minimizing the distance of audio features with similar semantic information. To pretrain, they create and preprocess a dataset of Freesound sounds and their associated tags, extracting a mel spectrogram from the former and multi-hot encoding the latter. Although the performance of other deep CNNs is still superior for genre classification in GTZAN and instrument classification in NSynth, the authors show their method outperforms both MFCCs but also a shallow CNN solely using the features from the audio autoencoder, suggesting the successful learning of music semantics from text. The authors later expand this approach to use a pretrained word embedding model to project the tags into a semantic space and an attention mechanism to better learn representation similarity [49].

Editorial Metadata Alonso-Jiménez et al. [50] collect a large set of metadata such as the artist, year, record label, and genres from the Discogs online music database. They opt to use COLA [51], a simple contrastive learning framework.

They design 4 experiments with different conditions for creating the positive and negative audio segment pairs, namely, whether they are from the same 1. track, 2. release, 3. artist, or 4. label. Using these objectives, they pretrain an EfficientNet [52], a CNN with width, depth, and resolution optimized for performance and inference time on image data, using the audio's mel spectrogram. The learned representations, as well as combinations of them from the different experiments, are evaluated using a 1-hidden-layer MLP on various downstream MIR tasks. All four experiments, with a slight edge for the artist association model, perform competitively or even better than other embedding models and task-specific models.

Playlists Ferraro et al. [53] use a very similar alignment-inspired approach to Favory et al. [48, 49]. They instead use three encoders, one aligning audio to genre tags, one for audio-to-playlist co-occurrences, and one for genre-to-playlist co-occurrences. They obtain data and metadata for training from Melon, a Korean music streaming service, and show that their approach utilizing contrastive loss to align the audio-genre-playlist correspondences typically outperforms approaches trying to directly predict genres or playlist co-occurrences. Alonso et al. [54] propose an approach utilizing a convolutional backbone and a contrastive learning framework similar to SimCLR [55]. They propose different heuristics for pair generation, including randomly selecting pairs based on playlist co-occurrences, prioritizing cases with the most co-occurrences, or aligning the projection of the audio representation to that of a Word2Vec model trained using the playlist name as sentences and the track IDs as words. They train the models on MSD using two architectural variants: one VGGish model [44], and one ResNet50 model [56]. They evaluate all these setups in multiple tasks, including genre, instrument, and mood tagging, showing generally higher performance compared to models trained on other image data.

Language Instead of text labels, other approaches have attempted to use natural language. Manco et al. [57] use an extension of BERT named Vision-and-Language BERT (ViL-BERT) [58] for joint music and language pretraining. They use a weakly aligned dataset of audio-caption pairs and pass audio segments through a MusiCNN model before feeding the embedding to the multimodal transformer. They introduce three learning objectives: 1. a Masked Language Modelling (MLM) objective, masking some of the tokens and tasking the model to predict the masks, 2. a Masked Audio Modelling (MAM) objective, following the same principle as MLM but for audio tasks, and 3. an Audio-Text Matching (ATM) objective, tasking the model to learn if an audio-text pair matches. After pretraining on approximately 200,000 audio-caption pairs, the learned representations are evaluated on a variety of downstream tasks, performing competitively with other embedding models and task-specific models. The authors also present an audio-language-learning framework based on contrastive learning [59]. The approach is similar to Favory et. all [48], although encoders are used for both the audio and text modalities with the objective that the resulting embeddings will lie as closely as possible in the joint embedding space if the audio and text are paired. The authors additionally propose a multi-task learning framework where the contrastive objective is combined with a self-supervised objective for the audio, similarly to SimCLR [55]. This approach enables state-of-the-art performance in audio-to-text and text-to-audio retrieval, and the produced embeddings are competitive in genre classification and auto-tagging.

Video Cramer et al. [60] exploit audio-visual correspondence for representation learning. They investigate different configurations of L^3-Net models [61], particularly focusing on the amount of data used for training and the impact of using an audio representation that is more perceptually relevant. They train the models on AudioSet [62] and demonstrate high performance with less training data compared to other models, particularly for the model configuration using a mel spectrogram rather than a linear one.

Self-Supervised Learning

Self-supervised contrastive learning, training a model with unlabeled data to be invariant to specific perturbations of that data, had been attracting interest for audio representation learning [63, 51]. Spijkervet and Burgoyne [64] employ this paradigm for music representation learning. They introduce domain-relevant perturbations such as additive white noise, gain reduction, random time offset, pitch shifting, and reverb applied to the waveform. In this framework, the perturbed signals would form a positive pair with the original signal, while other signals would form negative ones. The waveforms are then fed to a SampleCNN encoder [27], where the model's objective is to minimize the resulting embeddings' distance for positive pairs, while maximizing it for negative ones. Trained on MSD, this model was shown to perform at least on par with embeddings from supervised models, while being much smaller than models such as those in MusiCNN [25]. Similarly, Chang et al. [65] use contrastive learning to create suitable representations for audio fingerprinting. They choose relevant augmentations for identification such as adding noise and a random offset in the extraction window to create positive pairs and use the inner product between embeddings to make identification predictions. They train the model on a small subset of the Free Music Archive (FMA) dataset [66] and demonstrate good and robust predictions for audio identification, although using the representations for downstream tasks yielded mediocre results.

Inspired by the Bidirectional Encoder Representations from Transformers (BERT) architecture [67], Zhao and Guo [68] use a multi-layer bidirectional self-attention transformer encoder [69] with a combination of downsampled features such as a mel spectrogram and MFCCs as the input. They train the model with two objectives: 1. Contiguous Frames Masking, where multiple, non-overlapping groups of consecutive frames are masked, and, similarly, 2. Contiguous Channels Masking. A dataset containing Music4all, FMA-large, and MTG-Jamendo is used for training. The learned embeddings are evaluated in genre classification and auto-tagging, outperforming existing learned representations. More recently, Li et al. [70] introduce a family of models entitled MERT that are also based on a BERT-style transformer encoder. They propose teacher models for creating pseudo-labels for the Masked Language Modelling training objective, ultimately opting for an acoustic teacher based on a Residual Vector Quantization Variational AutoEncoder (RVQ-VAE) and a musical teacher based on the Constant-Q Transform. Multiple variations of the models are evaluated on a large set of tasks and datasets, with competitive and some state-of-the-art results.

Music Generation

Lastly, music generation has been exploited as a proxy for creating music audio representations. Castellon et al. [36] exploit the representations learned by Jukebox [71], a generative music model that uses Vector-Quantized Variational Autoencoders (VQ-VAE) to compress raw audio to tokens, models the tokens using autoregressive Transformers, and then decodes them to audio. Due to the size of the model and its representations, the authors decide to use a single layer as the representation for downstream tasks. They iterate through different layers of the model, and find that the middle layer is most performant on downstream tasks, considerably beating other embeddings from representation learning models in auto-tagging, genre classification, emotion recognition, and, especially, key detection.

1.2.2 Audio Representation Evaluation Frameworks

In the neighboring domain of audio, a few extensive efforts have been made to create tools for evaluating audio representations. One of those is the *Holistic Audio Representation Evaluation Suite (HARES)* [72, 73]. As described in the accompanying paper, they implement multiple tasks, including auto-tagging on MTAT and pitch estimation and instrument classification on NSynth, and have a fairly rigid 1-hidden-layer downstream MLP. A more comprehensive package is the *Evaluation package for Audio Representations (EVAR)* [74]. The package facilitates the use of multiple models, including pretrained ones, in a variety of tasks and datasets, including music-related ones such as GTZAN, NSynth, and the Pitch Audio Dataset. It further provides functionality for fine-tuning, controllable through a template *yaml* file. The *HEAR* benchmark [75, 76] consists of 19 tasks, including GTZAN genre and music vs speech, NSynth Pitch, and Beijing Opera Percussion. It provides data, code, and an API for researchers to integrate their model. Recently, a representation evaluation benchmark specifically aimed at MIR tasks was released, called the Music Audio Representation Benchmark for universaL Evaluation (MARBLE) [77]. MARBLE implements a wide range of MIR tasks and datasets, and, currently, is primarily submission-based. A fixed downstream setup is enforced, with a one-layer

512-unit MLP for all tasks apart from source separation, for which a 3-layer 512-unit LSTM is used.

1.3 Contributions

The primary contribution of this thesis project is the development and release of an open-source toolkit for music audio representation evaluation. The toolkit's focus is the easy configuration of extensive evaluation experiments for representation learning pipelines aimed at Music Information Retrieval tasks. It is built upon the principle of reproducibility, meaning all aspects of the pipeline are transparent and anyone can reproduce experiments simply from their configuration file. At the same time, it focuses on modularity and extendability, to make it easier for others to contribute their own datasets, tasks, models, and metrics. Lastly, it provides various functionalities for representation understanding, such as robustness evaluation, difficulty of information extraction analysis, interactive confusions visualization, and others.

The secondary contribution is the release of an extensive evaluation of deep representations from popular embedding models in a variety of MIR tasks. This includes an evaluation of performance in each task, of the different downstream model setups required for reaching good performance, and of the robustness of the representations to audio deformations relevant to each task.

Chapter 2

Methods

In this section, an overview of the design of the toolkit will be given (see Sec. 2.1), followed by more details about its core components (see Sec. 2.2) and of additional experiment analysis functionality that it offers (see Sec. 2.3).

Please note that, as is the case with living open-source software projects, it is important to always consult the latest documentation, guides, and other material accompanying the toolkit's releases for an up-to-date view of its functionality and design. This document describes and can only describe our initial software design aspirations for the toolkit's first release.

2.1 Toolkit Overview

The principles we laid out in Section 1.1.3 to expand our thinking beyond benchmarks should be evident in the software design. This is why the toolkit is developed as an open-source project, inviting external contributions. However, simply having the source code available is insufficient for achieving reproducible evaluation work. Challenges like labor-intensive data collection, unclear experiment setups, and complex result visualization must also be addressed.

To overcome these challenges, the toolkit provides a configuration-driven approach to evaluation. Researchers can craft a single configuration file that encompasses parameters for the representation learning pipeline. By utilizing this file, they can execute evaluation experiments without the need to collect data or code. This configuration file encapsulates essential experiment details, including tasks, datasets, embedding models, downstream models, training parameters, metrics, and even considerations like audio deformations for robustness assessment, all of which will be explored in Section 2.2. Anyone equipped with this configuration file can reliably replicate the experiments outlined within it.

Distinguishing itself from traditional benchmarks, where the whole representation learning pipeline apart from the representation itself is often fixed, this toolkit encourages users to assess the impact of various components of the pipeline to the system's behavior. For instance, users can investigate the effects of various embedding extraction frequencies, different techniques for embedding aggregation or dimensionality reduction, diverse audio perturbations, varying downstream models, and more. This flexibility is achieved through a modular design, aiming to keep components loosely interconnected. This modularity facilitates easy incorporation and contribution of new models, metrics, datasets, and other elements by users.

The toolkit format was chosen instead of the package format as we envision this software as a standalone suite of evaluation tools that are primarily extensible rather than wrappable. We make use of some excellent software from the MIR and audio research community, such as `mirdata` [78], `mir_eval` [79], Essentia [15] and Essentia models [80], and audiomentations [81], and we generally encourage contributions to those directly so that evaluation implementations are not fragmented further. We wrap these components in a way that your relevant contributions to those packages would be available for use with this toolkit too.

2.2 Core Components

The core components are the building blocks for every run, and they are set in the configuration file for each experiment as seen below.

```
# CONFIG.yml
```

```
experiments
  - task: ...
    datasets: ...
    deformations: ...
    embedding_models: ...
    downstream_models: ...
    metrics: ...
```

These components and a basic configuration for each, detailed in the sections that follow, are adequate for building an evaluation experiment, such as the one depicted in the flowchart in Fig. 2.2. Each experiment contains a single task, for which every combination of datasets, deformations, embedding and downstream models, and metrics will be computed. The experiment can be run with a single command that takes the name of the configuration file as an argument.

```
python run.py experiment -c CONFIG.yml
```

The user can optionally run parts of the experiment independently with different commands, if, for example, they want to run specific parts in different hardware.

```
python run.py deform -c CONFIG.yml  # compute deformations from audio
python run.py generate -c CONFIG.yml  # compute embeddings
python run.py train -c CONFIG.yml  # train downstream models
python run.py evaluate -c CONFIG.yml  # evaluate trained models
```

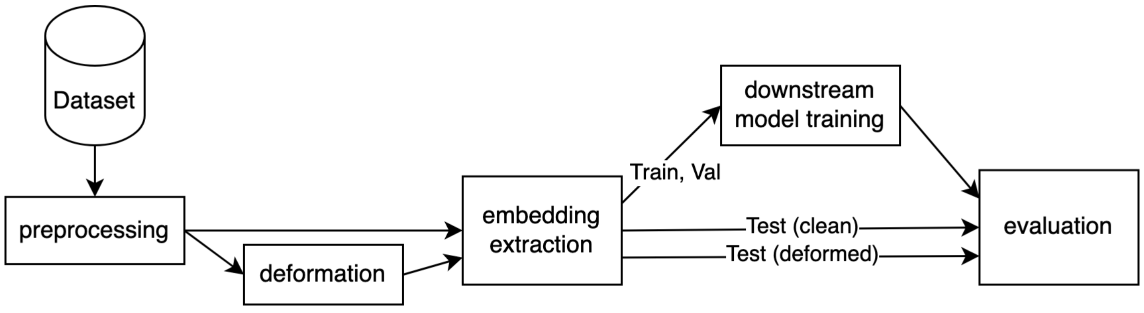

Figure 1: Sample flowchart for a basic evaluation experiment.

Options to resume incomplete or failed runs are provided, along with manual overwrites to skip computing embeddings for clean or deformed audio, to overwrite parts of the configuration, or to limit computational resource use.

2.2.1 Tasks

Every experiment revolves around a single task. The task configuration contains information about the name and type of the task, which helps automatically configure various parameters in the pipeline such as the type of downstream models, the type of metrics, and even the relevant labels to retrieve from a dataset. It also contains information about the embedding-prediction aggregation strategy used.

```
# TASK CONFIGURATION
task:
  name: autotagging
  type: multilabel_classification
  aggregation: embedding_mean  # or ‘prediction’
```

2.2.2 Datasets

The dataset parent class handles many core functionalities, such as downloading the audio, collecting relevant metadata for a task, providing the appropriate audio and embedding directories, and handling label encoding. Implementing a new dataset is as simple as implementing 4 key methods:

`dataset.download()`

The download method programmatically downloads the relevant audio and metadata for a dataset. In existing datasets, the `wget` [82] program is used download entries from Zenodo or download servers, allowing for download progress and resumption of failed downloads. If utilities for downloading a specific dataset are provided with the dataset, those are preferred. Checksum verification for downloaded files can also be implemented at this point. A `dataset.download_metadata()`

method can optionally also be implemented. If an operation does not require the full audio, an attempt will be made to simply download the metadata, if that option is implemented. A decorator wraps these two methods to check if the data exists at the right place and attempts to load the metadata.

`dataset.load_track_ids()`

After downloading, this method collects the relevant track IDs. Different IDs can be returned depending on the `dataset.task_type`, if, for example, certain IDs should not be included in a particular task. The method expects a list of strings or integers.

`dataset.load_label()`

Similarly to track ids, this methods collects the label(s) for each track ID in the form of a dictionary. Labels can be returned according to the `dataset.task_type` and `dataset.task_name`.

`dataset.load_audio_paths()`

This method expects a dictionary of filepaths keyed by their respective track ID.

Optionally, one can implement the `dataset.get_splits()` method, which can return predetermined splits for a dataset. If this method is not implemented, a random stratified split will be returned. The parent dataset class contains many more methods and decorators to ensure safe loading and handling of datasets. It always contains functionality like encoding the labels depending on the task type and providing a label decoder.

Several datasets are implemented as of the submission of this thesis project, including TinySOL [83], Beatport EDM [84][85], MagnaTagATune [22], MTG Jamendo [86], and VocalSet [87]. The first two are implemented using the `mirdata` package with a simple wrapper dataset child class. All `mirdata` datasets are theoretically supported in this toolkit out of the box, but not many of them have been tested in relevant tasks yet.

In the configuration file, the name and desired data directory for the dataset are all that is needed for the toolkit to download and set up the dataset for an experiment.

```
# DATASET CONFIGURATION
datasets:
  - name: magnatagatune
    dir: data/magnatagatune/
    split_type: all  # ‘all’ to use all available splits, or ‘single’
  - name: mtg_jamendo
    dir: data/mtg_jamendo/
    split_type: single
```

2.2.3 Audio Deformations

Deformation robustness is a great way to test a production MIR system. Deformation scenarios with one or multiple deformations can be created. The library `audiomentations` is used under the hood to generate the deformations, and the syntax expected by it is retained, meaning the user can simply check its documentation to create deformation scenarios from tens of options.

```
# DEFORMATION CONFIGURATION
deformations:
  - - type: Mp3Compression
      params:
        min_bitrate: 32
        max_bitrate: 32
        p: 1
  - - type: Gain
      params:
        min_gain_in_db: -12
        max_gain_in_db: -12
        p: 1
```

```
    - type: AddGaussianSNR
      params:
        min_snr_in_db: 0
        max_snr_in_db: 0
        p: 1
```

2.2.4 Embedding Models

Multiple embedding models are already implemented, including VGGish-AudioSet [44], MusiCNN-MSD [25], EffNet-Discogs [50], OpenL3 [60, 61], NeuralFP [65], CLMR [64], MERT-v1-95m [70], and MULE [88]. The first 4 are implemented with the Essentia models [80], which provides a simple-to-use wrapper for running inference from pre-trained models. Custom models are currently implemented to receive an input file path and return the expected embeddings. This functionality will be developed further in the future so that embedding models can operate consistently and with configurable batch sizes, since embedding inference is usually the slowest part of a representation learning evaluation pipeline.

```
# EMBEDDING MODEL CONFIGURATION
embedding_models:
  - vggish-audioset
  - openl3
  - mert-v1-95m
```

2.2.5 Downstream models

A typical downstream model configuration for a simple classifier is presented below.

```
# DOWNSTREAM MODEL CONFIGURATION
downstream_models:
  - type: classifier
    emb_dim_reduction: False  # ‘False’ or ‘PCA’
```

```
emb_shape: infer
hidden_units: [infer, infer]
output_activation: softmax
weight_decay: 1.0e-5
# optimizer
optimizer: adam
learning_rate: 1.0e-3
# training
batch_size: 100
epochs: 100
patience: 10
train_sampling: random
```

In an experiment, we might have multiple embeddings that are of very different sizes. This could make it harder to compare two embeddings with a fixed-size downstream model that might be benefiting one of the two embeddings more. Apart from providing the option to add as many different downstream models as desired, we also provide two different techniques to deal with this issue. The first is the option to perform dimensionality reduction of the input embeddings to a desired shape using Principal Component Analysis (PCA). This could mean larger embeddings won't have as unfair of an advantage compared to smaller ones. However, it is hard to know whether dimensionality reduction with PCA will work well for your task in particular. Thus, the second option is the ability to create models with dynamic hidden layer number and sizes based on the size of the input representation. For example, by configuring `hidden_units: [infer, infer]`, two hidden layers will be created with sizes determined through a linear interpolation of the input and output size. If `hidden_units: [power_infer]` is provided instead, a power regression for a single layer will be computed instead.

Other configuration options are fairly standard parameters for model training. Entering a parameter that might not be appropriate for the dataset and type of task will raise a helpful warning. Custom models can also be implemented by providing the model name in the `type` option and placing a Keras model in the downstream model directory.

2.2.6 Metrics

Every task comes with a set of relevant metrics already implemented, some of which are implemented with the `mir_eval` [79] package. However, preexisting metrics from one task can be added to the configuration file for other tasks, and additional custom metrics can be written in the metrics module and specified in the configuration.

```
# METRICS CONFIGURATION
metrics:
  - AUC-ROC
  - weighted_score_key
```

2.3 Experiment Analysis

For each run, a `run_id` can be specified, to aid with experiment tracking. If it is not provided, a run will be created and identified with a timestamp of the experiment start time. Runs are placed in the `logs` directory.

2.3.1 Training

During each model training, training and validation logs are produced using the Tensorboard [89] utility of Tensorflow [89]. Tensorboard allows you to visualize the training and validation metrics specified (see Fig. 2), which proves extremely useful when one can be using multiple downstream models in an experiment. In the default behavior, the weights of the model with the best validation performance will be saved for use later in the evaluation.

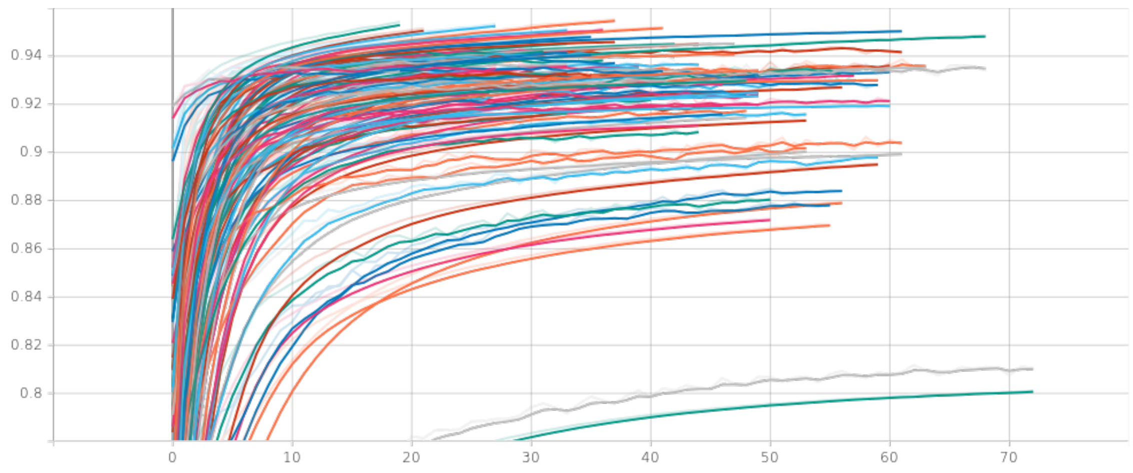

Figure 2: AUC of downstream models during training, visualized with Tensorboard.

2.3.2 Evaluation

Results Visualization

] Given the number of controllable parameters for an experiment, it is important to have a method for obtaining and visualizing the data of interest. This is why we have developed and will continue developing utilities for experiment and results analysis and visualization.

One of those, called `autotable`, creates tables out of the evaluation metrics of a specific run. It expects the run ID as well as a preset. Presets include some tables focusing on deformations, others focusing on downstream classifiers, and others trying to fit as much information in a table as possible. In fact, the tables for this paper were generated with `autotable` with the preset `all_deform_all_models`, which creates a table for a single class and dataset that includes all embedding models, and all deformations and downstream models.

Given the dimensionality of these evaluations, however, we have been developing an interactive version of `autotable` in which, instead of presets, particular variables-parameters can either be frozen or be made active, and a good table preset for the given active variables will be inferred and visualized live. This prototype is being built with streamlit [90], but it is not ready for release at this time.

Classification Analysis

Evaluation results save metrics per label in addition to the averaged ones, when applicable. That said, given that most runs might contain multiple experiment configurations, it might be impossible to manually analyze all results. Because of this, we have been experimenting with an interactive confusion matrix interface, which allows the user to select a misclassification between two classes, and a list of (some) misclassified audio is presented. The user then has the option to play the audio, or view the mel spectrogram of the audio, giving them an idea of why this particular sample was misclassified. The prototype is currently based on Weave [91], although other options are being considered.

Chapter 3

Evaluation

In this section, the experimental setup for each experiment will be given (see Sec. 3.1, followed by the results and an analysis of them (see Sec. 3.2.

3.1 Experimental Setup

To demonstrate the capabilities of the toolkit, we designed an evaluation of seven representation learning models, including a few different configurations of one model, in six different downstream tasks, four different datasets, and with five different downstream models, fitting the computational and temporal resources available for this thesis project. The configuration file to replicate these experiments is provided in Appendix A. Abbreviations we use in the results for downstream models and deformation scenarios are included in parenthesis next to their respective titles in the overview that follows.

3.1.1 Embedding Models

We choose seven pre-trained models, which were previously described in Sec. 1.2.1. These models vary in their architecture, size, amount of training data, training dataset, input representation, and training strategy. Their model architecture is presented in Table 3.1.1, and their specific pretraining dataset, training paradigm, and size of the chosen layer for embedding extraction are presented in Table 3.1.1. For the models VGGish-AudioSet, EffNet-Discogs, MusiCNN, and OpenL3 we note that we used the pretrained models and implementations from Essentia models [80]. We also computed four different embedding configurations for the MERT-v1-95M. Specifically, we used a configuration with the mean of layers 0, 1, 2, and 3, denoted MERT-v1-95M$^{1:4}$, one with the mean of layers 4, 5, 6, 7, 8, denoted MERT-v1-95M$^{4:8}$, one with the mean of layers 9, 10, 11, 12, denoted MERT-v1-95M$^{9:12}$, and one with just the middle layer, denoted MERT-v1-95M^{6}.

Table 1: Architecture of chosen embedding models.

	Architecture	Size	Input representation
VGGish [44]	CNN	∼100M	Mel Spectrogram
EffNet [50]	CNN	∼5.3M	Mel Spectrogram
MusiCNN [25]	CNN	∼8M	Mel Spectrogram
OpenL3 [60][61]	CNN & RNN	∼4.8M	Mel Spectrogram
NeuralFP [65]	CNN	∼20M	Mel Spectrogram
CLMR [64]	CNN	∼2.5M	Waveform
MERT-v1-95M [70]	CNN & Transformer	∼95M	Waveform

Table 2: Training configuration of chosen embedding models.

	Dataset	Training Paradigm	Emb. Size
VGGish [44]	AudioSet [62]	Tagging	128
EffNet [50]	Discogs [92]	Supervised Contrastive	1280
MusiCNN [25]	MSD [23]	Tagging	200
OpenL3 [60, 61]	AudioSet [62]	Self-Supervised	512
NeuralFP [65]	FMA [66]	Self-Supervised Contrastive	128
CLMR [64]	MTAT [22]	Self-Supervised Contrastive	512
MERT-v1-95M [70]	Private	Masked Language Model	768

Embeddings for each recording are computed and then mean-aggregated into one embedding per recording. The aggregated embeddings are then used for model training and prediction. While this is less computationally intensive than using the original embeddings and aggregating the predictions, it is not clear how system performance is affected. Future work will include an evaluation of the impact different embedding sampling and aggregation strategies have.

3.1.2 Datasets and Tasks

MagnaTagATune - Music Autotagging

The MagnaTagATune (MTAT) dataset [22] has been widely used for both training and evaluating music auto-tagging systems since its release in 2009. It contains 25,877 30-second audio clips, some of which are from the same track, and most of which have one or more tag annotations that were collected from an annotator agreement game. We use the provided low-fidelity, mono, `MP3` audio which has a sampling rate of 16,000 Hz and a bit rate of 32 kbps. We use the split used in [27] and all but the 50 most frequent tags, as is usually done for this dataset. We note that we also discard audio clips with no associated top 50 tags, a decision that is not always taken in previous works [37], meaning the dataset size is reduced to 21,108 excerpts. The top 50 tag frequency is presented in Fig. 3. As is the case in previous work, we use the Area Under the Receiver Operating Characteristic curve (AUC-ROC) and the Area Under the Precision-Recall curve (AUC-PR), metrics that measure the area under the True Positive Rate - False Positive Rate plot and the area under the Precision-Recall (True Positive Rate) plot respectively for different classification prediction thresholds.

Beatport EDM - Key Estimation

The Beatport EDM dataset [84, 85], released in 2017, is comprised of 1,486 two-minute music audio clips from the Beatport online music store. The clips are in various subgenres of the Electronic Dance Music (EDM) genre and have accompanying global key annotations. They are stereo `MP3` files, with a sampling rate of 44,100 Hz and a bit rate of 96 kbps. We discard clips that contain multiple or no key annotations, reducing the dataset size to 1,272 clips, and create an 80-10-10 train-validation-test split. We report the micro-average and macro-average of a weighted score which evaluates the quality of the prediction. For this, we use the implementation of the `mir_eval` package [79], whose scoring rules are given in Table 3. Key annotation frequencies for the Beatport EDM split used are presented in Fig. 4.

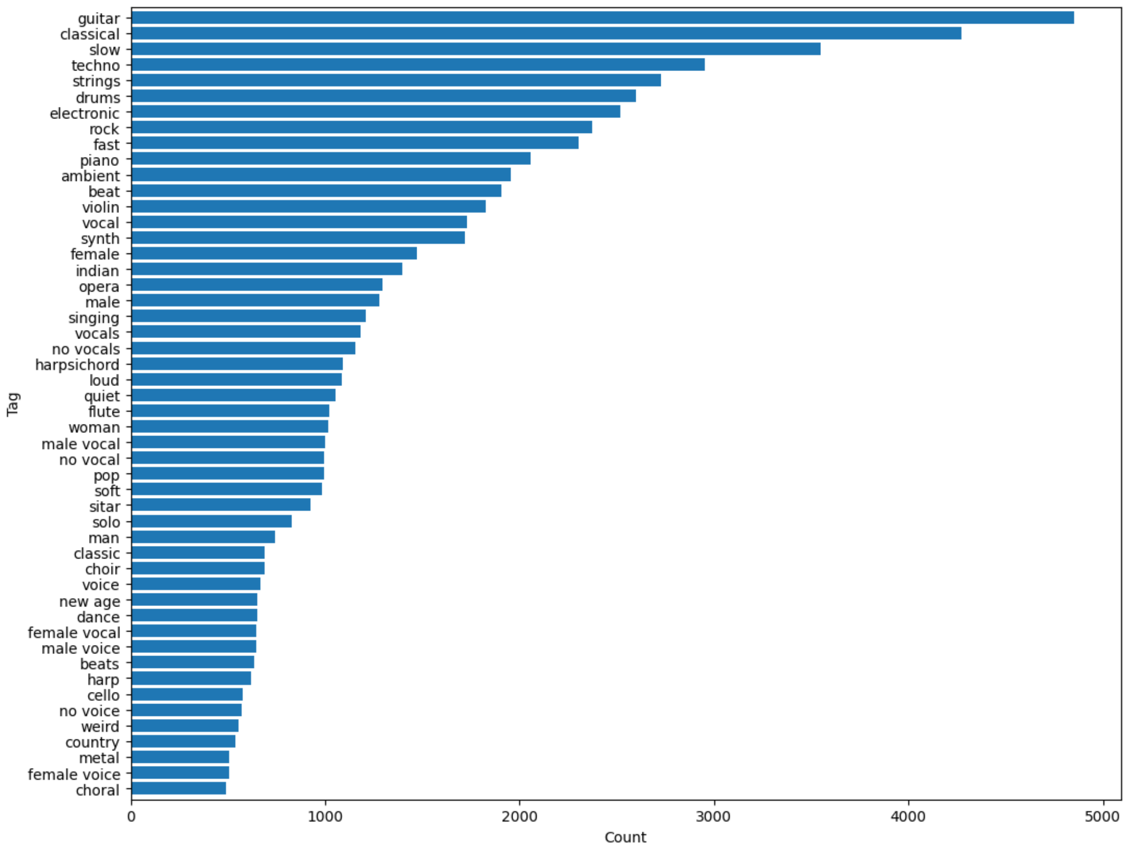

Figure 3: Counts of the top 50 tags in the MagnaTagATune split used.

TinySOL - Instrument Recognition, Pitch Class Classification

TinySOL [83] is a dataset containing single musical note recordings from 14 instruments released in 2020. It strictly contains notes played in "ordinario" style, meaning no extended playing techniques or other physical instrument modifications are used. The 2,913 audio clips included are provided in the `WAV` format in mono at a sampling rate of 44,100 Hz a bit depth of 16 bits. For each recording, annotations for the instrument, its instrument family, the pitch, the dynamics, and other

Table 3: Scoring rules for key detection

Relationship	Example (Truth, Prediction)	Score
Same key and mode	C minor, B$^\sharp$ minor	1.0
Prediction perfect fifth above truth	C major, G major	0.5
Relative major/minor	C major, A minor	0.3
Parallel major/minor	C major, C minor	0.2
Other	C major, F minor	0.0

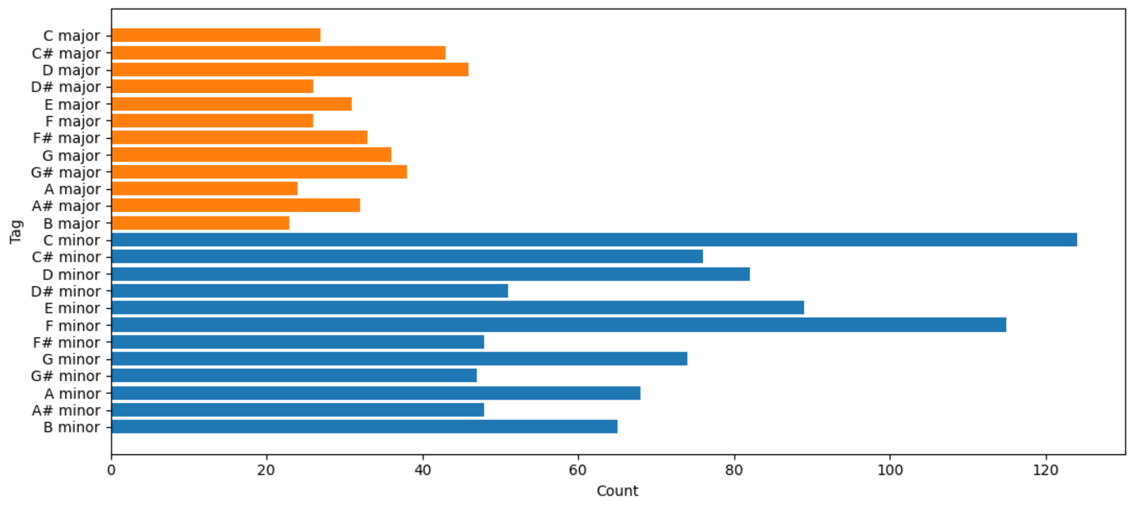

Figure 4: Key annotation counts for the Beatport EDM split used. Enharmonic keys are assigned to their respective sharp (#) key.

instrument-playing details are included. We use TinySOL in two tasks: instrument recognition and pitch class classification. We chose these two lower-level tasks to gain some understanding of whether the deep representations contain lower-level information that could be relevant to other tasks. For instrument recognition, we use only the first data split provided by the authors of the dataset. We report the micro- and macro-average F1-score, the harmonic mean of precision and recall often preferred to accuracy in tasks using unbalanced datasets. The micro-average refers to scores for each sample being given the same weights, while the macro-average gives the same weight to each class. For pitch class classification, no official splits are provided, so we create an 80-10-10 stratified train-validation-test split. We use the same metrics as for instrument recognition. The instrument and pitch class frequencies are displayed in Fig. 5 and 6 respectively.

3.1.3 VocalSet - Singer Identification, Vocal Technique Identification

The VocalSet dataset [87], released in 2018, contains 3,560 a capella singing recordings of twenty professional singers, eleven male and nine female. There are multiple recordings for each singer capturing a broad range of vowels, singing techniques, and

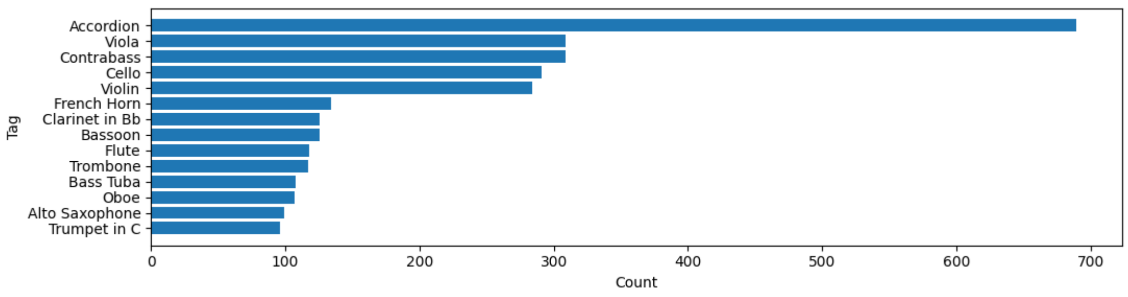

Figure 5: Counts of the instrument annotations in the TinySOL dataset.

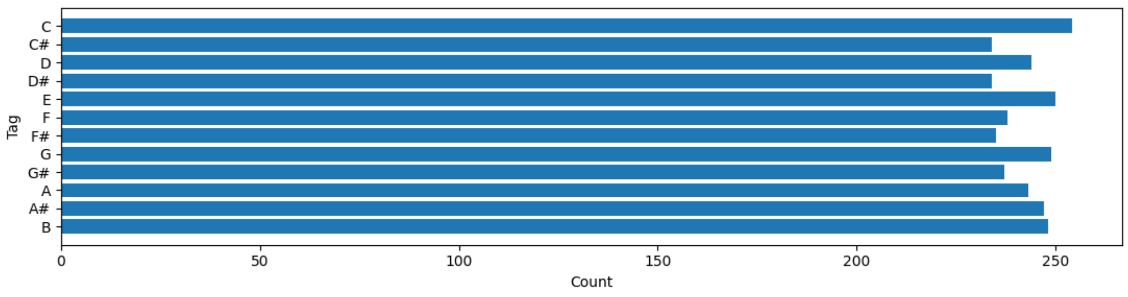

Figure 6: Count of each pitch class annotation in the TinySOL dataset. Enharmonic pitch classes are assigned to their respective sharp (#) pitch class.

contexts such as scales, arpeggios, long tones, and excerpts. The audio recordings are mono and provided in `WAV` format, with a sampling rate of 44,100 Hz and a bit depth of 16 bits. We use VocalSet for two experiments: singer identification and vocal technique identification. For singer identification, we create an 80-10-10 stratified train-validation-test split and report the micro- and macro-average F1-scores. For vocal technique identification, we modify the original split provided by the dataset authors by creating a validation set of three singers (female 9, male 9, and male 11) that previously belonged to the training set. We decided to create a validation set for consistency across all experiments in the criterion used for model selection for the evaluation. Because of the high imbalance of samples for each singing technique, we opt to use only ten vocal techniques, the same ones as those used in the experiment by the dataset's authors [87], reducing the audio clip number to 1,912. As before, we report the micro-average and macro-average F1-scores. Annotation counts for the two tasks that use the VocalSet dataset are presented in Fig. 7 and Fig. 8 respectively.

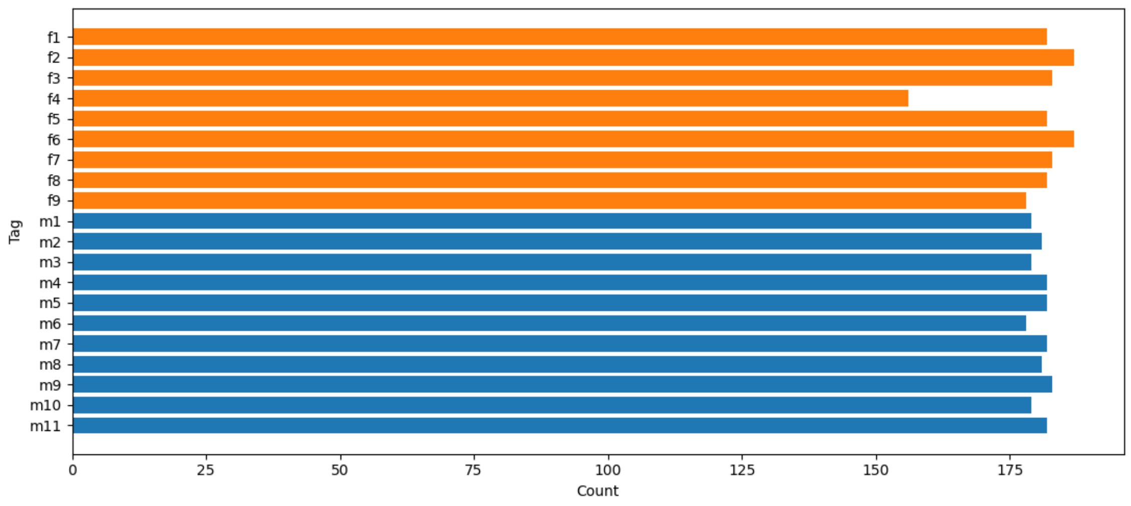

Figure 7: Counts of recordings from each singer in VocalSet, where "f" indicates female, and "m" indicates male.

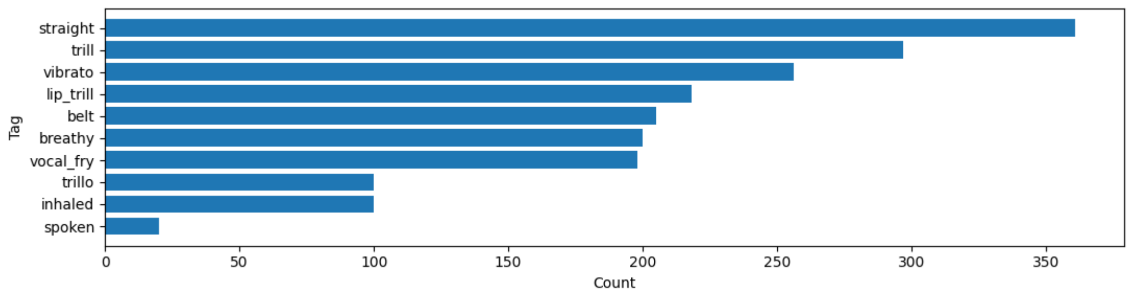

Figure 8: Counts for top 10 vocal techniques in VocalSet.

3.1.4 Deformations

We test four deformation scenarios on every task, as presented below. These deformations are relevant to systems attempting to solve all tasks in this evaluation, albeit the difficulty of being robust to a scenario might differ per task. We believe that these deformations would generally not affect or very mildly affect the performance of a human with normal hearing in this experiment's tasks.

1. Gaussian White Noise is added to the audio with a Signal to Noise (SNR) ratio of 15 dB (**D1**). While the occurrence of white noise is infrequent in real-world audio situations, its inclusion serves as a test for assessing the system's overall resilience to noise across the entire spectrum. Noise at 15 dB SNR can be

characterized as moderately quiet, and it's not an uncommon occurrence in analog radio or when you are conversing with someone during a light rainfall. It is fair to assume most humans' performance in the experiment's tasks would not decrease under this scenario.

2. Gaussian White Noise is added to the audio with a Signal to Noise (SNR) ratio of 0 dB (**D2**). Noise at this level has the same energy as the signal, and, although this does not exactly translate to having the same perceived loudness, it's a level where humans might need to listen to a sample for a bit longer in order to interpret its content.

3. `MP3` compression to a bit rate of 32 kbps (**D3**). This is the lowest supported bit rate in the `MPEG-1` audio standard, although `MPEG-2` and `MPEG-2.5` support even lower bit rates. Still, at this compression level, instrument timbre might require more careful listening to be identified, especially when multiple instruments are present. Since the low-fidelity audio provided in the MTAT dataset already has a bit rate of 32 kbps, this deformation is ignored for the task of auto-tagging.

4. Gain reduction of 12 dB (**D4**). In digitized audio, a gain reduction could be considered a type of bit depth reduction, since the audio can be normalized to its original amplitude but would have lost some of its amplitude resolution or been subjected to quantization errors. Importantly, different versions of recorded audio can exist at different gain levels, and the hope is that for most MIR tasks deployed systems would be able to perform consistently across the different gain levels.

3.1.5 Downstream models

Five different classifiers are used for all tasks.

- Single-Layer Perceptron (SLP); no hidden layers (**M1**)
- Multi-Layer Perceptron (MLP); one layer; [128] hidden units (**M2**)

- Multi-Layer Perceptron (MLP); two layers; [256, 128] hidden units (**M3**)
- Multi-Layer Perceptron (MLP); one layer; hidden units inferred from linear interpolation of input and output size (**M4**)
- Multi-Layer Perceptron (MLP); two layers; hidden units inferred from linear interpolation of input and output size (**M5**)

One of the models contains no hidden layers to test whether information from the embedding is linearly separable. Two of the models have a fixed size, and two of the models' hidden units depend on the input and output size. For a given task with a fixed output size, this means that representations that are larger in size will use larger downstream models.

For all models, the Adam optimizer is used with a learning rate of 1.0e-3. Hidden dense layers use ReLU activation, and L2 regularization with a weight decay of 1.0e-5 is used for kernel and bias regularization. The output activation is sigmoid for auto-tagging on MTAT, and softmax for all other tasks. Models can train for up to 100 epochs, although training stops if the validation loss has not improved for 10 epochs.

3.1.6 Hardware

Embedding inference and downstream model training and evaluation were primarily run on an NVIDIA GeForce RTX 2060 Mobile (80 Watt TDP) with 6 GBs of VRAM, with some inference of the MERT and CLMR models being completed on an AMD Ryzen 7 4800H CPU and a system with an NVIDIA GeForce RTX 2080Ti (250 Watt TDP) with 11 GBs of VRAM.

3.2 Results

3.2.1 Music Auto-Tagging, MagnaTagATune

Table 3.2.1 shows the different versions of MERT that we tested outperform other models both in terms of AUC-ROC and AUC-PR, albeit not always using the same layer representation. Interestingly, using just the middle layer or the mean of several middle layers of MERT seems to work better than representations taken from the first and last layers. CLMR and OpenL3 come at a close second, with both performing particularly well in the clean audio scenario, followed by EffNet-Discogs. NeuralFP, VGGish-AudioSet, and MusiCNN-MSD perform the poorest in terms of AUC-PR. Moderate differences can be observed between results of the same embedding model using a separate downstream classifier, apart from the case of NeuralFP, where the difference in AUC-ROC between the SLP (M1) and the larger 2-layer MLP (M5) is more than 10, although the difference in AUC-PR isn't as notable. M5's first and second hidden layers are double and equal to the NeuralFP embedding size respectively, which, coupled with the poorer performance using a linear classifier, might point to some information extraction difficulty. As an audio fingerprinting model, NeuralFP's identifiability objective might not be well-aligned with the musical aspects we first pay attention to in a track.

Looking at the deformations, we observe that most of the models only have small performance decreases for the light noise (D1) and `MP3` gain reduction deformations. That's not the case, however, with MusiCNN-MSD and NeuralFP, whose AUC-ROC decreases by around 10 in the presence of light noise, with a more moderate drop for the gain reduction scenario. The MERT models experience almost no AUC-ROC decrease in these scenarios, particularly in the compression one. However, in the stronger noise scenario, we observe performance decreases in both AUC-ROC and AUC-PR ranging from around 5 to around 15 points. MERT is, again, the most resilient, with all other models experiencing quite significant drops. An encouraging result from this experiment is the very mild performance drops across the board in the gain reduction scenario, with the highest observed drops being up to around 5 points in AUC-ROC and AUC-PR for specific downstream classifiers for MusiCNN and NeuralFP.

3.2.2 Key Estimation, Beatport EDM

Table 3.2.2 shows significantly lower scores across the board in the task of global key estimation, although, again, a MERT version utilizing middle layers comes on top. Key estimation could be an especially challenging task for these representations, owing to the resolution of the mel spectrogram most use, or to the pitch invariance objectives they might have been trained with. Models utilizing a mel spectrogram input, apart from NeuralFP, perform the worst in this task, but so does CLMR, a waveform-based model, likely because one of the augmentations it uses during contrastive training is pitch shifting between -5 and +5 semitones. Even though NeuralFP is a mel-spectrogram-based model, it comes second with a large performance difference over the rest. Just like CLMR, it is based on self-supervised contrastive learning, but it does not train for pitch invariance.

Performance differences between the micro average and macro average weighted score are quite significant. All models exhibit worse macro average scores, pointing to the class imbalance of the dataset, but also to the fact that more preprocessing and training optimizations could have a significant impact in a key estimation system utilizing these representations. When it comes to audio deformations, there is a lot more variance between systems. Models like VGGish-AudioSet and OpenL3 seem almost unaffected by all deformations. Others like EffNet-Discogs, NeuralFP, and CLMR underperform significantly during the loud noise scenario (D2) and using the SLP classifier (M1), however, they manage to recover some performance with the larger classifiers.

3.2.3 Instrument Recognition, TinySOL

Table 3.2.3 shows good overall performance on instrument recognition from single note and instrument recordings for most models. VGGish-AudioSet and MusiCNN-MSD show poor performance using the SLP, particularly when it comes to their macro average F1-score, but have big or moderate performance increases, respectively, with the larger downstream classifiers. NeuralFP yields the lowest, by far, performance on the linear classifier, and the sizeable performance boost larger downstream classifiers give it is still not enough to make it useable for instrument recognition. As was the case in previous experiments, MERT performs extremely well, but this time OpenL3, paired with the larger two-hidden-layer model, outperforms it. In spite of the instrument class imbalances in the dataset, the differences between micro and macro average F1-scores are a lot less striking.

Perhaps the most interesting aspect of this table is the behavior of the representations under different deformations. Unlike previous experiments, all models are significantly affected by the quiet noise deformation, with OpenL3 and CLMR performance plummeting by around 50 and 60 points respectively in some cases. All models become unusable in the louder noise scenario, although different MERT layers exhibit different behaviors, with the middle layer managing to reach an F1-score of around half. Milder but still noticeable drops can be seen for the compression scenarios, and only OpenL3 and MERT manage to perform well under gain reduction. Given the instruments are discernible by humans for each of these deformation scenarios, these deformation behaviors are worrying, particularly in how they might affect behavior of the representation in a higher-level task.

3.2.4 Pitch Class Classification, TinySOL

Low performance can be observed in Table 3.2.4 showing results for pitch class classification. Being such a low-level task, it was chosen to investigate if pitch information is present and easily accessible. All models but MERT and NeuralFP are unable to discern pitch class, at least at the resolution of a 12 equal-tempered tone scale, a pattern similar to that of the global key detection experiment (see Table 3.2.2). MERT is the clear winner, with near-perfect performance using the M5 classifier, and while NeuralFP struggles with the SLP, it does perform well with M5, suggesting that pitch information is there but is possibly harder to extract. Unlike previous experiments, the last layers of MERT are the ones performing better in this task. Mild to moderate performance drops are observed for MERT and NeuralFP in the deformation scenarios.

3.2.5 Singer Identification, VocalSet

Table 3.2.5 shows the singer identification performance. Again, MERT performs the best, with the middle layers getting close-to-perfect scores. CLMR and OpenL3 follow, although at some distance. VGGish-AudioSet, EffNet-Discogs, MusiCNN-MSD, and NeuralFP all struggle with this task especially using the SLP, although larger downstream models increase their performance significantly. Given the classes for this task are relatively balanced, the macro average F1-score expectedly does not differ much from the micro average one.

We do, however, observe another case of significant performance reduction in the presence of deformations. Even the quiet noise scenario plummets performance in all models and reduces the F1-score of MERT to around half, with the louder noise scenario resulting in a near-complete inability to predict. `MP3` compression has a significant but less pronounced impact, while gain reduction equally affects all models negatively apart from MERT and OpenL3. Coupled with the findings from the instrument recognition experiment (see Table 3.2.3), it seems that the models' representation of timbre-related characteristics is much more susceptible to these deformations than chroma-related ones.

3.2.6 Vocal Technique Identification, VocalSet

Finally, table 3.2.6 contains results for another musical concept more closely related to timbre than chroma: vocal technique. Performance is lower across the board compared to previous tasks, although the later layers of MERT achieve the highest micro average F1-score, and its middle layers achieve the highest macro average F1-score. Unlike the rest of the models, results for VGGish-AudioSet and NeuralFP have significant micro to macro average performance differences. These two models also exhibit the highest performance difference between the M1 and M5 classifiers of around 25 points, with MERT and MusiCNN-MSD exhibiting closer to 10 points difference. Significant performance drops are, again, observed in the presence of quiet and louder noise, while compression and gain reduction only marginally affect predictions.

Table 4: Music Auto-Tagging results for each embedding model, downstream model, and deformation scenario on the MagnaTagATune dataset.

		AUC-ROC				AUC-PR			
		C	D1	D2	D4	C	D1	D2	D4
VGGish-AudioSet	M1	83.7	77.4	61.7	81.8	31.2	24.3	11.6	29.5
	M2	87.6	79.4	67.1	85.5	37.1	25.9	14.0	33.4
	M3	88.1	78.0	66.7	86.2	38.0	24.6	14.5	34.4
	M4	87.8	79.5	65.9	85.5	37.5	26.8	13.5	33.6
	M5	88.6	78.4	67.1	86.6	39.2	25.9	13.9	35.3
EffNet-Discogs	M1	87.9	82.0	68.8	85.5	39.0	30.6	15.9	34.5
	M2	88.9	82.6	67.0	86.7	41.2	31.0	14.8	37.0
	M3	89.4	82.4	66.0	87.3	41.3	29.5	13.9	37.1
	M4	88.8	82.1	65.1	86.5	40.8	30.5	14.5	36.5
	M5	89.5	82.3	66.5	87.4	41.9	30.5	15.1	38.3
MusiCNN-MSD	M1	84.9	78.1	64.9	81.9	33.2	25.0	12.7	28.4
	M2	87.0	78.0	61.2	83.6	36.1	24.3	11.5	30.6
	M3	87.6	79.3	64.8	83.8	37.5	25.4	12.7	31.4
	M4	86.9	79.4	59.7	83.0	36.2	25.4	11.0	30.5
	M5	87.8	79.9	66.8	83.8	37.7	26.3	13.6	31.9
OpenL3	M1	89.2	87.1	78.8	89.0	40.9	36.4	25.2	40.6
	M2	89.6	87.4	78.2	89.5	41.3	36.7	23.1	41.2
	M3	90.4	87.8	77.8	90.2	42.3	36.6	22.3	42.0
	M4	90.1	87.6	77.7	89.9	41.7	36.7	22.6	41.4
	M5	90.4	87.0	75.8	90.1	42.7	36.7	21.2	42.3
NeuralFP	M1	73.0	70.2	60.9	71.8	19.6	17.7	11.5	18.2
	M2	80.9	73.1	61.7	76.9	26.8	19.6	12.3	23.0
	M3	83.4	73.0	58.3	78.8	29.7	19.0	11.1	24.7
	M4	81.4	73.0	60.1	77.4	27.2	19.7	11.9	23.3
	M5	84.5	73.6	59.7	79.7	31.5	19.9	11.5	26.0
CLMR	M1	89.1	85.3	75.6	88.7	41.7	33.8	21.3	40.9
	M2	89.9	85.8	76.2	89.5	43.0	35.0	21.6	42.0
	M3	89.9	86.0	76.4	89.4	42.6	34.5	21.4	41.8
	M4	89.7	85.9	76.7	89.4	42.3	34.3	21.8	41.5
	M5	89.8	85.9	76.4	89.4	42.5	34.5	21.4	41.7
MERT-v1-95m$^{0:3}$	M1	86.9	83.3	78.0	86.7	37.5	30.7	24.3	37.2
	M2	89.4	86.2	79.1	89.3	41.7	34.5	25.4	41.5
	M3	90.5	86.4	77.7	90.2	43.7	34.8	23.6	43.4
	M4	89.5	86.0	78.6	89.3	42.0	34.4	24.7	41.7
	M5	90.6	86.1	77.0	90.3	44.2	34.8	22.1	43.7
MERT-v1-95m$^{4:8}$	M1	89.3	87.0	82.4	89.3	41.2	36.1	**29.9**	41.1
	M2	91.0	89.0	82.5	91.0	45.4	40.0	29.4	45.3
	M3	91.2	89.2	**82.8**	91.2	45.6	40.5	28.5	45.6
	M4	90.9	89.0	**82.8**	90.9	45.1	39.8	29.3	45.0
	M5	91.3	**89.2**	81.7	**91.3**	46.1	40.6	27.5	46.0
MERT-v1-95m$^{9:12}$	M1	88.1	85.5	80.8	88.0	39.0	33.6	27.3	38.8
	M2	90.3	87.8	81.6	90.3	43.6	37.5	27.2	43.3
	M3	90.8	87.7	80.0	90.7	44.9	38.0	24.9	44.8
	M4	90.3	87.8	81.8	90.2	43.5	37.4	27.3	43.3
	M5	90.7	87.6	79.7	90.6	44.2	37.6	24.7	44.1
MERT-v1-95m^{6}	M1	89.6	87.4	82.2	89.5	41.6	36.7	29.6	41.5
	M2	91.2	89.1	82.4	91.1	45.7	40.5	28.9	45.8
	M3	**91.4**	89.4	81.6	**91.3**	**46.3**	**41.0**	28.1	**46.2**
	M4	91.0	88.9	81.8	91.0	44.8	39.9	29.1	44.7
	M5	91.1	89.0	81.3	91.0	44.8	40.2	27.4	44.7

Table 5: Global Key Estimation results for each embedding model, downstream model, and deformation scenario on the Beatport EDM dataset.

		micro avg weighted score					macro avg weighted score				
		C	D1	D2	D3	D4	C	D1	D2	D3	D4
	M1	11.6	13.3	12.6	11.6	13.1	7.1	8.2	8.3	7.4	7.7
	M2	15.6	15.6	15.4	15.6	14.8	8.3	8.3	8.0	8.3	8.0
VGGish-AudioSet	M3	14.5	14.8	16.0	14.8	14.7	8.8	7.2	8.6	9.1	8.7
	M4	13.8	15.4	16.2	14.8	13.4	7.3	7.7	8.3	8.0	8.0
	M5	13.6	15.6	13.2	11.8	12.3	9.3	8.5	7.3	8.3	8.3
	M1	13.8	12.3	5.9	12.0	14.0	12.1	12.4	5.3	10.6	11.4
	M2	17.2	15.5	12.5	15.0	15.9	9.4	9.0	8.3	8.3	8.8
EffNet-Discogs	M3	14.4	16.0	13.3	15.3	12.7	9.2	10.6	9.1	10.1	8.9
	M4	14.5	14.3	9.5	10.4	17.4	9.7	10.5	7.8	8.2	10.1
	M5	14.4	11.4	9.5	11.9	11.3	8.1	7.4	7.1	7.6	7.0
	M1	13.4	12.0	14.9	7.8	15.2	9.8	8.5	10.0	6.6	10.3
	M2	13.1	13.4	12.6	13.6	9.6	8.5	9.7	8.3	8.4	6.2
MusiCNN-MSD	M3	13.5	12.3	9.5	9.5	9.8	7.3	7.8	6.3	6.0	6.5
	M4	15.5	13.8	9.0	9.1	14.5	10.1	9.6	6.6	5.3	8.8
	M5	11.6	9.7	9.2	11.4	10.2	8.5	7.2	7.3	7.6	6.9
	M1	15.6	15.6	15.6	15.6	15.6	8.3	8.3	8.3	8.3	8.3
	M2	15.6	15.6	15.6	16.6	15.6	8.3	8.3	8.3	9.8	8.3
OpenL3	M3	13.3	15.2	12.8	14.8	13.7	6.8	7.6	7.2	7.4	6.9
	M4	15.6	15.6	15.6	15.6	15.6	8.3	8.3	8.3	8.3	8.3
	M5	14.5	15.2	13.5	14.8	14.5	7.3	7.6	7.6	7.4	7.3
	M1	32.7	29.4	15.2	37.8	30.8	24.3	22.1	8.7	31.1	24.2
	M2	35.5	30.5	19.4	38.4	28.6	28.1	23.3	12.6	31.8	23.7
NeuralFP	M3	35.5	31.4	20.2	36.7	34.8	29.1	25.3	13.3	31.2	28.3
	M4	33.8	29.3	24.5	32.0	30.0	26.8	23.1	17.9	29.1	25.0
	M5	33.2	33.5	20.5	35.0	28.0	26.1	27.6	15.2	28.2	27.1
	M1	15.9	15.5	9.0	15.7	14.5	10.9	7.9	8.2	10.3	9.9
	M2	16.4	16.0	17.6	16.4	16.8	9.0	8.9	10.1	9.0	9.3
CLMR	M3	13.9	15.9	15.4	11.3	13.1	7.8	8.1	8.6	6.4	7.8
	M4	13.5	14.6	17.0	9.2	12.5	8.5	8.5	9.0	6.9	8.4
	M5	14.6	17.4	13.2	21.8	11.9	11.1	11.7	12.5	15.0	9.1
	M1	52.6	45.0	37.5	41.2	53.6	46.1	36.8	31.3	37.0	46.9
	M2	51.0	44.5	39.5	**54.1**	52.6	43.2	36.2	30.0	46.9	44.2
MERT-v1-95m$^{0:3}$	M3	45.1	41.3	34.6	48.6	44.5	39.1	34.4	27.9	43.6	38.7
	M4	52.7	44.8	41.3	52.3	53.9	47.2	37.1	32.4	48.0	47.2
	M5	50.2	41.3	31.3	49.4	51.2	46.2	36.7	26.2	**49.2**	48.3
	M1	53.6	55.0	49.5	44.1	54.0	46.8	47.1	42.5	36.3	46.7
	M2	55.1	**56.7**	**50.8**	41.4	56.1	48.2	47.9	**42.7**	31.8	48.8
MERT-v1-95m$^{4:8}$	M3	**59.3**	53.4	50.2	46.6	**59.6**	**55.1**	47.6	41.2	41.8	**53.7**
	M4	58.5	55.5	45.5	41.3	58.3	50.2	48.1	35.6	33.1	50.1
	M5	54.0	55.9	50.2	45.8	54.8	47.1	**51.3**	42.2	37.3	48.4
	M1	52.2	52.9	43.3	46.6	51.4	45.1	46.8	36.1	39.0	44.4
	M2	56.3	52.7	46.7	50.6	56.3	52.1	48.4	40.8	44.1	52.1
MERT-v1-95m$^{9:12}$	M3	50.7	51.0	45.0	44.8	49.6	44.3	44.7	39.5	38.5	43.3
	M4	56.8	53.3	43.1	52.7	55.6	48.6	44.5	36.6	45.3	47.6
	M5	57.7	50.9	38.9	53.0	55.0	52.7	44.7	31.1	45.6	50.8
	M1	50.1	51.1	44.5	43.9	50.1	41.6	43.4	38.4	36.3	41.6
	M2	54.7	53.3	45.5	42.5	53.4	48.0	46.7	40.9	34.0	45.8
MERT-v1-95m^{6}	M3	52.7	46.6	37.1	41.0	52.1	46.0	38.8	31.3	31.2	45.3
	M4	56.1	54.6	44.3	40.1	56.3	48.7	46.4	37.2	31.1	49.0
	M5	55.2	53.7	48.4	45.7	54.5	49.3	47.4	40.3	39.8	47.7

Table 6: Instrument recognition results for each embedding model, downstream model, and deformation scenario on the TinySOL dataset.

		micro avg F1-score					macro avg F1-score				
		C	D1	D2	D3	D4	C	D1	D2	D3	D4
VGGish-AudioSet	M1	62.0	28.5	19.2	52.9	48.3	51.2	15.5	10.2	35.7	35.6
	M2	77.1	24.2	12.5	61.3	63.6	68.9	16.5	8.7	49.3	54.8
	M3	80.9	27.7	17.9	59.3	64.8	71.8	19.1	11.3	47.5	55.6
	M4	80.4	22.7	11.2	61.9	62.9	72.8	15.6	6.7	49.8	53.5
	M5	84.9	29.4	14.3	60.8	64.6	78.4	17.6	7.8	50.7	55.6
EffNet-Discogs	M1	74.4	32.3	13.7	62.4	53.1	62.8	22.8	8.0	46.9	41.3
	M2	81.8	33.0	13.2	68.9	57.0	72.9	23.7	10.9	56.5	49.3
	M3	80.6	36.8	14.8	67.4	52.6	71.9	28.5	12.5	54.6	43.3
	M4	84.7	30.8	12.9	73.7	56.0	77.5	24.4	11.6	63.1	46.1
	M5	83.7	35.6	12.4	70.6	59.8	75.4	28.1	10.8	60.3	52.0
MusiCNN-MSD	M1	60.8	45.2	19.8	57.0	44.8	52.2	33.5	10.2	46.8	34.0
	M2	67.5	49.3	21.8	61.3	51.5	58.7	38.2	14.9	51.7	39.4
	M3	68.2	50.5	21.5	63.2	51.7	60.8	37.9	12.0	53.8	41.3
	M4	67.0	49.5	20.3	62.0	50.7	57.8	36.3	13.3	50.5	39.3
	M5	70.4	52.4	23.2	64.6	54.0	60.7	38.6	13.9	53.0	42.2
OpenL3	M1	96.0	45.0	24.2	65.1	93.5	93.9	22.1	3.4	51.1	91.3
	M2	95.5	46.7	27.8	67.9	92.3	93.2	24.2	5.6	59.3	89.6
	M3	96.6	44.7	28.5	60.7	93.3	94.7	22.3	7.4	46.3	91.5
	M4	96.9	50.5	24.7	66.3	96.0	95.2	27.3	4.1	56.7	93.8
	M5	**97.8**	51.5	28.7	65.5	**96.4**	**96.8**	28.5	6.3	55.0	**95.8**
NeuralFP	M1	28.7	28.4	27.0	28.5	24.6	9.5	11.1	10.4	9.5	4.7
	M2	53.6	23.9	15.5	53.4	43.3	34.4	18.6	11.5	35.1	22.5
	M3	60.3	23.4	7.6	57.9	52.7	42.4	20.0	7.7	40.6	35.3
	M4	56.7	25.9	12.7	55.7	46.9	40.3	21.4	10.5	39.4	26.5
	M5	60.0	21.1	6.0	58.8	50.9	43.0	19.1	6.4	42.0	34.0
CLMR	M1	90.0	30.8	15.3	79.0	70.3	85.5	20.8	6.3	73.3	65.1
	M2	94.0	30.4	16.3	84.0	74.2	90.5	19.9	9.8	80.8	69.7
	M3	93.5	28.2	16.3	82.5	71.8	89.7	19.5	9.1	80.0	66.6
	M4	94.2	35.6	14.6	84.4	73.2	90.9	25.8	8.4	80.7	68.6
	M5	94.5	30.6	16.0	85.4	74.9	91.6	21.3	7.4	81.9	71.2
MERT-v1-95m$^{0:3}$	M1	95.9	62.4	24.6	82.5	93.8	93.6	48.9	11.8	75.7	91.6
	M2	96.6	60.3	21.8	83.0	95.9	94.5	49.9	11.0	78.2	93.9
	M3	96.7	57.6	21.3	80.9	94.8	94.9	52.5	13.4	74.4	92.4
	M4	96.2	65.5	24.7	84.2	95.2	94.3	53.3	11.8	78.9	92.4
	M5	95.7	57.7	26.6	81.8	95.0	92.7	51.4	14.1	75.3	92.1
MERT-v1-95m$^{4:8}$	M1	94.2	81.8	27.5	86.8	93.1	92.2	70.1	25.1	80.1	90.6
	M2	96.4	80.1	27.7	87.8	95.9	95.3	70.0	28.0	80.4	94.8
	M3	96.6	77.7	21.0	86.6	95.5	94.9	67.7	20.2	80.2	93.7
	M4	96.4	80.6	29.7	89.0	95.7	94.9	70.3	31.7	82.3	94.1
	M5	96.2	81.3	39.5	89.5	96.2	95.0	**73.5**	37.6	**85.0**	95.0
MERT-v1-95m$^{9:12}$	M1	91.9	67.7	17.5	83.3	91.8	88.5	59.7	12.4	73.4	88.6
	M2	95.7	69.8	15.6	84.4	94.3	93.8	63.4	13.4	75.2	92.6
	M3	97.1	69.4	13.7	86.9	95.9	95.8	63.8	10.3	81.7	94.3
	M4	95.2	67.9	15.5	82.6	93.8	93.1	63.0	13.8	71.9	91.9
	M5	96.2	67.9	14.8	86.4	94.7	94.4	62.5	13.0	78.8	93.1
MERT-v1-95m^{6}	M1	94.3	80.6	46.6	88.3	93.6	91.9	70.0	39.1	82.3	91.1
	M2	96.0	81.4	51.0	88.7	95.0	94.7	71.4	44.8	82.6	93.8
	M3	96.2	80.6	51.5	**90.2**	95.9	95.1	71.3	**46.4**	84.4	94.3
	M4	96.7	81.4	45.0	90.0	95.7	95.2	72.8	39.6	84.5	94.1
	M5	95.9	**82.1**	**47.3**	89.7	94.7	93.7	72.9	45.1	83.9	92.4

Table 7: Pitch Class classification results for each embedding model, downstream model, and deformation scenario on the TinySOL dataset.

		micro avg F1-score					macro avg F1-score				
		C	D1	D2	D3	D4	C	D1	D2	D3	D4
	M1	6.5	9.5	8.9	6.5	5.8	4.8	4.7	3.3	4.2	4.7
	M2	8.2	7.6	9.6	8.6	7.9	5.5	5.0	5.3	4.7	4.3
VGGish-AudioSet	M3	10.3	8.4	7.7	8.9	8.8	5.8	3.4	2.8	5.3	4.2
	M4	7.6	5.8	8.2	9.1	8.9	5.1	2.8	3.0	5.2	5.7
	M5	12.2	6.9	8.1	10.3	12.7	10.2	2.9	3.1	8.4	10.1
	M1	18.6	14.6	11.9	17.2	16.7	17.9	10.7	4.6	15.6	15.3
	M2	23.0	14.8	11.9	18.6	18.4	22.5	10.6	4.4	17.0	16.9
EffNet-Discogs	M3	19.6	13.1	10.5	16.0	16.7	18.5	9.6	4.1	13.7	14.8
	M4	22.5	14.8	11.9	19.1	19.8	21.9	10.4	5.5	17.5	19.0
	M5	25.3	15.5	12.9	20.8	20.8	24.4	11.5	8.2	19.1	20.6
	M1	8.1	8.6	9.6	7.9	7.0	7.7	7.2	4.3	7.2	5.9
	M2	14.9	13.9	9.1	13.7	11.5	13.7	11.9	5.4	12.5	10.3
MusiCNN-MSD	M3	17.2	14.4	8.4	17.0	15.5	15.7	12.9	4.3	15.7	13.8
	M4	16.2	16.2	11.0	16.3	12.4	15.9	14.1	6.4	15.7	11.7
	M5	19.6	15.1	9.5	19.1	15.8	18.9	14.7	5.8	18.4	15.2
	M1	13.1	8.9	6.5	12.0	13.2	13.0	5.8	2.2	8.4	13.3
	M2	10.7	14.8	10.5	11.9	10.5	9.1	10.4	3.1	10.6	8.6
OpenL3	M3	12.5	8.2	9.1	8.2	11.9	11.0	2.6	2.7	5.9	10.0
	M4	23.0	12.9	9.8	14.4	22.0	20.5	8.1	3.0	11.8	19.1
	M5	24.1	19.8	12.2	19.1	24.1	18.8	14.2	5.6	15.1	19.5
	M1	49.0	48.3	45.0	48.3	35.9	50.4	49.0	44.9	49.6	35.6
	M2	70.3	68.9	58.8	69.4	60.1	71.6	70.2	60.3	70.8	62.9
NeuralFP	M3	76.8	73.0	60.3	76.1	66.0	77.7	73.8	62.4	77.0	67.8
	M4	74.6	71.0	60.3	72.9	64.1	75.6	72.2	61.7	74.1	66.6
	M5	80.2	76.5	63.1	79.6	74.4	80.6	78.0	64.5	80.1	74.9
	M1	8.9	6.2	7.6	9.3	11.0	8.2	2.0	2.5	8.3	10.0
	M2	10.3	9.1	8.1	10.1	10.1	9.6	5.5	3.8	8.1	8.7
CLMR	M3	16.8	9.1	7.9	16.3	14.4	16.2	6.2	3.6	15.7	13.7
	M4	15.3	8.9	7.6	15.1	16.2	15.0	5.9	3.6	14.4	15.6
	M5	23.7	8.2	7.7	20.4	20.6	23.5	5.0	4.7	19.6	20.7
	M1	77.7	77.0	68.6	78.5	78.4	77.9	77.6	69.5	78.8	78.7
	M2	86.6	84.0	75.1	85.9	86.1	86.7	84.9	75.6	85.8	86.3
MERT-v1-95m$^{0:3}$	M3	90.5	86.1	77.8	90.0	89.7	90.5	86.7	78.1	89.9	89.7
	M4	88.5	81.8	73.2	88.8	88.3	88.5	82.9	73.4	88.6	88.3
	M5	91.9	79.9	66.2	91.8	90.4	91.9	81.6	67.9	91.7	90.4
	M1	85.4	82.6	77.8	84.9	84.0	85.5	83.1	79.6	84.9	84.0
	M2	93.6	90.5	83.8	92.8	93.6	93.6	90.6	84.2	92.7	93.6
MERT-v1-95m$^{4:8}$	M3	94.7	93.5	87.5	94.8	94.8	94.6	93.6	87.8	94.8	94.8
	M4	94.0	91.6	86.4	94.0	94.3	93.9	91.6	86.4	93.9	94.2
	M5	95.4	91.4	83.5	96.4	95.0	95.3	91.3	82.7	96.3	94.9
	M1	93.3	92.4	90.5	93.3	93.1	93.3	92.5	90.8	93.3	93.2
	M2	97.8	97.6	**97.3**	98.3	97.4	97.8	97.6	**97.3**	98.3	97.4
MERT-v1-95m$^{9:12}$	M3	98.3	98.1	96.6	98.1	97.9	98.3	98.1	96.5	98.1	97.9
	M4	97.9	**98.3**	96.4	**98.5**	**98.1**	98.0	**98.3**	96.5	**98.4**	**98.1**
	M5	**98.5**	97.9	96.6	**98.5**	97.9	**98.4**	98.0	96.7	**98.4**	97.9
	M1	83.7	79.7	72.5	83.5	83.7	83.9	80.1	74.3	83.5	83.7
	M2	93.3	90.0	82.0	93.3	93.1	93.2	90.1	82.2	93.2	93.1
MERT-v1-95m^{6}	M3	95.4	91.4	83.2	94.8	95.2	95.4	91.5	83.7	94.8	95.2
	M4	94.7	91.2	83.2	94.3	93.6	94.6	91.3	83.4	94.3	93.6
	M5	95.0	92.3	80.8	94.7	94.3	95.0	92.4	81.3	94.6	94.3

Table 8: Singer Identification results for each embedding model, downstream model, and deformation scenario on the VocalSet dataset.

		micro avg F1-score					macro avg F1-score				
		C	D1	D2	D3	D4	C	D1	D2	D3	D4
VGGish-AudioSet	M1	47.9	14.4	10.8	21.1	34.1	45.7	10.2	3.2	15.2	29.2
	M2	70.1	7.8	10.2	18.6	36.8	70.5	5.1	4.2	15.1	34.3
	M3	68.4	7.5	9.7	22.2	36.8	68.5	4.5	3.7	20.0	36.4
	M4	71.2	8.0	12.2	21.3	40.2	71.6	5.3	4.6	19.0	39.1
	M5	77.0	7.2	13.6	21.9	33.2	77.0	5.0	5.2	21.2	32.1
EffNet-Discogs	M1	68.1	17.5	10.0	46.8	35.7	67.8	15.0	6.0	42.9	36.8
	M2	71.7	16.6	10.8	46.8	32.1	70.8	12.8	5.0	42.3	31.1
	M3	73.4	13.3	7.2	46.8	31.9	72.7	9.4	2.3	42.3	31.6
	M4	75.3	15.5	7.8	49.9	32.4	74.5	14.1	3.4	45.1	31.8
	M5	76.7	12.7	6.6	48.5	37.1	76.0	9.4	1.8	44.8	35.5
MusiCNN-MSD	M1	45.7	20.2	9.1	42.7	17.7	43.1	17.8	5.4	39.3	14.2
	M2	54.6	23.3	5.5	48.8	23.0	52.2	22.9	2.4	46.0	17.8
	M3	58.2	22.7	6.6	51.8	23.8	56.8	19.8	3.3	49.2	22.2
	M4	57.3	22.2	7.5	49.9	19.9	56.5	21.2	2.3	47.4	18.3
	M5	64.8	28.3	6.1	54.8	25.2	63.6	24.8	1.8	52.7	21.6
OpenL3	M1	84.8	16.3	9.1	44.3	82.0	85.1	10.7	4.2	35.7	82.2
	M2	80.3	16.3	6.9	43.2	77.6	80.8	11.3	2.7	36.4	77.2
	M3	84.8	12.5	5.5	37.1	81.4	85.0	6.2	1.4	35.1	81.1
	M4	90.0	15.0	5.8	35.7	84.5	89.9	10.4	1.8	34.3	84.2
	M5	89.8	15.0	6.6	31.9	89.5	89.8	9.9	3.3	28.6	89.2
NeuralFP	M1	38.8	36.3	17.7	38.0	25.8	33.0	29.0	12.1	32.4	17.5
	M2	59.0	38.5	9.4	57.1	39.1	57.1	37.1	4.4	55.4	37.7
	M3	62.9	25.8	6.6	61.8	43.5	61.9	24.5	2.0	60.3	42.8
	M4	62.3	36.6	7.8	60.4	41.0	61.0	34.4	3.7	58.4	39.2
	M5	69.8	22.4	5.8	66.5	50.7	68.3	21.1	1.5	64.3	48.1
CLMR	M1	79.8	13.3	5.8	40.4	48.5	79.5	10.1	2.6	30.5	43.4
	M2	85.3	13.6	6.4	41.0	48.8	85.0	9.6	2.2	33.7	44.2
	M3	82.8	16.6	7.2	44.6	42.4	82.6	11.9	3.5	38.2	38.5
	M4	83.7	14.1	6.4	43.2	49.0	83.4	10.4	2.6	36.0	45.0
	M5	86.1	15.8	5.5	45.7	47.4	86.1	13.1	1.8	38.7	43.8
MERT-v1-95m$^{0:3}$	M1	92.2	32.7	5.0	64.5	92.8	92.5	22.9	0.5	62.9	93.2
	M2	94.7	31.0	6.1	64.3	93.9	95.0	20.0	1.9	61.0	94.0
	M3	94.5	26.0	8.6	63.2	94.2	94.7	16.1	2.3	59.3	94.5
	M4	94.2	28.0	6.6	64.0	93.6	94.5	19.2	2.2	60.5	93.9
	M5	91.4	23.8	6.1	59.3	92.2	91.6	14.9	1.9	55.6	92.4
MERT-v1-95m$^{4:8}$	M1	92.8	49.0	16.6	73.1	92.5	92.7	46.2	13.2	71.0	92.4
	M2	95.6	39.3	8.3	76.2	95.3	95.4	37.4	2.6	74.5	95.2
	M3	95.0	40.2	10.2	70.9	95.3	94.8	37.3	5.7	68.6	95.2
	M4	**97.0**	43.2	8.0	72.0	**96.7**	**96.9**	41.1	3.5	71.3	**96.8**
	M5	95.0	41.6	14.4	69.5	94.7	94.7	36.9	10.1	68.9	94.4
MERT-v1-95m$^{9:12}$	M1	93.6	**52.6**	19.9	**77.0**	93.6	93.8	**51.7**	**15.2**	77.1	93.7
	M2	96.7	50.4	19.7	73.7	96.4	96.5	48.4	14.7	72.8	96.4
	M3	96.4	46.5	18.8	71.2	95.8	96.4	44.4	12.2	70.0	95.8
	M4	95.6	49.6	**20.2**	72.9	95.3	95.8	47.4	15.2	71.5	95.4
	M5	95.3	42.7	14.1	68.7	95.0	95.2	41.4	9.9	68.0	95.1
MERT-v1-95m^{6}	M1	94.2	46.5	12.2	75.1	94.5	94.1	45.4	9.6	73.7	94.3
	M2	96.1	40.7	8.6	70.9	95.8	96.0	39.1	2.3	70.8	95.7
	M3	95.3	38.8	6.4	72.0	94.2	95.1	37.1	2.7	71.4	93.9
	M4	94.5	39.3	10.0	**77.0**	94.2	94.2	37.5	3.6	**75.6**	93.8
	M5	94.5	36.3	9.1	72.0	94.2	94.0	32.7	2.3	71.2	93.7

Table 9: Vocal Technique Identification results for each embedding model, downstream model, and deformation scenario on the VocalSet dataset.

		micro avg F1-score					macro avg F1-score				
		C	D1	D2	D3	D4	C	D1	D2	D3	D4
	M1	42.6	18.5	17.6	36.4	37.2	24.8	4.5	3.4	17.7	19.3
	M2	62.5	17.3	15.6	50.0	56.8	53.0	7.7	4.0	35.3	40.0
VGGish-AudioSet	M3	62.8	22.4	12.8	49.4	51.7	53.6	14.1	7.4	33.3	37.3
	M4	64.2	22.2	14.5	49.4	56.5	55.5	11.4	7.1	35.7	41.1
	M5	70.2	19.9	16.8	47.2	54.8	66.9	9.5	4.5	35.6	45.5
	M1	63.9	26.7	18.2	58.8	56.0	61.6	19.1	4.4	55.9	53.8
	M2	69.0	29.8	19.0	53.1	63.1	69.9	23.6	5.5	52.8	63.8
EffNet-Discogs	M3	69.9	30.1	20.2	52.0	64.5	70.7	25.9	6.5	54.0	63.5
	M4	67.9	31.3	19.6	52.8	59.9	69.5	23.9	6.2	52.7	58.9
	M5	69.0	29.3	20.2	50.3	63.4	69.7	24.3	6.8	49.5	63.1
	M1	62.5	49.4	29.8	62.5	46.9	55.1	40.1	15.5	54.9	44.9
	M2	68.5	36.4	21.9	68.2	50.3	63.9	31.8	9.0	63.8	49.4
MusiCNN-MSD	M3	66.8	31.8	25.6	65.1	51.7	64.3	32.3	15.4	63.1	53.2
	M4	71.3	31.5	24.7	68.8	51.7	69.2	29.9	10.4	67.1	50.7
	M5	71.3	33.2	23.0	71.3	53.7	67.5	33.3	14.6	68.3	52.2
	M1	61.6	37.5	24.7	49.7	61.9	60.9	35.0	14.1	43.5	61.7
	M2	63.9	32.1	21.9	48.6	64.2	64.6	30.2	13.4	40.8	63.9
OpenL3	M3	67.9	30.1	23.9	44.3	68.2	64.0	23.9	14.6	34.4	64.2
	M4	65.6	31.8	24.4	46.6	64.5	63.2	20.5	16.0	39.2	57.3
	M5	69.3	34.7	27.3	41.8	71.0	63.2	21.7	17.2	31.9	64.6
	M1	31.3	27.3	17.3	31.3	29.8	12.0	9.1	3.3	11.9	11.6
	M2	51.1	35.2	17.9	50.6	34.1	35.2	22.5	3.6	34.7	22.1
NeuralFP	M3	55.7	30.7	17.6	53.7	36.6	41.6	18.3	3.3	38.8	29.3
	M4	50.9	32.7	17.6	50.6	34.4	34.8	20.7	3.3	34.6	22.4
	M5	53.7	31.0	17.6	52.8	35.8	40.2	18.8	3.3	39.2	28.7
	M1	67.9	35.2	21.6	56.3	64.5	68.7	24.9	11.6	49.8	64.6
	M2	73.6	36.1	23.6	59.4	71.9	75.6	25.6	17.7	59.8	72.6
CLMR	M3	75.0	35.2	24.1	57.1	70.7	76.6	26.2	16.6	56.9	70.1
	M4	73.3	35.5	23.0	58.0	69.6	76.0	26.3	16.6	55.8	70.4
	M5	75.0	34.4	22.7	57.4	70.7	76.7	24.7	16.2	55.3	72.5
	M1	60.8	24.4	12.5	57.1	61.1	54.4	15.8	4.2	46.6	54.6
	M2	68.8	18.8	12.5	50.0	69.0	71.1	7.2	3.9	42.7	73.3
MERT-v1-95m$^{0:3}$	M3	71.6	19.9	12.5	51.7	71.3	76.0	8.1	3.8	47.3	75.7
	M4	68.5	17.6	13.1	50.3	68.8	73.1	7.3	5.0	42.1	73.1
	M5	71.0	20.2	11.6	49.4	70.7	72.6	7.9	2.3	42.6	71.9
	M1	68.8	43.5	**29.8**	65.3	68.8	63.2	32.8	20.7	60.5	63.2
	M2	76.4	44.0	29.0	70.7	76.7	79.4	37.3	**22.5**	71.4	79.7
MERT-v1-95m$^{4:8}$	M3	78.1	44.9	24.1	69.9	77.8	83.1	34.9	18.9	70.5	82.9
	M4	76.5	44.2	27.1	70.7	76.6	81.1	38.1	22.3	71.5	82.0
	M5	78.0	45.1	28.9	70.9	78.0	**83.2**	35.6	18.9	70.7	**83.2**
	M1	68.5	45.7	22.7	68.5	69.0	62.7	34.1	10.8	61.9	63.5
	M2	80.1	**49.4**	23.6	72.4	**80.7**	83.1	41.0	11.6	75.7	83.4
MERT-v1-95m$^{9:12}$	M3	79.0	48.6	27.0	69.3	78.4	81.2	43.6	19.0	72.4	80.7
	M4	80.1	48.6	23.3	**75.9**	80.1	82.6	41.0	11.0	**79.9**	82.6
	M5	**80.4**	52.0	23.9	69.3	**80.7**	82.9	**45.8**	13.4	72.0	83.1
	M1	69.9	45.7	27.8	65.1	69.9	73.3	34.9	19.5	62.9	73.3
	M2	79.8	47.4	24.4	73.9	79.3	82.8	45.1	19.5	76.2	82.3
MERT-v1-95m^{6}	M3	77.0	46.3	20.7	69.9	77.6	80.4	38.3	15.8	71.5	80.8
	M4	77.0	46.6	24.7	73.0	76.7	80.7	42.4	19.7	74.2	80.6
	M5	78.1	48.6	23.9	71.6	78.1	81.4	42.3	16.6	74.5	81.4

Chapter 4

Conclusions

4.1 Discussion

The results of the experiments that were conducted show that design factors of the representation learning pipeline aside from the deep representation itself can contribute substantially to the overall performance and suitability of a representation learning system. These systems are not necessarily as robust to audio perturbations as we might need them to be; in some cases, mild noise in the audio was able to nullify the prediction capability of systems that were otherwise performing well with clean audio (see Tables 3.2.3, 3.2.5, 3.2.6). Additionally, significant differences in performance across different downstream models were sometimes observed for the same representation. A particularly frequent case of this was with embeddings that were performing poorly with the linear SLP classifier but performing well with the 2-layer MLP classifiers, potentially hinting that the information needed for the task was not linearly separable in the embedding (see Tables 3.2.5, 3.2.6). Another interesting insight was the inability of most models to detect the pitch class in a single-note recording (see Table 3.2.4), probably hinting at limitations of their input representation or training paradigm.

More generally, this evaluation shows that there are still multiple components of the representation learning pipeline that we need to investigate and understand better, including the representations themselves. As the interest in universal audio representations grows, it's more relevant than ever before to understand and evaluate deep representations more holistically. This thesis project attempted to make a step in that direction through the development of an open-source toolkit for representation learning evaluation, and the release of a reproducible set of experiments utilizing deep representations.

4.2 Future work

As mentioned previously, the toolkit presented is aimed to be a living, open-source, collaborative project. Out of the list of features that we will look into implementing in the next months, a few related to the accompanying analysis tools feel like the most crucial. The interactive table utility mentioned in Sec. 2.1 as an extension to the existing `autotable` tool will likely become more of an indispensable requirement rather than just a discretionary feature as the evaluation experiments become more extensive. At the same, our usage of this toolkit as an aid for representation development is likely to increase. In this context, the interactive confusion matrix would offer substantial assistance, and so would the implementation of handcrafted features as baselines for comparison.

That said, given that the toolkit is now in a working and useful state, our focus will be on discussing with the community how software like this can help us think beyond benchmarks. Accordingly, we want to discuss how the toolkit can be helpful to others, what are other potential use cases, how we can improve the design, what we've gotten wrong so far, and what features we are missing.

List of Figures

List of Tables

Bibliography

[1] Mermelstein, P. Distance measures for speech recognition, psychological and instrumental. *Pattern recognition and artificial intelligence* **116**, 374–388 (1976).

[2] Davis, S. & Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **28**, 357–366 (1980).

[3] Eronen, A. Automatic musical instrument recognition. *Master's thesis, Tampere Univer- sity of Technology* (2001).

[4] Eronen, A. Comparison of features for musical instrument recognition. In *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)*, 19–22 (2001).

[5] Essid, S., Richard, G. & David, B. Musical instrument recognition on solo performances. In *2004 12th European Signal Processing Conference*, 1289–1292 (2004).

[6] Shepard, R. N. Geometrical approximations to the structure of musical pitch. *Psychological review* **89 4**, 305–33 (1982). URL `https://api.semanticscholar.org/CorpusID:41092670`.

[7] Fujishima, T. Realtime chord recognition of musical sound: a system using common lisp music. In *International Conference on Mathematics and Computing* (1999). URL `https://api.semanticscholar.org/CorpusID:38716842`.

[8] Sheh, A. & Ellis, D. P. Chord segmentation and recognition using em-trained hidden markov models (2003).

[9] Mauch, M. & Dixon, S. Simultaneous estimation of chords and musical context from audio. *IEEE Transactions on Audio, Speech, and Language Processing* **18**, 1280–1289 (2010).

[10] Hu, N., Dannenberg, R. & Tzanetakis, G. Polyphonic audio matching and alignment for music retrieval. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684)*, 185–188 (2003).

[11] Ewert, S., Muller, M. & Grosche, P. High resolution audio synchronization using chroma onset features. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 1869–1872 (2009).

[12] Li, T., Ogihara, M. & Li, Q. A comparative study on content-based music genre classification. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, 282–289 (Association for Computing Machinery, New York, NY, USA, 2003). URL `https://doi.org/10.1145/860435.860487`.

[13] Scaringella, N., Zoia, G. & Mlynek, D. Automatic genre classification of music content: a survey. *IEEE Signal Processing Magazine* **23**, 133–141 (2006).

[14] Kim, Y. *et al.* Music emotion recognition: A state of the art review. *Proceedings of the 11th International Society for Music Information Retrieval Conference* (2010).

[15] Bogdanov, D. *et al.* Essentia: An audio analysis library for music information retrieval. In *International Society for Music Information Retrieval Conference* (2013). URL `https://api.semanticscholar.org/CorpusID:11200511`.

[16] Essentia 2.1-beta6-dev documentation. URL `https://essentia.upf.edu/streaming_extractor_music.html`.

[17] Fiebrink, R. & Cook, P. The wekinator: A system for real-time, interactive machine learning in music. *Proceedings of The Eleventh International Society for Music Information Retrieval Conference* (2010).

[18] Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L. & Weinberger, K. (eds.) *Advances in Neural Information Processing Systems*, vol. 25 (Curran Associates, Inc., 2012). URL `https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[19] Hannun, A. *et al.* Deepspeech: Scaling up end-to-end speech recognition (2014).

[20] Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research. *IEEE Signal Processing Magazine* **36**, 128–137 (2019).

[21] Chen, W. *et al.* Data usage in mir: History & future recommendations. In *International Society for Music Information Retrieval Conference* (2019).

[22] Law, E., West, K., Mandel, M. I., Bay, M. & Downie, J. S. Evaluation of algorithms using games: The case of music tagging. In *International Society for Music Information Retrieval Conference* (2009). URL `https://api.semanticscholar.org/CorpusID:9788545`.

[23] Bertin-Mahieux, T., Ellis, D. P., Whitman, B. & Lamere, P. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval* (2011).

[24] Choi, K., Fazekas, G. & Sandler, M. B. Automatic tagging using deep convolutional neural networks. In *International Society for Music Information Retrieval Conference* (2016). URL `https://api.semanticscholar.org/CorpusID:582314`.

[25] Pons, J. & Serra, X. Musicnn: Pre-trained convolutional neural networks for music audio tagging. *International Society for Music Information Retrieval Conference, Late-Breaking Demo* (2017).

[26] Kim, T., Lee, J. & Nam, J. Sample-level cnn architectures for music auto-tagging using raw waveforms. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 366–370 (2018).

[27] Lee, J., Park, J., Kim, K. L. & Nam, J. Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification. *Applied Sciences* **8** (2018). URL `https://www.mdpi.com/2076-3417/8/1/150`.

[28] Lee, H., Largman, Y., Pham, P. & Ng, A. Y. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'09, 1096–1104 (Curran Associates Inc., Red Hook, NY, USA, 2009).

[29] Hamel, P. & Eck, D. Learning features from music audio with deep belief networks. In *International Society for Music Information Retrieval Conference* (2010).

[30] Dieleman, S., Brakel, P. & Schrauwen, B. Audio-based music classification with a pretrained convolutional network. In *International Society for Music Information Retrieval Conference* (2011).

[31] Lau, M. M. & Lim, K. H. Investigation of activation functions in deep belief network. In *2017 2nd International Conference on Control and Robotics Engineering (ICCRE)*, 201–206 (2017).

[32] van den Oord, A., Dieleman, S. & Schrauwen, B. Transfer learning by supervised pre-training for audio-based music classification. In *International Society for Music Information Retrieval Conference* (2014).

[33] Liang, D., Zhan, M. & Ellis, D. P. W. Content-aware collaborative music recommendation using pre-trained neural networks. In *International Society for Music Information Retrieval Conference* (2015).

[34] Choi, K., Fazekas, G., Sandler, M. B. & Cho, K. Transfer learning for music classification and regression tasks. In *International Society for Music Information Retrieval Conference* (2017).

[35] Hershey, S. *et al.* Cnn architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2017). URL `https://arxiv.org/abs/1609.09430`.

[36] Castellon, R., Donahue, C. & Liang, P. Codified audio language modeling learns useful representations for music information retrieval. In *International Society for Music Information Retrieval Conference* (2021).

[37] Won, M., Ferraro, A., Bogdanov, D. & Serra, X. Evaluation of cnn-based automatic music tagging models. In *Sound and Music Computing Conference* (2020).

[38] Music information retrieval evaluation exchange (mirex) wiki. URL `https://www.music-ir.org/mirex/wiki/MIREX_HOME`.

[39] Stephen Downie, J. *et al.* Ten years of mirex: Reflections, challenges and opportunities. 657–662 (2014). 15th International Society for Music Information Retrieval Conference.

[40] Hinton, G. E., Osindero, S. & Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural Comput.* **18**, 1527–1554 (2006). URL `https://doi.org/10.1162/neco.2006.18.7.1527`.

[41] Bengio, Y., Lamblin, P., Popovici, D. & Larochelle. Greedy layer-wise training of deep networks. In Schölkopf, B., Platt, J. & Hoffman, T. (eds.) *Advances in Neural Information Processing Systems*, vol. 19 (MIT Press, 2006). URL `https://proceedings.neurips.cc/paper_files/paper/2006/file/5da713a690c0`.

[42] Tzanetakis, G. & Cook, P. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing* **10**, 293–302 (2002).

[43] Coates, A. & Ng, A. Learning feature representations with k-means. In *Neural Networks* (2012).

[44] Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. 1–14 (Computational and Biological Learning Society, 2015).

[45] Won, M., Chun, S., Nieto, O. & Serrc, X. Data-driven harmonic filters for audio representation learning. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 536–540 (2020).

[46] Park, J., Lee, J., Park, J., Ha, J.-W. & Nam, J. Representation learning of music using artist labels. In *International Society for Music Information Retrieval Conference* (2017).

[47] Lee, J., Park, J. & Nam, J. Representation learning of music using artist, album, and track information. *ArXiv* **abs/1906.11783** (2019).

[48] Favory, X., Drossos, K., Virtanen, T. & Serra, X. Coala: Co-aligned autoencoders for learning semantically enriched audio representations (2020). 2006.08386.

[49] Favory, X., Drossos, K., Virtanen, T. & Serra, X. Learning contextual tag embeddings for cross-modal alignment of audio and tags. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 596–600 (2021).

[50] Alonso-Jiménez, P., Serra, X. & Bogdanov, D. Music representation learning based on editorial metadata from discogs. In *International Society for Music Information Retrieval Conference* (2022).

[51] Saeed, A., Grangier, D. & Zeghidour, N. Contrastive learning of general-purpose audio representations. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3875–3879 (2021).

[52] Tan, M. & Le, Q. EfficientNet: Rethinking model scaling for convolutional neural networks. In Chaudhuri, K. & Salakhutdinov, R. (eds.) *Proceedings of the 36th International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, 6105–6114 (PMLR, 2019). URL `https://proceedings.mlr.press/v97/tan19a.html`.

[53] Ferraro, A., Favory, X., Drossos, K., Kim, Y. & Bogdanov, D. Enriched music representations with multiple cross-modal contrastive learning. *IEEE Signal Processing Letters* **28**, 733–737 (2021).

[54] Alonso-Jiménez, P. *et al.* Pre-training strategies using contrastive learning and playlist information for music classification and similarity. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5 (2023).

[55] Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. E. A simple framework for contrastive learning of visual representations. *CoRR* **abs/2002.05709** (2020). URL `https://arxiv.org/abs/2002.05709`. `2002.05709`.

[56] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition (2015). `1512.03385`.

[57] Manco, I., Benetos, E., Quinton, E. & Fazekas, G. Learning music audio representations via weak language supervision. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 456–460 (2021).

[58] Lu, J., Batra, D., Parikh, D. & Lee, S. *ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks* (Curran Associates Inc., Red Hook, NY, USA, 2019).

[59] Manco, I., Benetos, E., Quinton, E. & Fazekas, G. Contrastive audio-language learning for music. In *International Society for Music Information Retrieval Conference* (2022).

[60] Cramer, A. L., Wu, H.-H., Salamon, J. & Bello, J. P. Look, listen, and learn more: Design choices for deep audio embeddings. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3852–3856 (2019).

[61] Arandjelović, R. & Zisserman, A. Look, listen and learn. *2017 IEEE International Conference on Computer Vision (ICCV)* 609–617 (2017). URL `https://api.semanticscholar.org/CorpusID:10769575`.

[62] Gemmeke, J. F. *et al.* Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 776–780 (2017).

[63] Niizumi, D., Takeuchi, D., Ohishi, Y., Harada, N. & Kashino, K. Byol for audio: Self-supervised learning for general-purpose audio representation. 1–8 (2021).

[64] Spijkervet, J. & Burgoyne, J. A. Contrastive learning of musical representations. In *International Society for Music Information Retrieval Conference* (2021).

[65] Chang, S. *et al.* Neural audio fingerprint for high-specific audio retrieval based on contrastive learning. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3025–3029 (2020).

[66] Defferrard, M., Benzi, K., Vandergheynst, P. & Bresson, X. Fma: A dataset for music analysis. In *International Society for Music Information Retrieval Conference* (2016).

[67] Devlin, J., Chang, M., Lee, K. & Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C. & Solorio, T. (eds.) *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 4171–4186 (Association for Computational Linguistics, 2019). URL `https://doi.org/10.18653/v1/n19-1423`.

[68] Zhao, Y. & Guo, J. Musicoder: A universal music-acoustic encoder based on transformer. In *MultiMedia Modeling: 27th International Conference, MMM 2021, Prague, Czech Republic, June 22–24, 2021, Proceedings, Part I*, 417–429 (Springer-Verlag, Berlin, Heidelberg, 2021). URL `https://doi.org/10.1007/978-3-030-67832-6_34`.

[69] Vaswani, A. *et al.* Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 6000–6010 (Curran Associates Inc., Red Hook, NY, USA, 2017).

[70] Li, Y. *et al.* Mert: Acoustic music understanding model with large-scale self-supervised training (2023). `2306.00107`.

[71] Dhariwal, P. *et al.* Jukebox: A generative model for music (2020). `2005.00341`.

[72] Wang, L. *et al.* Towards learning universal audio representations. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4593–4597 (2022).

[73] Deepmind. Deepmind/slowfast_nfnets. URL `https://github.com/deepmind/slowfast_nfnets`.

[74] Nttcslab. Nttcslab/eval-audio-repr: EVAR evaluation package for audio representations. URL `https://github.com/nttcslab/eval-audio-repr`.

[75] HEAR: Holistic evaluation of audio representations (neurips 2021 competition). vol. 166 of *Proceedings of Machine Learning Research* (PMLR).

[76] Holistic evaluation of audio representations. URL `https://hearbenchmark.com/`.

[77] Yuan, R. *et al.* Marble: Music audio representation benchmark for universal evaluation (2023). `2306.10548`.

[78] Bittner, R. M. *et al.* mirdata: Software for reproducible usage of datasets. In *International Society for Music Information Retrieval Conference* (2019). URL `https://api.semanticscholar.org/CorpusID:208334929`.

[79] Raffel, C. *et al.* Mir_eval: A transparent implementation of common mir metrics. In *International Society for Music Information Retrieval Conference* (2014).

[80] Alonso-Jiménez, P., Bogdanov, D., Pons, J. & Serra, X. Tensorflow audio models in Essentia. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020).

[81] Jordal, I. *et al.* iver56/audiomentations .

[82] Nikšić, H. Foundation, Free Software. URL `https://www.gnu.org/software/wget/`.

[83] Cella, C.-E. *et al.* Orchideasol: a dataset of extended instrumental techniques for computer-aided orchestration. *ArXiv* **abs/2007.00763** (2020). URL `https://api.semanticscholar.org/CorpusID:220302614`.

[84] Faraldo, Á. Beatport EDM Key Dataset (2018). 10.5281/zenodo.1154586.

[85] Faraldo, Á. *Tonality Estimation in Electronic Dance Music: A Computational and Musically Informed Examination.* Ph.D. thesis (2017).

[86] Bogdanov, D., Won, M., Tovstogan, P., Porter, A. & Serra, X. The mtg-jamendo dataset for automatic music tagging. In *International Conference on Machine Learning* (2019). URL `https://api.semanticscholar.org/CorpusID:196187495`.

[87] Wilkins, J., Seetharaman, P., Wahl, A. & Pardo, B. Vocalset: A singing voice dataset. In *International Society for Music Information Retrieval Conference* (2018).

[88] McCallum, M. C., Korzeniowski, F., Oramas, S., Gouyon, F. & Ehmann, A. F. Supervised and unsupervised learning of audio representations for music understanding (2022).

[89] Abadi, M. *et al.* TensorFlow: Large-scale machine learning on heterogeneous systems (2015). URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[90] URL `https://streamlit.io/`. Streamlit • A faster way to build and share data apps.

[91] URL `https://wandb.ai/site/weave`. A visual, interactive, and composable way of building ML data apps.

[92] Discogs. URL `https://www.discogs.com/`.

[93] The official YAML website (2000). URL `https://www.yaml.org`.

Appendix A

Experiment Configuration File

The configuration file in YAML format [93] for the replication of this thesis project's evaluation experiments is present below. As future versions of the toolkit might contain breaking changes, it is recommended to use the original release version and the stated versions of the toolkit's dependencies. For the evaluation experiments of this thesis, we decided to use the same deformation scenarios, embedding models, and downstream models for each task and dataset. Therefore, for brevity, those three components will be presented in separate code blocks, and a reference to them will be made from the main code block of the configuration file.

```yaml
# CONFIGURATION.yml
experiments:
  - task:
      name: autotagging
      type: multilabel_classification
      embedding_aggregation: mean
    datasets:
      - name: magnatagatune
        type: custom
        dir: data/magnatagatune/
        split_type: single
    deformations:
        DEFORMATIONS
    embedding_models:
        EMBEDDING_MODELS
    downstream_models:
        DOWNSTREAM_MODELS
  - task:
      name: key_estimation
      type: multiclass_classification
      embedding_aggregation: mean
    datasets:
      - name: beatport_key
        type: mirdata
        dir: data/beatport_key/
        split_type: single
    deformations:
        DEFORMATIONS
    embedding_models:
        EMBEDDING_MODELS
    downstream_models:
        DOWNSTREAM_MODELS
  - task:
      name: instrument_classification
      type: multiclass_classification
      embedding_aggregation: mean
    datasets:
      - name: tinysol
        type: mirdata
        dir: data/tinysol/
        split_type: single
    deformations:
        DEFORMATIONS
    embedding_models:
        EMBEDDING_MODELS
    downstream_models:
        DOWNSTREAM_MODELS
  - task:
      name: pitch_class_classification
      type: multiclass_classification
      embedding_aggregation: mean
    datasets:
      - name: tinysol
        type: mirdata
        dir: data/tinysol/
        split_type: single
    deformations:
        DEFORMATIONS
    embedding_models:
        EMBEDDING_MODELS
    downstream_models:
        DOWNSTREAM_MODELS
  - task:
      name: singer_identification
      type: multiclass_classification
      embedding_aggregation: mean
    datasets:
      - name: vocalset
        type: custom
        dir: data/vocalset/
        split_type: single
    deformations:
        DEFORMATIONS
    embedding_models:
        EMBEDDING_MODELS
    downstream_models:
        DOWNSTREAM_MODELS
  - task:
      name: technique_identification
      type: multiclass_classification
      embedding_aggregation: mean
    datasets:
      - name: vocalset
        type: custom
        dir: data/vocalset/
        split_type: single
    deformations:
        DEFORMATIONS
    embedding_models:
        EMBEDDING_MODELS
    downstream_models:
        DOWNSTREAM_MODELS
```

```
# DEFORMATIONS.yml
- - type: AddGaussianSNR
  params:
    min_snr_in_db: 15
    max_snr_in_db: 15
    p: 1
- - type: AddGaussianSNR
  params:
    min_snr_in_db: 0
    max_snr_in_db: 0
    p: 1
- - type: Mp3Compression
  params:
    min_bitrate: 32
    max_bitrate: 32
    p: 1
- - type: Gain
```

```
  params:
    min_gain_in_db: -12
    max_gain_in_db: -12
    p: 1
```

```
# EMBEDDING_MODELS.yml
- vggish-audioset
- effnet-discogs
- msd-musicnn
- openl3
- neuralfp
- clmr-v2
- mert-v1-95m-0-1-2-3
- mert-v1-95m-4-5-6-7-8
- mert-v1-95m-9-10-11-12
- mert-v1-95m-6
```

```
# DOWNSTREAM_MODELS.yml
- type: classifier
  emb_dim_reduction: False
  emb_shape: infer
  hidden_units: []
  # depending on task
  output_activation: softmax / sigmoid
  weight_decay: 1.0e-5
  optimizer: adam
  learning_rate: 1.0e-3
  batch_size: 100
  epochs: 100
  patience: 10
  train_sampling: random
  # depending on task
  save_criterion: val_categorical_accuracy / val_auc
- type: classifier
  emb_dim_reduction: False
  emb_shape: infer
  hidden_units: [infer]
  # depending on task
  output_activation: softmax / sigmoid
  weight_decay: 1.0e-5
  optimizer: adam
  learning_rate: 1.0e-3
  batch_size: 100
  epochs: 100
  patience: 10
  train_sampling: random
  # depending on task
  save_criterion: val_categorical_accuracy / val_auc
- type: classifier
  emb_dim_reduction: False
  emb_shape: infer
  hidden_units: [infer, infer
  # depending on task
  output_activation: softmax / sigmoid
  weight_decay: 1.0e-5
  optimizer: adam
  learning_rate: 1.0e-3
  batch_size: 100
  epochs: 100
  patience: 10
  train_sampling: random
  # depending on task
  save_criterion: val_categorical_accuracy / val_auc
- type: classifier
  emb_dim_reduction: False
  emb_shape: infer
  hidden_units: [128
  # depending on task
  output_activation: softmax / sigmoid
  weight_decay: 1.0e-5
  optimizer: adam
  learning_rate: 1.0e-3
  batch_size: 277
  epochs: 100
  patience: 10
  train_sampling: random
  # depending on task
  save_criterion: val_categorical_accuracy / val_auc
- type: classifier
  emb_dim_reduction: False
  emb_shape: infer
  hidden_units: [256, 128]
  # depending on task
  output_activation: softmax / sigmoid
  weight_decay: 1.0e-5
  optimizer: adam
  learning_rate: 1.0e-3
  batch_size: 100
  epochs: 100
  patience: 10
  train_sampling: random
  # depending on task
  save_criterion: val_categorical_accuracy / val_auc
```