One Deep Music Representation to Rule Them All?
A comparative analysis of different representation learning strategies

Jaehun Kim · Julián Urbano · Cynthia C. S. Liem · Alan Hanjalic

Received: date / Accepted: date

Abstract Inspired by the success of deploying deep learning in the fields of Computer Vision and Natural Language Processing, this learning paradigm has also found its way into the field of Music Information Retrieval. In order to benefit from deep learning in an effective, but also efficient manner, deep transfer learning has become a common approach. In this approach, it is possible to reuse the output of a pre-trained neural network as the basis for a new learning task. The underlying hypothesis is that if the initial and new learning tasks show commonalities and are applied to the same type of input data (e.g. music audio), the generated deep representation of the data is also informative for the new task. Since, however, most of the networks used to generate deep representations are trained using a single initial learning source, the validity of the above hypothesis is questionable for an arbitrary future task. In this paper, we present the results of our investigation of what the most important factor to generate deep representations for the data and learning tasks in the music domain. We conducted this investigation via an extensive empirical study that involves multiple learning sources, as well as multiple deep learning architectures with varying levels of information sharing between sources, in order to learn music representations. We then validate these representations considering multiple target datasets for evaluation. The results of our experiments yield several insights on how to approach the design of methods for learning widely deployable deep data representations in the music domain.

Keywords Representation Learning · Music Information Retrieval · Multi-Task Learning
1 Introduction

In the Music Information Retrieval (MIR) field, many research problems of interest involve the automatic description of properties of musical signals, employing concepts that are understood by humans. For this, tasks are derived that can be solved by automated systems. In such cases, algorithmic processes are employed to map raw music audio information to humanly understood descriptors (e.g. genre labels or descriptive tags). To achieve this, historically, the raw audio would first be transformed into a representation based on hand-crafted features, which are engineered by humans to reflect dedicated semantic signal properties. The feature representation would then serve as input to various statistical or Machine Learning (ML) approaches [1].

The framing as described above can generally be applied to many applied ML problems: complex real-world problems are abstracted into a relatively simpler form, by establishing tasks that can be computationally addressed by automatic systems. In many cases, the task involves making a prediction based on a certain observation. For this, modern ML methodologies can be employed, that automatically can infer the logic for the prediction directly from (a numeric representation of) the given data, by optimizing an objective function defined for the given task.

However, music is a multimodal phenomenon, that can be described in many parallel ways, ranging from objective descriptors to subjective preference. Given this broad spectrum of descriptions and interpretations, it is non-trivial to establish a proper general music representation in the context of a single universal task, in which a single model would learn every possible musical aspect. A more promising strategy is to employ Multi-Task Learning (MTL), in which a single learning framework hosts multiple tasks at once, allowing for models to perform better by sharing commonalities between involved tasks [2]. MTL has been successfully used in a range of applied ML works [3-10], also including the music domain [11, 12].

Following successes in the fields of Computer Vision (CV) and Natural Language Processing (NLP), deep learning approaches have recently also gained increasing interest in the MIR field, in which case deep representations of music audio data are directly learned from the data, rather than being hand-crafted. Many works employing such approaches reported considerable performance improvements in various music analysis, indexing and classification tasks [13-20].

In many deep learning applications, rather than training a complete network from scratch, pre-trained networks are commonly used to generate deep representations, which can be either directly adopted or further adapted for the current task at hand. In CV and NLP, (parts of) certain pre-trained networks [21-24] have now been adopted and adapted in a very large number of works. These ‘standard’ deep representations have typically been obtained by training a network for a single learning task, such as visual object recognition, employing large amounts of training data. The hypothesis on why these representations are effective in a broader spectrum of tasks than they originally were trained for, is that deep transfer learning (DTL) is happening: information initially picked up by the network is beneficial also for new learning tasks performed on the same type of raw input data. Clearly, the validity of this hypothesis is linked to the extent to which the new task can rely on similar data characteristics as the task on which the pre-trained network was originally trained.

Although a number of works deployed DTL for various learning tasks in the music domain [25-28], to our knowledge, however, transfer learning and the employment of pre-trained networks are not as standard in the MIR domain as in the CV domain. Again, this may be due to the broad and partially subjective range and nature of possible music descriptions. Following the considerations above, it may then be useful to combine deep transfer learning with multi-task learning.
One Deep Music Representation to Rule Them All?

Fig. 1: Simplified illustration of the conceptual difference between traditional deep transfer learning (DTL) based on a single learning task (above) and multi-task based deep transfer learning (MTDTL) (below). The same color used for a learning and an target task indicates that the tasks have commonalities, which implies that the learned representation is likely to be informative for the target task. At the same time, this representation may not be that informative to another future task, leading to a low transfer learning performance. The hypothesis behind MTDTL is that relying on more learning tasks increases robustness of the learned representation and its usability for a broader set of target tasks.

Indeed, in order to increase robustness to a larger scope of new learning tasks and datasets, the concept of MTL also has been applied in training deep networks for representation learning, both in the music domain \[11\] \[12\] and in general \[3\] p. 2. As the model learns several tasks and datasets in parallel, it may pick up commonalities among them. As a consequence, the expectation is that a network learned with MTL will yield robust performance across different tasks, by transferring shared knowledge \[2\] \[3\]. A simple illustration of the conceptual difference between traditional DTL and deep transfer learning based on MTL (further referred to as multi-task based deep transfer learning (MTDTL)) is shown in Fig. 1.

The mission of this paper is to investigate the effect of conditions around the setup of MT-DTL, which are important to yield effective deep music representations. Here, we understand an ‘effective’ representation to be a representation that is suitable for a wide range of new tasks and datasets. Ultimately, we aim for providing a methodological framework to systematically obtain and evaluate such transferable representations. We pursue this mission by exploring the effectiveness of MTDTL and traditional DTL, as well as concatenations of multiple deep representations, obtained by networks that were independently trained on separate single learning tasks. We consider these representations for multiple choices of learning tasks and considering multiple target datasets.

Our work will address the following research questions:

– **RQ1**: Given a set of learning sources that can be used to train a network, what is the influence of the number and type of the sources on the effectiveness of the learned deep representation?
– **RQ2**: How do various degrees of information sharing in the deep architecture affect the ultimate success of a learned deep representation?
By answering the RQ1 we arrive at an understanding of important factors regarding the composition of a set of learning tasks and datasets (which in the remainder of this work will be denoted as learning sources) to achieve an effective deep music representation, specifically on the number and nature of learning sources. The answer to RQ2 provides insight in how to choose the optimal multi-task network architecture under a MTDTL context. For example, in MTL, multiple sources are considered under a joint learning scheme, that partially shares inferences obtained from different learning sources in the learning pipeline. In MTL applications using deep neural networks, this means that certain layers will be shared between all sources, while at other stages, the architecture will ‘branch’ out into source-specific layers \(^2\)\(^5\)\(^8\)\(^12\)\(^29\). However, investigation is still needed on where in the layered architecture branching should ideally happen—if a branching strategy would turn out strategic in the first place.

To reach the aforementioned answers, it is necessary to conduct a systematic assessment to examine relevant factors. For RQ1, we investigate different numbers and combinations of learning sources. For RQ2, we study different architectural strategies. However, we wish to ultimately investigate effectiveness of the representation with respect to new, target learning tasks and datasets (which in the remainder of this paper will be denoted by target datasets). While this may cause combinatorial explosion with respect to possible experimental configurations, we will make strategic choices in the design and evaluation procedure of the various representation learning strategies.

The scientific contribution of this work can be summarized as follows:

- We provide insight into the effectiveness of a various deep representation learning strategies under the multi-task learning context.
- We offer in-depth insight into ways to evaluate desired properties of a deep representation learning procedure.
- We propose and release several pre-trained music representation networks, based on different learning strategies for multiple semantic learning sources.

The rest of this work is presented as following: a formalization of this problem, as well as the global outline of how learning will be performed based on different learning tasks from different sources, will be presented in Section 2. Detailed specifications of the deep architectures we considered for the learning procedure will be discussed in Section 3. Our strategy to evaluate the effectiveness of different representation network variants by employing various target datasets will be the focus of Section 4. Experimental results will be discussed in Section 5 after which general conclusions will be presented in Section 6.

2 Framework for Deep Representation Learning

In this section, we formally define the deep representation learning problem. As Fig. 2 illustrates, any domain-specific MTDTL problem can be abstracted into a formal task, which is instantiated by a specific dataset with specific observations and labels. Multiple tasks and datasets are involved to emphasize different aspects of the input data, such that the learned representation is more adaptable to different future tasks. The learning part of this scheme can be understood as the MTL phase, which is introduced in Section 2.1. Subsequently in Section 2.2 we discuss learning sources involved in this work, which consist of various tasks and datasets to allow investigating their effects on the transfer learning. Further, we introduce the label preprocessing procedure that is applied in this work in Section 2.3 ensuring that the learning sources are more regularized, such that their comparative analysis is clearer.
Fig. 2: Schematic overview of what this work investigates. The upper scheme illustrates a general problem solving framework in which multi-task transfer learning is employed. The tasks $t \in \{t_0, t_1, \cdots, t_M\}$ are derived from a certain problem domain, which are instantiated by datasets, that often are represented as sample pairs of observations and corresponding labels $(X_t, y_t)$. Sometimes, the original dataset is processed further into simpler representation forms $(X_t, z_t)$, to filter out undesirable information and noise. Once a model or system $f_t(X_t)$ has learned the necessary mappings within the learning sources, this knowledge can be transferred to another set of target datasets, leveraging commonalities already obtained by the pre-training. Below the general framework, we show a concrete example, in which the broad MIR problem domain is abstracted into various sub-problems with corresponding tasks and datasets.
2.1 Problem Definition

A machine learning problem, focused on solving a specific task $t$, can be formulated as a minimization problem, in which a model function $f_t$ must be learned that minimizes a loss function $L$ for given dataset $D_t = \{(x_t^{(i)}, y_t^{(i)}) \mid i \in \{1, \cdots, I\}\}$, comparing the model’s predictions given by the input $x_t$ and actual task-specific learning labels $y_t$. This can be formulated using the following expression:

$$
\min_{\theta} E_{D_t} L(y_t, f_t(x_t; \theta)) \quad (1)
$$

where $x_t \in \mathbb{R}^d$ is, traditionally, a hand-crafted $d$-dimensional feature vector and $\theta$ is a set of model parameters of $f$.

When deep learning is employed, the model function $f$ denotes a learnable network. Typically, the network model $f$ is learned in an end-to-end fashion, from raw data at the input to the learning label. In the speech and music field, however, using true end-to-end learning is still not a common practice. Instead, raw data is typically transformed first, before serving as network input. More specifically, in the music domain, common input to function $f$ would be $X \in \mathbb{R}^{c \times n \times b}$, replacing the originally hand-crafted feature vector $x \in \mathbb{R}^d$ from (1) by a time-frequency representation of the observed music data, usually obtained through the Short-Time Fourier Transform (STFT), with potential additional filter bank applications (e.g. mel-filter bank). The dimensions $c$, $n$, $b$ indicate channels of the audio signal, time steps, and frequency bins respectively.

If such a network still is trained for a specific single machine learning task $t$, we can now reformulate (1) as follows:

$$
\min_{\theta} E_{D_t} L(y_t, f_t(X_t; \theta)) \quad (2)
$$

In MTL, in the process of learning the network model $f$, different tasks will need to be solved in parallel. In case of deep neural networks, this is usually realized by having a network in which lower layers are shared for all tasks, but upper layers are task-specific. Given $m$ different tasks $t$, each having the learning label $y_t$, we can formulate the learning objective of the neural network in a MTL scenario as follows:

$$
\min_{\theta^{*}, \theta} \mathbb{E}_{t \in T} E_{D_t} L(y_t, f_t(X_t; \theta_t^{*}, \theta^{s})) \quad (3)
$$

Here, $T = \{t_1, t_2, \ldots, t_m\}$ is a given set of tasks to be learned and $\theta^{*} = \{\theta_1, \theta_2, \ldots, \theta^m\}$ indicates a set of model parameters $\theta^t$ with respect to each task. Since the deep architecture initially shares lower layers and branches out to task-specific upper layers, the parameters of shared layers and task-specific layers are referred to separately as $\theta^t$ and $\theta^s$. Updates for all parameters can be achieved through standard back-propagation. Further specifics on network architectures and training configurations will be given in Section 3.

Given the formalizations above, the first step in our framework is to select a suitable set $T$ of learning tasks. These tasks can be seen as multiple concurrent descriptions or transformations of the same input fragment of musical audio: each will reflect certain semantic aspects of the music. However, unlike the approach in a typical MTL scheme, solving multiple specific learning tasks is actually not our main goal; instead, we wish to learn an effective representation that captures as many semantically important factors in the low-level music representation. Thus, rather than using learning labels $y_t$, our representation learning process will employ reduced learning labels $z_t$, which capture a reduced set of semantic factors from $y_t$. We then can reformulate (3) as follows:
where $z_t \in \mathbb{R}^k$ is a $k$-dimensional reduced learning labels for a specific task $t$. Each $z_t$ will be obtained through task-specific factor extraction methods, as described in Section 2.3.

2.2 Learning Sources

In the MTDTL context, a training dataset can be seen as the ‘source’ to learn the representation, which will be further transferred to the future ‘target’ dataset. Different learning sources of different nature can be imagined, that can be globally categorized as Algorithm or Annotation. As for the Algorithm category, by employing traditional feature extraction or representation transformation algorithms, we will be able to automatically extract semantically interesting aspects from input data. As for the Annotation category, these include different types of label annotations of the input data by humans.

The dataset used as resource for our learning experiments is the Million Song Dataset (MSD)\cite{30}. In its original form, it contains metadata and precomputed features for a million songs, with several associated data resources, e.g. considering Last.fm social tags and listening profiles from the Echo Nest. While the MSD does not distribute audio due to copyright reasons, through the API of the 7digital service, 30-second audio previews can be obtained for the songs in the dataset. These 30-second previews will form the source for our raw audio input.

Using the MSD data, we consider several subcategories of learning sources within the Algorithm and Annotation categories; below, we give an overview of these, and specify what information we considered exactly for the learning labels in our work.

2.2.1 Algorithm

– **Self.** The music track is the learning source itself; in other words, intrinsic information in the input music track should be captured through a learning procedure, without employing further data. Various unsupervised or auto-regressive learning strategies can be employed under this category, with variants of Autoencoders, including the Stacked Autoencoder\cite{31,32}, Restricted Boltzmann Machines (RBM)\cite{33}, Deep Belief Networks (DBN)\cite{34} and Generative Adversarial Networks (GAN)\cite{35}. As another example within this category, variants of the Siamese networks for similarity learning can be considered\cite{36–38}.

In our case, we will employ the Siamese architecture to learn a metric that measures whether two input music clips belong to the same track, or two different tracks. This can be formally formulated as follows:

\[
\min_{\theta^{self}, \theta^*} \mathbb{E}_{X_l, X_r \sim D_{self}} \mathcal{L}(y_{self}, f_{self}(X_l, X_r; \theta^{self}, \theta^*))
\]

(5)

\[
y_{self} = \begin{cases} 
1, & \text{if } X_l \text{ and } X_r \text{ sampled from same track} \\
0, & \text{otherwise}
\end{cases}
\]

(6)

where $X_l$ and $X_r$ are a pair of randomly sampled short music snippets (taken from the 30-second MSD audio previews) and $f^{self}$ is a network for learning a metric between given input
representations in terms of the criteria imposed by $y^{\text{self}}$. It is composed of one or more fully-connected layers and one output layer with softmax activation. An global outline illustration of our chosen architecture is given in Fig. 3. Further specifications of the representation network and sampling strategies will be given in Section 3.

Fig. 3: Siamese architecture adopted for the self learning task. For further details of the Representation Network, see Section 3.1 and Fig. 4.

- **Feature.** Many algorithms exist already for extracting features out of musical audio, or for transforming musical audio representations. By running such algorithms on musical audio, learning labels are automatically computed, without the need for soliciting human annotations. Algorithmically computed outcomes will likely not be perfect, and include noise or errors. At the same time, we consider them as a relatively efficient way to extract semantically relevant and more structured information out of a raw input signal.
  
  In our case, under this category, we use Beat Per Minute (BPM) information, released as part of the MSD’s precomputed features. The BPM values were computed by an estimation algorithm, as part of the Echo Nest API.

2.2.2 Annotation

- **Metadata.** Typically, metadata will come ‘for free’ with music audio, specifying side information, such as a release year, the song title, the name of the artist, the corresponding album name, and the corresponding album cover image.
  
  Considering that this information describes categorization facets of the musical audio, metadata can be a useful information source to learn a music representation. In our experiments, we use release year information, which is readily provided as metadata with each song in the MSD.

- **Crowd.** Through interaction with music streaming or scrobbling services, large numbers of users, also designated as the crowd, left explicit or implicit information regarding their perspectives on musical content. For example, they may have created social tags, ratings, or social media
mentionings of songs. With many services offering API access to these types of descriptors, crowd data therefore offers scalable, spontaneous and diverse (albeit noisy) human perspectives on music signals.

In our experiments, we use social tags from Last.fm\(^1\) and user listening profiles from the Echo Nest.

- **Professional.** As mentioned in \([1]\), annotation of music tracks is a complicated and time-consuming process: annotation criteria frequently are subjective, and considerable domain knowledge and annotation experience may be required before accurate and consistent annotations can be made. Professional experts in categorization have this experience, and thus are capable of indicating clean and systematic information about musical content. It is not trivial to get such professional annotations at scale; however, these types of annotations may be available in existing professional libraries.

In our case, we use professional annotations from the Centrale Discotheek Rotterdam (CDR), the largest music library in The Netherlands, holding all music ever released in the country in physical and digital form in its collection. The CDR collection can be digitally accessed through the online Muziekweb\(^2\) platform. For each musical album in the CDR collection, genre annotations were made by a professional annotator, according to a fixed vocabulary of 367 hierarchical music genres.

As another professional-level ‘description’, we adopted lyrics information per each track, which is provided in Bag-of-Words format with the MSD. To filter out trivial terms such as stop-words, we applied TF-IDF\(^3\).

- **Combination.** Finally, learning labels can be derived from combinations of the above categories.

In our experiment, we used combination of artist information and social tags, by making a bag of tags at the artist level as a learning label.

Not all songs in the MSD actually include learning labels from all the sources mentioned above. Clearly, it is another advantage of using MTL that one can use such imbalanced datasets in a single learning procedure, to maximize the coverage of the dataset. However, on the other hand, if one uses an imbalanced number of samples across different learning sources, it is not trivial to compare the effect of individual learning sources. We therefore choose to work with a subset of the dataset, in which equal numbers of samples across learning sources can be used. As a consequence, we managed to collect 46,490 clips of tracks with corresponding learning source labels. A 41,841 / 4,649 split was made for training and validation for all sources from both MSD and CDR. Since we mainly focus on transfer learning, we used the validation set mostly for monitoring the training, to keep the network from overfitting.

### 2.3 Latent Factor Preprocessing

Most learning sources are noisy. For instance, social tags include tags for personal playlist management, long sentences, or simply typos, which do not actually show relevant nuances in describing the music signal. The algorithmically extracted BPM information also is imperfect, and likely contains octave errors, in which BPM is under- or overestimated by a factor of 2. To deal with this noise, several previous works using the MSD \([16, 26]\) applied a frequency-based filtering strategy

---

\(^1\) [https://labrosa.ee.columbia.edu/millionsong/lastfm](https://labrosa.ee.columbia.edu/millionsong/lastfm)

\(^2\) [https://www.muziekweb.nl/](https://www.muziekweb.nl/)
Table 1: Properties of learning sources.

| Identifier | Category | Data          | Dimensionality | Preprocessing |
|------------|----------|---------------|----------------|---------------|
| self       | Algorithm| MSD - Track   | 1              |               |
| bpm        | Feature  | MSD - BPM     | 1              | GMM           |
| year       | Metadata | MSD - Year    | 1              | GMM           |
| tag        | Crowd    | MSD - Tag     | 174,156        | pLSA          |
| taste      | Annotation| MSD - Taste   | 949,813        | pLSA          |
| cdr_tag    | Professional| CDR - Tag    | 367            | pLSA          |
| lyrics     | Professional| MSD - Lyrics | 5,000          | pLSA, TF-IDF  |
| artist     | Combination| MSD - Artist & Tag | 522,366       | pLSA          |

Table 2: Examples of Latent Topics extracted with pLSA from MSD social tags

| Topic | Strongest social tags                  |
|-------|----------------------------------------|
| tag1  | indie rock, indie, british, Scottish    |
| tag2  | pop, pop rock, dance, male vocalists   |
| tag3  | soul, rnb, funk, Neo-Soul              |
| tag4  | Melodic Death Metal, black metal, doom metal, Gothic Metal |
| tag5  | fun, catchy, happy, Favorite           |

Along with top-down domain knowledge. However, this shrinks the available sample size. As an alternative way to handle noisiness, several other previous works [11, 17, 27, 40–42] apply latent factor extraction using various low-rank approximation models to preprocess the label information. We also choose to do this in our experiments.

A full overview of chosen learning sources, their category, origin dataset, dimensionality and preprocessing strategies is shown in Table 1. In most cases, we apply probabilistic latent semantic analysis (pLSA), which extracts latent factors as a multinomial distribution of latent topics [43]. Table 2 illustrates several examples of strong social tags within extracted latent topics.

For situations in which learning labels are a scalar, non-binary value (BPM and release year), we applied a Gaussian Mixture Model (GMM) to transform each value into a categorical distribution of Gaussian components. In case of the Self category, as it basically is a binary membership test, no factor extraction was needed in this case.

After preprocessing, learning source labels $y_t$ are now expressed in the form of probabilistic distributions $z_t$. Then, the learning of a deep representation can take place by minimizing the Kullback–Leibler (KL) divergence between model inferences $f_t(X)$ and label factor distributions $z_t$.

Along with the noise reduction, another benefit from such preprocessing is that the regularization of the scale of the objective function between different tasks involved in the learning, when the resulted factors have the identical size. This regularity between the objective functions is particularly helpful for comparing different tasks and datasets. For this purpose, we used a fixed single value $k = 50$ for the number of factors (pLSA) and the number of Gaussians (GMM). In the remainder of this paper, the datasets and tasks processed in above manner will be denoted by learning sources for coherent presentation and usage of the terminology.
3 Representation Network Architectures

In this section, we present the detailed specification of the deep representation neural network architecture we exploited in this work. We will discuss the base architecture of the network, and further discuss the shared architecture with respect to different fusion strategies that one can take in the MTDTL context. Also, we introduce details on the preprocessing related to the input data served into networks.

3.1 Base Architecture

As the deep base architecture for feature representation training and learning, we choose a Convolutional Neural Network (CNN) architecture inspired by [21], as illustrated in Fig. 4.

The CNN is one of the most popular architectures in many music-related machine learning tasks [16, 17, 20, 25, 44–55]. Many of these works adopt an architecture having cascading blocks of 2-dimensional filters and max-pooling, derived from well-known works in image recognition [21, 56]. Although variants of CNN using 1-dimensional filters also were suggested by [12, 57–59] to learn features directly from a raw audio signal in an end-to-end manner, not many works managed to use them on music classification tasks successfully [60].

The main difference between the base architecture and [21] is that the Global Average Pooling (GAP) and the Batch Normalization (BN) layers. BN is applied to accelerate the training and stabilize the internal covariate shift for every convolution layer and the fc-feature layer [61]. Also, the global spatial pooling is adopted as the last pooling layer of the cascading convolution blocks, which is known as effectively summarizing the spatial dimensions both image [22] and music domain [20]. We also applied the approach to ensure the fc-feature layer not to have a huge number of parameters.

We applied the Rectified Linear Unit (ReLU) [62] to all convolution layers and the fc-feature layer. For the fc-output layer, softmax activation is used. For each convolution layer, we applied the zero-padding such that the input and the output have the same spatial shape. As for the regularization, we choose to apply drop-out [63] on the fc-feature layer. We added $L^2$ regularization across all the parameters with the same weight $\lambda = 10^{-6}$. Further details of the base architecture are summarized in Table 3.

3.1.1 Audio Preprocessing

We aim to learn a music representation from as-raw-as-possible input data to fully leverage the capability of the neural network. For this purpose, we use the dB-scale mel-scale magnitude spectrum of an input audio fragment, extracted by applying 128-band mel-filter banks on the Short-Time Fourier Transform (STFT). mel-spectrograms have generally been a popular input representation choice for CNNs applied in music-related tasks [16, 17, 20, 26, 41, 64]; besides, it also was reported recently that their frequency-domain summarization, based on psycho-acoustics, is efficient and not easily learnable through data-driven approaches [65, 66]. We choose a 1024-sample window size and 256-sample hop size, translating to about 46 ms and 11.6 ms respectively for a sampling rate of 22 kHz. We also applied standardization to each mel spectrum, making use of the mean and variance of all individual mel spectra in the training set.
3.1.2 Sampling

During the learning process, in each iteration, a random batch of songs is selected. Audio corresponding to these songs originally is 30 seconds in length; for computational efficiency, we randomly crop 2.5 seconds out of each song each time. Keeping stereo channels of the audio, the size of a single input tensor $X^*$ we used for the experiment ended up with $2 \times 216 \times 128$, where the first dimension indicates number of channels, and following dimensions mean time steps and mel-bins, respectively. Along with the computational efficiency, a number of literatures in MIR field reported that using the small chunk of the input not only inflates the dataset, but also shows good performance on the high-level tasks such as music auto-tagging \cite{20, 57, 60}. For the self case, we generate batches with equal numbers of songs for both membership categories in $y_{self}$.

![Fig. 4: Default CNN architecture for supervised single-source representation learning. Details of the Representation Network are presented at the left of the global architecture diagram. The numbers inside the parentheses indicate either the number of filters, or the number of units with respect to the type of layer.](image)

3.2 Multi-Source Architectures with Various Degrees of Shared Information

When learning a music representation based on various available learning sources, different strategies can be taken regarding the choice of architecture. We will investigate the following setups:

- As a base case, a **Single-Source Representation (SS-R)** can be learned for a single source only. As mentioned earlier, this would be the typical strategy leading to pre-trained networks, that later would be used in transfer learning. In our case, our base architecture from Section 3.1 and Fig. 4 will be used, for which the layers in the Representation Network also are illustrated in Fig. 5a. Out of the fc-feature layer, a $d$-dimensional representation is obtained.

Table 3: Configuration of the base CNN. \texttt{conv} and \texttt{max-pool} indicate a 2-dimensional convolution and max-pooling layer, respectively. We set the stride size with 2 on the time dimension of \texttt{conv1}, to compress dimensionality at the early stage. Otherwise, all strides are set as 1 across all the convolution layers. \texttt{gap} corresponds to the global average pooling used in \cite{22}, which averages out all the spatial dimensions of the filter responses. \texttt{fc} is an abbreviation of fully-connected layer. We use dropout with $p = 0.5$ only for the \texttt{fc-feature} layer, where the intermediate latent representation is extracted and evaluated. For simplicity, we omit the batch-size dimension of the input shape.

| Layer      | Input Shape | Weight Shape | Sub-Sampling | Activation |
|------------|-------------|--------------|--------------|------------|
| conv1      | $2 \times 216 \times 128$ | $2 \times 16 \times 5 \times 5$ | $2 \times 1$ | ReLU       |
| max-pool1  | $16 \times 108 \times 128$ | $16 \times 32 \times 3 \times 3$ | $2 \times 2$ | ReLU       |
| conv2      | $16 \times 54 \times 64$ | $16 \times 64 \times 3 \times 3$ | $2 \times 2$ | ReLU       |
| max-pool2  | $32 \times 54 \times 64$ | $32 \times 64 \times 3 \times 3$ | $2 \times 2$ | ReLU       |
| conv3      | $32 \times 27 \times 32$ | $32 \times 64 \times 3 \times 3$ | $2 \times 2$ | ReLU       |
| max-pool3  | $64 \times 27 \times 32$ | $64 \times 128 \times 3 \times 3$ | $2 \times 2$ | ReLU       |
| conv4      | $64 \times 13 \times 16$ | $64 \times 256 \times 1 \times 1$ | $2 \times 2$ | ReLU       |
| max-pool4  | $128 \times 6 \times 8$ | $128 \times 256 \times 3 \times 3$ | $2 \times 2$ | ReLU       |
| conv5      | $128 \times 3 \times 4$ | $128 \times 256 \times 1 \times 1$ | $2 \times 2$ | ReLU       |
| max-pool5  | $256 \times 3 \times 4$ | $256 \times 256 \times 1 \times 1$ | $2 \times 2$ | ReLU       |
| gap        | 256         | 256          | Learning source specific | Softmax    |
| fc-feature | 256         | 256          | Learning source specific | Softmax    |
| dropout    | 256         |              |              |            |
| fc-output  | 256         |              |              |            |

- If multiple perspectives on the same content, as reflected by the multiple learning labels, should also be reflected in the ultimate learned representation, one can learn \texttt{SS-R} representations for each learning source, and simply concatenate them afterwards. With $d$ dimensions per source and $m$ sources, this leads to a $d \times m$ \textbf{Multiple Single-Source Concatenated Representation (MSS-CR)}. In this case, independent networks are trained for each of the sources, and no shared knowledge will be transferred between sources. A layer setup of the corresponding Representation Network is illustrated in Fig. 5b.

- When applying MTL learning strategies, the deep architecture should involve shared knowledge layers, before branching out to various individual learning sources, whose learned representations will be concatenated in the final $d \times m$-dimensional representation. We call these \textbf{Multi-Source Concatenated Representations (MS-CR)}. As the branching point can be chosen at different stages, we will investigate the effect of various prototypical branching point choices: at the second convolution layer (\texttt{MS-CR@2}, Fig. 5c), the fourth convolution layer (\texttt{MS-CR@4}, Fig. 5d), and the sixth convolution layer (\texttt{MS-CR@6}, Fig. 5e). The later the branching point occurs, the more shared knowledge the network will employ.

- In the most extreme case, branching would only occur at the very last fully connected layer, and a \textbf{Multi-Source Shared Representation (MS-SR)} (or, more specifically, \texttt{MS-SR@FC}) is learned, as illustrated in Fig. 5f. As the representation is obtained from the \texttt{fc-feature} layer, no concatenation takes place here, and a $d$-dimensional representation is obtained.

A summary of these different representation learning architectures is given in Table 4. Beyond the strategies we choose, further approaches can be thought of to connect representations learned for different learning sources in neural network architectures. For example, for different tasks, represen-
Table 4: Properties of the various categories of representation learning architectures.

|                | Multi Source | Shared Network | Concatenation | Dimensionality |
|----------------|--------------|----------------|---------------|----------------|
| SS-R           | No           | No             | No            | $d$            |
| MSS-CR         | Yes          | No             | Yes           | $d \times m$   |
| MS-CR          | Yes          | Partial        | Yes           | $d \times m$   |
| MS-SR          | Yes          | Yes            | No            | $d$            |

Representations can be extracted from different intermediate hidden layers, benefiting from the hierarchical feature encoding capability of the deep network [26]. However, considering that learned representations are usually taken from a specific fixed layer of the shared architecture, we focus on the strategies as we outlined above.

3.3 MTL Training Procedure

**Algorithm 1:** Training a Multi-Source CNN

1. Initialize $\Theta$: \{$\theta^t$, $\theta^s$\} randomly;
2. for epoch in 1...N do
3. for iteration in 1...L do
4. Pick a learning source $t$ randomly;
5. Pick batch of samples from learning source $t$;
6. Derive learning label $z_t$;
7. Sub-sample chunk $X^*$ from track $X$;
8. Forward-pass:
   - $\mathcal{L}(y_{self}, \Theta, X_l, X^*) = \text{Eq. 5}$ for self;
   - $\mathcal{L}(z_t, \Theta, X^*) = \text{Eq. 2}$ otherwise;
9. Backward-pass: $\nabla_{\Theta} \mathcal{L}$;
10. Update model: $\Theta \leftarrow \Theta - \epsilon \nabla_{\Theta}$;

Similar to [4, 11], we choose to train the MTL models with a stochastic update scheme as described in Algorithm 1. At every iteration, a learning source is selected randomly. After the learning source is chosen, a batch of observation-label pairs ($X, z_l$) is drawn. For the audio previews belonging to the songs within this batch, an input representation $X^*$ is cropped randomly from its super-sample $X$. The updates of the parameters $\Theta$ are conducted through back-propagation using the Adam algorithm [67]. For each neural network we train, we set $L = lm$, where $l$ is the number of iterations needed to visit all the training samples with fixed batch size $b = 128$, and $m$ is the number of learning sources used in the training. Across the training, we used a fixed learning rate $\epsilon = 0.00025$. After a fixed number of epochs $N$ is reached, we stop the training.
### Fig. 5: The various model architectures considered in the current work. Beyond single-source architectures, multi-source architectures with various degrees of shared information are studied. For simplification, multi-source cases are illustrated here for two sources. The fc-feature layer from which representations will be extracted is the FC(256) layer in the illustrations (see Table 3).

(a) SS-R: Base setup.
(b) MSS-CR: Concatenation of multiple independent SS-R networks.
(c) MS-CR@2: network branches to source-specific layers from 2nd convolution layer.
(d) MS-CR@4: network branches to source-specific layers from 4th convolution layer.
(e) MS-CR@6: network branches to source-specific layers from 6th convolution layer.
(f) MS-SR@FC: heavily shared network, source-specific branching only at final FC layer.
3.4 Implementation Details

We used *PyTorch* \[68\] to implement the CNN models and parallel data serving. For evaluation models and cross-validation, we made extensive use of functionality in *Scikit-Learn* \[69\]. Furthermore, *Librosa* \[70\] was used to process audio files and its raw features including mel spectrograms. The training is conducted with 8 Graphical Processing Unit (GPU) computation nodes, composed of 2 NVIDIA GRID K2 GPUs and 6 NVIDIA GTX 1080Ti GPU.

Fig. 6: Overall system framework. The first row of the figure illustrates the learning scheme, where the representation learning is happening by minimizing the KL divergence between the network inference $f_t(X)$ and the preprocessed learning label $z_t$. The preprocessing is conducted by the blue blocks which transform the original noisy labels $y_t$ to $z_t$, reducing noise and summarizing the high-dimensional label space into a smaller latent space. The second row describes the entire evaluation scenario. The representation is first extracted from the representation network, which is transferred from the upper row. The sequence of representation vectors is aggregated as the concatenation of their means and standard deviations. The purple block indicates a machine learning model employed to evaluate the representation’s effectiveness.

4 Evaluation

So far, we discussed the details regarding the learning phase of this work, which corresponds to the upper row of Fig. 6. This included various choices of sources for the representation learning, and various choices of architecture and fusion strategies. In this section, we present the evaluation methodology we followed, as illustrated in the second row of Fig. 6. First, we will discuss the chosen target tasks and datasets in Section 4.1 followed in Section 4.2 by the baselines against which
our representations will be compared. Section 4.3 explains our experimental design, and finally we
discuss the implementation of our evaluation experiments in Section 4.4.

4.1 Target Datasets

In order to gain insight into the effectiveness of learned representations with respect to multiple
potential future tasks, we consider a range of target datasets. In this work, our target datasets
are chosen to reflect various semantic properties of music, purposefully chosen semantic biases, or
popularity in the MIR literature. Furthermore, the representation network should not be configured
or learned to explicitly solve the chosen target datasets.

While for the learning sources, we could provide categorizations on where and how the learning
labels were derived, and also consider algorithmic outcomes as labels, existing popular research
datasets mostly fall in the Professional or Crowd categories. In our work, we choose 7 evaluation
datasets commonly used in MIR research, which reflect three conventional types of MIR tasks,
namely classification, regression and recommendation:

Table 5: Properties of target datasets used in our experiments. Because of time constraints, we
sampled the Lastfm dataset as described in 4.1; the original size appears between parentheses. In
case particular data splits are defined by an original author or follow up study, we apply the same
split, including the reference in which the split is introduced. Otherwise, we applied either a random
split stratified by the label (Ballroom), or simple filtering based on reported faulty entries (IRMAS).

| Task        | Data      | #Tracks   | #Class | Split Method          |
|-------------|-----------|-----------|--------|-----------------------|
| Classification | FMA [71] | Genre     | 25,000 | 16 | Artist Filtered [71]  |
| Classification | GTZAN [72] | Genre     | 1,000  | 10 | Artist Filtered [73]  |
| Classification | Ext. Ballroom [74, 75] | Genre | 3,390  | 13 | N/A |
| Classification | IRMAS [76] | Instrument | 6,705  | 11 | Song Filtered         |
| Regression   | Music Emotion [77] | Arousal   | 744    | Genre Stratified [77] |
| Regression   | Music Emotion [77] | Valence   | 744    | Genre Stratified [77] |
| Recommendation | Lastfm* [78] | Listening Count | 27,093 (961,416) | N/A |

- **Classification.** Different types of classification tasks exist in MIR. In our experiments, we
  consider several datasets used for genre classification and instrument classification.
  For genre classification, we chose the GTZAN [72] and FMA [71] datasets as main exemplars.
  Even though GTZAN is known for its caveats [79], we deliberately used it, because its popularity
can be beneficial when comparing with previous and future work. We note though that there
may be some overlap between the tracks of GTZAN and the subset of the MSD we use in our
experiments; the extent of this overlap is unknown, due to the lack of a confirmed and exhaustive
track listing of the GTZAN dataset. We choose to use a fault-filtered data split for the training
and evaluation, which is suggested in [73]. The split originally includes a training, validation
and evaluation split; in our case, we also included the validation split as training data.

Among the various packages provided by the FMA, we chose the top-genre classification task of
FMA-Medium [74]. This is a classification dataset with an unbalanced genre distribution. We
used the data split provided by the dataset for our experiment, where the training is validation
set are combined as the training.
Considering another type of genre classification, we selected the Extended Ballroom dataset \[74, 75\]. Because the classes in this dataset are highly separable with regard to their BPM [80], we specifically included this ‘purposefully biased’ dataset as an example of how a learned representation may effectively capture temporal dynamics properties present in a target dataset, as long as learning sources also reflected these properties. Since no pre-defined split is provided or suggested by other literatures, we used stratified random sampling based on the genre label. The last dataset we considered for classification is the training set of the IRMAS dataset [76], which consists of short music clips annotated with the predominant instruments present in the clip. Compared to the genre classification task, instrument classification is generally considered as less subjective, requiring features to separate timbral characteristics of the music signal as opposed to high-level semantics like genre. We split the dataset to make sure that observations from the same music track are not split into training and test set. As performance metric for all these classification tasks, we used classification accuracy.

- **Regression.** As exemplars of regression tasks, we evaluate our proposed deep representations on the dataset used in the MediaEval Music Emotion prediction task [77]. It contains frame-level and song-level labels of a two-dimensional representation of emotion, with valence and arousal as dimensions [81]. Valence is related to the positivity or negativity of the emotion, and arousal is related to its intensity [77]. The song-level annotation of the V-A coordinates was used as the learning label, and we trained separate models for the two emotional dimensions. As for the dataset split, we used the split provided by the dataset, which is done by the random split stratified by the genre distribution. As evaluation metric, we measured the coefficient of determination $R^2$ of each model.

- **Recommendation.** Finally, we employed the ‘Last.fm - 1K users’ dataset [78] to evaluate our representations in the context of a content-aware music recommendation task (which will be denoted as Lastfm in the remaining of the paper). This dataset contains 19 million records of listening events across 961,416 unique tracks collected from 992 unique users. In our experiments, we mimicked a cold-start recommendation problem, in which items not seen before should be recommended to the right users. For efficiency, we filtered out users who listened to less than 5 tracks and tracks known to less than 5 users.

As for the audio content of each track, we obtained the mapping between the MusicBrainz Identifier (MBID) with the Spotify identifier (SpotifyID) using the MusicBrainz API. After cross-matching, we collected 30 seconds previews of all track using the Spotify API. We found that there is a substantial amount of the missing mapping information between the SpotifyID and MBID in MusicBrainz database, where only approximately 30% of mappings are available. Also, because of the substantial amount of inactive users and unpopular tracks in the dataset, we ultimately acquired a dataset of 985 unique users and 27,093 unique tracks with audio content. Similar to [28], we considered the outer matrix performance for un-introduced songs; in other words, the model’s recommendation accuracy on the items newly introduced to the system [28].

This was done by holding out certain tracks when learning user models, and then predicting user preference scores based on all tracks, including those that were held out, resulting in a ranked track list per user. As evaluation metric, we consider Normalized Discounted Cumulative Gain (nDCG@500), only treating held-out tracks that were indeed liked by a user as relevant items. Further details on how hold-out tracks were chosen are given in Section 4.4.
A summary of all evaluation datasets, their origins and properties, can be found in Table 5.

4.2 Baselines

We examined three baselines to compare with our proposed representations:

- **Mel-Frequency Cepstral Coefficients (MFCC).** These are some of the most popular audio representations in MIR research. In this work, we extract and aggregate MFCC following the strategy in [26]. In particular, we extracted 20 coefficients and also used their first- and second-order derivatives. After obtaining the sequence of MFCCs and its derivatives, we performed aggregation by taking the average and standard deviation over the time dimension, resulting in a 120-dimensional vector representation.

- **Random Network Feature (Rand).** We extracted the representation at the fc-feature layer without any representation network training. With random initialization, this representation therefore gives a random baseline for a given CNN architecture. We refer to this baseline as Rand.

- **Latent Representation from Music Auto-Tagger (Choi).** The work in [26] focused on a music auto-tagging task, and can be considered as yielding a state-of-the-art deep music representation for MIR. While the model’s focus on learning a representation for music auto-tagging can be considered as our SS-R case, there are a number of issues that complicate direct comparisons between this work and ours. First, the network in [26] is trained with about 4 times more data samples than in our experiments. Second, it employed a much smaller network than our architecture. Further, intermediate representations were extracted, which is out of the scope of our work, as we only consider representations at the fc-feature layer. Nevertheless, despite these caveats, the work still is very much in line with ours, making it a clear candidate for comparison. Throughout the evaluation, we could not fully reproduce the performance reported in the original paper [26]. When reporting our results, we therefore will report the performance we obtained with the published model, referring to this as Choi.

4.3 Experimental Design

In order to investigate our research questions, we carried out an experiment to study the effect of the number and type of learning sources on the effectiveness of deep representations, as well as the effect of the various architectural learning strategies described in Section 3.2. For the experimental design we consider the following factors:

- Representation strategy, with 6 levels: SS-R, MS-SR@FC, MS-CR@6, MS-CR@4, MS-CR@2, and MSS-CR).
- 8 2-level factors indicating the presence or not of each of the 8 learning sources: self, year, bpm, taste, tag, lyrics, cdv_tag and artist.
- Number of learning sources present in the learning process (1 to 8). Note that this is actually calculated as the sum of the eight factors above.
- Target dataset, with 7 levels: Ballroom, FMA, GTZAN, IRMAS, Lastfm, Arousal and Valence.

Given a learned representation, fitting dataset-specific models is much more efficient than learning the representation, so we decided to evaluate each representation on all 7 target datasets. The
Fig. 7: Aliasing among main effects in the final experimental design.

The experimental design is thus restricted to combinations of representation and learning sources, and for each such combination we will produce 7 observations. However, given the constraint of SS-R relying on a single learning source, that there is only one possible combination for \( n = 8 \) sources, as well as the high imbalance in the number of sources\(^5\), we proceeded in three phases:

1. We first trained the SS-R representations for each of the 8 sources, and repeated 6 times each. This resulted in 48 experimental runs.
2. We then proceeded to train all five multi-source strategies with all sources, that is, \( n = 8 \). We repeated this 5 times, leading to 25 additional experimental runs.
3. Finally, we ran all five multi-source strategies with \( n = 2, \ldots, 7 \). The full design matrix would contain 5 representations and 8 sources, for a total of 1,230 possible runs. Such an experiment was unfortunately infeasible to run exhaustively given available resources, so we decided to follow a fractional design. However, rather than using a pre-specified optimal design with a fixed amount of runs \( ^5 \), we decided to run sequentially for as long as time would permit us, generating at each step a new experimental run on demand in a way that would maximize desired properties of the design up to that point, such as balance and orthogonality\(^6\).

We did this with the greedy Algorithm. From the set of still remaining runs \( A \), a subset \( O \) is selected such that the expected imbalance in the augmented design \( B \cup \{ o \} \) is minimal. In this case, the imbalance of a design is defined as the maximum imbalance found between the levels of any factor, except for those already exhausted. From \( O \), a second subset \( P \) is selected such that the expected aliasing in the augmented design is minimal, here defined as the maximum absolute aliasing between main effects. Finally, a run \( p \) is selected at random from \( P \), the corresponding representation is learned, and the algorithm iterates again after updating \( A \) and \( B \).

\(^5\) For instance, from the 255 possible combinations of up to 8 sources, there are 70 combinations of \( n = 4 \) sources, but 28 with \( n = 2 \), or only 8 for \( n = 7 \). Simple random sampling from the 255 possible combinations would lead to a very imbalanced design, that is, a highly non-uniform distribution of observation counts across the levels of the factor (\( n \) in this case). A balanced design is desired to prevent aliasing and maximize statistical power. See \( ^{82} \) for details.

\(^6\) An experimental design is orthogonal if the effects of any factor balance out across the effects of the other factors. In a non-orthogonal design effects may be aliased, meaning that the estimate of one effect is partially biased with the effect of another, the extent of which ranges from 0 (no aliasing) to 1 (full aliasing). Aliasing is sometimes referred to as confounding. See \( ^{82} \) for details.
Following this on demand methodology, we managed to run another 352 experimental runs from all the 1,230 possible.

**Algorithm 2**: Sequential generation of experimental runs.

```plaintext
1 Initialize $A$ with all possible 1,230 runs to execute;
2 Initialize $B \leftarrow \emptyset$ for the set of already executed runs;
3 while time allows do
   4 Select $O \subseteq A$ s.t. $\forall o \in O$, the imbalance in $B \cup \{o\}$ is minimal;
   5 Select $P \subseteq O$ s.t. $\forall p \in P$, the aliasing in $B \cup \{p\}$ is minimal;
   6 Select $p \in P$ at random;
   7 Update $A \leftarrow A - \{p\}$;
   8 Update $B \leftarrow B \cup \{p\}$;
   9 Learn the representation coded by $p$;
```

After going through the three phases above, the final experiment contained $48 + 25 + 352 = 425$ experimental runs, each producing a different deep music representation. We further evaluated each representation on all 7 target datasets, leading to a grand total of $42 \times 7 = 2,975$ datapoints. Fig. 7 plots the alias matrix of the final experimental design, showing that the aliasing among main factors is indeed minimal. The final experimental design matrix can be downloaded along with the rest of the supplemental material.

Each considered representation network was trained using the CNN representation network model from Section 3 based on the specific combination of learning sources and deep architecture as indicated by the experimental run. In order to reduce variance, we fixed the number of training epochs to $N = 200$ across all runs, and applied the same base architecture, except for the branching point. This entire training procedure took approximately 5 weeks with given computational hardware resources introduced in Section 3.4.

### 4.4 Implementation Details

In order to assess how our learned deep music representations perform on the various target datasets, transfer learning will now be applied, to consider our representations in the context of these new target datasets.

As a consequence, new machine learning pipelines are set up, focused on each of the target datasets. In all cases, we applied the pre-defined split if it is feasible. Otherwise, we randomly split the dataset in a 80% training and 20% test set. For every dataset, we repeated the training and evaluation for 5 times. In most of our evaluation cases, validation will take place on the test set; in case of the the recommendation problem, the test set represents a set of tracks to be held out during user model training, and re-inserted for validation. In all cases, we will extract representations from evaluation dataset audio as detailed in Section 4.4.1 and then learn relatively simple models based on them, as detailed in Section 4.4.2. Employing the metrics as mentioned in the previous section, we will then take average performance scores over the 5 different train-test splits for final performance reporting.
4.4.1 Feature Extraction and Preprocessing

Taking raw audio from the evaluation datasets as input, we take non-overlapping slices out of this audio with a fixed length of 2.5 seconds. Based on this, we apply the same preprocessing transformations as discussed in Section 3.1.1. Then, we extract a deep representation from this preprocessed audio, employing the architecture as specified by the given experimental run. As in the case of Section 3.2, representations are extracted from the $fc$-feature layer of each trained CNN model. Depending on the choice of architecture, the final representation may consist of concatenations of representations obtained by separate representation networks.

Input audio may originally be (much) longer than 2.5 seconds; therefore, we aggregate information in feature vectors over multiple time slices by taking their mean and standard deviation values. As a result, we get a representation with averages per learned feature dimension, and another representation with standard deviations per feature dimension. These will be concatenated, as illustrated in Fig. 6.

4.4.2 Target Dataset-Specific Models

As our goal is not to over-optimize dataset-specific performance, but rather perform a comparative analysis between different representations (resulting from different learning strategies), we keep the model simple, and use fixed hyper-parameter values for each model across the entire experiment.

To evaluate the trained representations, we used different models according to the target dataset. For classification and regression tasks, we used Multi Layer Perceptron (MLP) model \[84\]. More specifically, the MLP model has two hidden layers, whose dimensionality is 256. As for the non-linearity, we choose ReLU \[62\] for all nodes, and the model is trained with ADAM optimization technique \[67\] for 200 iterations. In evaluation, we used the Scikit-Learn’s implementation for ease of distributed computing on multiple CPU computation nodes.

For the recommendation task, we choose a similar model as suggested in \[28, 85\], in which the learning objective function $L$ is defined as

$$
\min_{U,V,W} \|P - UV^T\|_C + \frac{\lambda^V}{2} \|V - XW\| + \frac{\lambda^U}{2} \|U\| + \frac{\lambda^W}{2} \|W\| \tag{7}
$$

where $P \in \mathbb{R}^{u \times i}$ is an binary matrix indicating whether there is interaction between $u$ users and $i$ items, $U \in \mathbb{R}^{u \times r}$ and $V \in \mathbb{R}^{i \times r}$ are $r$ dimensional user factors and item factors for the low-rank approximation of $R$, $W \in \mathbb{R}^{d \times r}$ is a free parameter for the projection from $d$-dimensional feature space to the factor space. $X \in \mathbb{R}^{i \times d}$ is the feature matrix where each row corresponds to a track. Finally, $\| \cdot \|_C$ is the Frobenious norm weighted by the confidence matrix $C \in \mathbb{R}^{u \times i}$, which controls the credibility of the model on the given interaction data, given as follows:

$$
C = 1 + \alpha R \tag{8}
$$

where $R \in \mathbb{R}^{u \times i}$ is the matrix containing the original interactions between users and items, $\alpha$ controls credibility. As for hyper-parameters, we set $\alpha = 0.1$, $\lambda^V = 0.00001$, $\lambda^U = 0.00001$, and $\lambda^W = 0.1$, respectively. For the number of factors we choose $r = 50$ to focus only on the relative impact of the representation over the different conditions. We implemented an update rule with the Alternating Least Squares (ALS) algorithm similar to \[28\], and updated parameters during 15 iterations.
5 Results and Discussion

In this section, we present results and discussion related to the proposed deep music representations. In Section 5.1, we will first compare the performance across the SS-Rs, to show how different individual learning sources work for each target dataset. Then, we will present general experimental results related to the performance of the multi-source representations. In Section 5.2, we discuss the effect of the number of learning sources exploited in the representation learning, in terms of their general performance, reliability, and model compactness. In Section 5.3, we discuss effectiveness of different representations in MIR. Finally, we present some initial evidence for multifaceted semantic explainability of the proposed MTDTL in Section 5.5.
5.1 Single-Source and Multi-Source Representation

Fig. 8 presents the performance of SS-R representations on each of the 7 target datasets. We can see that all sources tend to outperform the Rand baseline on all datasets, except for a handful of cases involving sources self and bpm. Looking at the top performing sources, we find that tag, cdr, tag and artist perform better or on-par with the most sophisticated baseline, Choi, except for the IRMAS dataset. The other sources are found somewhere between these two baselines, except for datasets Lastfm and Arousal, where they perform better than Choi as well. Finally, the MFCC is generally outperformed in all cases, with the notable exception of the IRMAS dataset, where only Choi performs better.

Zooming in to dataset-specific observed trends, the bpm learning source shows a highly skewed performance across target datasets: it clearly outperforms all other learning sources in the Ballroom dataset, but it achieves the worst or second worst performance in the other datasets. As shown in [80], this confirms that the Ballroom dataset is well-separable based on BPM information alone. Indeed, representations trained on the bpm learning source seem to contain a latent representation close to the BPM of an input music signal. In contrast, we can see that the bpm representation achieves the worst results in the Arousal dataset, where both temporal dynamics and BPM are considered as important factors determining the intensity of emotion.

On the IRMAS dataset, we see that all the SS-Rs perform worse than the MFCC and Choi baselines. Given that they both take into account low-level features, either by design or by exploiting low-level layers of the neural network, this suggests that predominant instrument sounds are harder to distinguish based solely on semantic features, which is the case of the representations studied here.

Also, we find that there is small variability for each SS-R runs within the training setup we applied. Specifically, 50% of cases we have within-SS-R variability less than 15% of the within-dataset variability. 90% of the cases are within 30% of the within-dataset variability.

We now consider how the various representations based on multiple learning sources perform, in comparison to those based on single learning sources. The boxplots in Fig. 9 show the distributions of performance scores for each architectural strategy and per target dataset. For comparison, the gray boxes summarize the distributions depicted in Fig. 8 based on the SS-R strategy. We can see that these SS-R obtain the lowest scores, followed by MS-SR@FC. Given that these representations have the same dimensionality, these results suggest that adding a single source-specific layer on top of a heavily shared model may help improving the adaptability of the neural network models, especially when there is no prior knowledge regarding the well-matching learning sources for the target datasets. The MS-CR and MSS-CR representations obtain the best results in general, which is somewhat expected because of their larger dimensionality.

5.2 Effect of Number of Learning Sources and Fusion Strategy

While the plots in Fig. 9 suggest that MSS-CR and MS-CR are the best strategies, the high observed variability makes this statement still rather unclear. In order to gain better insight of the effects of dataset, architecture strategies and number and type of learning sources, we further analyzed the results using a hierarchical or multilevel linear model on all observed scores [80].

For the reproducibility, we release all relevant materials including code, models and extracted features at [https://github.com/eldrin/MTLMusicRepresentation-PyTorch](https://github.com/eldrin/MTLMusicRepresentation-PyTorch).
such a model is essentially that it accounts for the structure in our experiment, where observations nested within datasets are not independent.

By Fig. 9 we can anticipate a very large dataset effect because of the inherently different levels of difficulty, as well as a high level of heteroskedasticity. We therefore analyzed standardized performance scores rather than raw scores. In particular, the i-th performance score \( y_i \) is standardized with the within-dataset mean and standard deviation scores, that is, \( y_i^* = (y_i - \bar{y}_{d[i]})/s_{d[i]} \), where \( d[i] \) denotes the dataset of the i-th observation. This way, the dataset effect is effectively 0 and the variance is homogeneous. In addition, this will allow us to compare the relative differences across strategies and number of sources using the same scale in all datasets.

We also transformed the variable \( n \) that refers to the number of sources to \( n^* \), which is set to \( n^* = 0 \) for SS-Rs and to \( n^* = n - 2 \) for the other strategies. This way, the intercepts of the linear model will represent the average performance of each representation strategy in its simplest case, that is, SS-R \((n = 1)\) or non-SS-R with \( n = 2 \). We fitted a first analysis model as follows:

\[
\begin{align*}
y_i^* &= \beta_{0r[i]d[i]} + \beta_{1r[i]d[i]} \cdot n_i^* + e_i \\
\beta_{0rd} &= \beta_{0r} + u_{0rd} \\
\beta_{1rd} &= \beta_{1r} + u_{1rd}
\end{align*}
\]

\( e_i \sim N(0, \sigma_e^2) \quad (9) \)

\( u_{0rd} \sim N(0, \sigma_{0r}^2) \quad (10) \)

\( u_{1rd} \sim N(0, \sigma_{1r}^2) \), \quad (11)
where $\beta_{0r[i,d][i]}$ is the intercept of the corresponding representation strategy within the corresponding dataset. Each of these coefficients is defined as the sum of a global fixed effect $\beta_{0r}$ of the representation, and a random effect $u_{0rd}$ which allows for random within-dataset variation.\footnote{We note that hierarchical models do not fit each of the individual $u_{0rd}$ coefficients (a total of 42 in this model), but the amount of variability they produce, that is, $\sigma_{0r}^2$ (6 in total).} This way, we separate the effects of interest (ie. each $\beta_{0r}$) from the dataset-specific variations (ie. each $u_{0rd}$). The effect of the number of sources is similarly defined as the sum of a fixed representation-specific coefficient $\beta_{1r}$ and a random dataset-specific coefficient $u_{1rd}$. Because the slope depends on the representation, we are thus implicitly modeling the interaction between strategy and number of sources, which can be appreciated in Fig. 10 especially with $MS-SR@FC$.

Fig. 11 shows the estimated effects and bootstrap 95% confidence intervals. The left plot confirms the observations in Fig. 9. In particular, they confirm that $SS-R$ performs significantly worse than $MS-SR@FC$, which is similarly statistically worse than the others. When carrying out pairwise comparisons, $MSS-CR$ outperforms all other strategies except $MS-CR@2 (p = 0.32)$, which outperforms all others except $MS-CR@6 (p = 0.09)$. The right plot confirms the qualitative observation from Fig. 10 by showing a significantly positive effect of the number of sources except for...
One Deep Music Representation to Rule Them All? 27

Fig. 11: Fixed effects and bootstrap 95% confidence intervals estimated for the first analysis model. The left plot depicts the effects of the representation strategy (β₀r intercepts) and the right plot shows the effects of the number of sources (β₁r slopes).

MS-SR@FC, where it is not statistically different from 0. The intervals suggest a very similar effect in the best representations, with average increments of about 0.16 per additional source —recall that scores are standardized.

To gain better insight into differences across representation strategies, we used a second hierarchical model where the representation strategy was modeled as an ordinal variable r* instead of the nominal variable r used in the first model. In particular, r* represents how many parameters are shared before they branch out to source-specific layers, so we coded SS-R as 0, MS-SR@FC as 0.2, MS-CR@6 as 0.4, MS-CR@4 as 0.6, MS-CR@2 as 0.8, and MSS-CR as 1 (see Fig. 5). In detail, this second model is as follows:

\[
y^*_i = \beta_0 + \beta_{1d[i]} \cdot r^*_i + \beta_{2d[i]} \cdot n^*_i + \beta_{3d[i]} \cdot r^*_i \cdot n^*_i + e_i \hspace{1cm} e_i \sim N(0, \sigma^2_e) \tag{12}
\]

\[
\beta_{1d} = \beta_{10} + u_{1d} \hspace{1cm} u_{1d} \sim N(0, \sigma^2_{1d}) \tag{13}
\]

\[
\beta_{2d} = \beta_{20} + u_{2d} \hspace{1cm} u_{2d} \sim N(0, \sigma^2_{2d}) \tag{14}
\]

\[
\beta_{3d} = \beta_{30} + u_{3d} \hspace{1cm} u_{3d} \sim N(0, \sigma^2_{3d}) \tag{15}
\]

In contrast to the first model, there is no representation-specific fixed intercept but an overall intercept β₀. The effect of the (level of sharing in the) representation is similarly modeled as the sum of an overall fixed slope β₁₀ and a random dataset-specific effect u₁d. Likewise, this model includes the main effect of the number of sources (fixed effect β₂₀), as well as its interaction with the level of representation share (fixed effect β₃₀). Fig. 12 shows the fitted coefficients, confirming the statistically positive effect of the level of sharing in the networks and, to a smaller degree but still significant, of the number of sources. The interaction term is not statistically significant, probably because of the unclear benefit of the number of sources in MS-SR@FC.

Overall, these analyses confirm that all multi-source strategies outperform the single-source representations, with a direct relation to the number of parameters in the network. In addition, there is a clearly positive effect of the number of sources, with a minor interaction between both factors.

Fig. 10 also suggests that the variability of performance scores decreases with the number of learning sources used. This implies that if there are more learning sources available, one can expect less variability across instantiations of the network. Most importantly, variability obtained for a single learning source (n = 1) is always larger than the variability with 2 or more sources. The
Ballroom dataset shows much smaller variability when BPM is included in the combination. For this specific dataset, this indicates that once bpm is used to learn the representation, the expected performance is stable and does not vary much, even if we keep including more sources. Section 5.3 provides more insight in this regard.

5.3 Single-Source vs. Multi-Source

The evidence so far tells us that, on average, learning from multiple sources leads to better performance than learning from a single source. However, it could be possible that the SS-R representation with the best learning source for the given target dataset still performs better than a multi-source alternative. In fact, in Fig. 10 there are many cases where the best SS-R representation (black circles at n = 1) already perform quite well compared to the more sophisticated alternatives. Fig. 13 presents similar scatter plots, but now explicitly differentiating between representations using the single best source (filled circles, solid lines) and not using it (empty circles, dashed lines). The results suggest that even if the strongest learning source for the specific dataset is not used, the others largely compensate for it in the multi-source representations, catching up and even surpassing the best SS-R representations. The exception to this rule is again bpm in the Ballroom dataset, where it definitely makes a difference. As the plots shows, the variability for low numbers of learning sources is larger when not using the strongest source, but as more sources are added, this variability reduces.

To further investigate this issue, for each target dataset, we also computed the variance component due to each of the learning sources, excluding SS-R representations [87]. A large variance due to one of the sources means that, on average and for that specific dataset, there is a large difference in performance between having that source or not. Table 6 shows all variance components, highlighting the per-dataset largest. Apart from bpm in the Ballroom dataset, there is no clear evidence that one single source is specially good in all datasets, which suggests that in general there is not a single source that one would use by default. Notably though, sources artist, tag and self tend to have large variance components.

In addition, we observe that the sources with largest variance are not necessarily the sources that obtain the best results by themselves in an SS-R representation (see Fig. 5). We examined this...
relationship further by calculating the correlation between variance components and (standardized) performance of the corresponding SS-Rs. The Pearson correlation is 0.38, meaning that there is a mild association. Fig. 14 further shows this with a scatterplot, with a clear distinction between poorly-performing sources (year, taste and lyrics at the bottom) and well-performing sources (tag, cdr_tag and artist at the right).

This result implies that even if some SS-R is particularly strong for a given dataset, when considering more complex fusion architectures, the presence of that one source is not necessarily required because the other sources make up for its absence. This is especially important in practical terms, because different tasks generally have different best sources, and practitioners rarely have sufficient domain knowledge to select them up front. Also, and unlike the Ballroom dataset, many real-world problems are not easily solved with a single feature. Therefore, choosing a general representation based on multiple sources is a much simpler way to proceed which still yields comparable or better results.

In other words, if “a single deep representation to rule them all” is pre-trained, it is advisable to base this representation on multiple learning sources. At the same time, given that MSS-CR repre-
Table 6: Variance components (as percent of total) of the learning sources, within each of the target datasets, and for non-SS-R representations. Largest per dataset in bold face.

|                | Ballroom | FMA | GTZAN | IRMAS | Lastfm | Arousal | Valence |
|----------------|----------|-----|-------|-------|--------|---------|---------|
| self           | 2        | 32  | 39    | 18    | 29     | 6       | 10      |
| year           | <1       | 6   | <1    | 1     | 2      | 2       | <1      |
| bpm            | 96       | 3   | <1    | 8     | 16     | <1      | 42      |
| taste          | <1       | <1  | <1    | <1    | <1     | <1      | 6       |
| tag            | 1        | 17  | 21    | 16    | 20     | 33      | 14      |
| lyrics         | <1       | <1  | <1    | 3     | <1     | 11      | <1      |
| cdr_tag        | <1       | 9   | 12    | 16    | 2      | 16      | 14      |
| artist         | 1        | 32  | 28    | 37    | 32     | 31      | 15      |

sentations also generally show strong performance (albeit that they will bring high dimensionality), and that they will come ‘for free’ as soon as SS-R networks are trained, alternatively, we could imagine an ecosystem in which the community could pre-train and release many SS-R networks for different individual sources in a distributed way, and practitioners can then collect these into MSS-CR representations, without the need for retraining.

5.4 Compactness

Under an MTDTL setup with branching (the MS-CR architectures), as more learning sources are used, not only the representation will grow larger, but so will the necessary deep network to learn it: see Fig. 15 for an overview of necessary model parameters for the different architectures. When using all the learning sources, MS-CR@6, which for a considerable part encompasses a shared network architecture and branches out relatively late, has an around 6.3 times larger network size compared to the network size needed for SS-R. In contrast, MS-SR@FC, which is the most heavily shared MTDTL case, uses a network that is only 1.2 times larger than the network needed for SS-R.

Also, while the representations resulting from the MSS-CR and various MS-CR architectures linearly depend on the chosen number of learning sources $m$ (see Table 4), for MS-SR@FC, which
One Deep Music Representation to Rule Them All? 31

number of learning sources
10^5
10^6
10^7
number of model parameters
SS-R
MSS-CR
MS-CR@2
MS-CR@4
MS-CR@6
MS-SR@fc

Fig. 15: Number of network parameters by number of learning sources.

has a fixed dimensionality of $d$ independent of $m$, we do notice increasing performance as more
learning sources are used, except IRMAS dataset. This implies that under MTDTL setups, the
network does learn as much as possible from the multiple sources, even in case of fixed network
capacity.

5.5 Multiple Explanatory Factors

By training representation models on multiple learning sources in the way we did, our hope is
that the representation will reflect latent semantic facets that will ultimately allow for semantic
explainability. In Fig. 16, we show a visualization that suggests this indeed may be possible. More
specifically, we consider one of our MS-CR models trained on 5 learning sources. For each learning
source-specific block of the representation, using the learning source-specific fc-out layers, we can
predict a factor distribution $z_t$ for each of the learning sources. Then, from the predicted $z_t$, one can
either map this back on the original learning labels $y_t$, or simply consider the strongest predicted
topics (which we visualized in Fig. 16), to relate the representation to human-understandable facets
or descriptions.\footnote{The specific model used in the visualization is 232th model from the experimental design we introduce in Section 4.3, which is performing better than 95% of other models on GTZAN target dataset.}

\footnote{Note that, as soon as a pre-trained representation network model will be adapted to a new dataset through transfer learning, the fc-out layer cannot be used to obtain such explanations from the learning sources used in the representation learning, since the layers will then be fine-tuned to another dataset. However, we hypothesize it may be possible that the semantic explainability can still be preserved, if fine-tuning is jointly conducted with the original learning sources used during the pre-training time in the multi-objective strategy.}
Fig. 16: Potential semantic explainability of DTMTL music representations. Here, we provide a visualization using t-SNE [88], plotting 2-dimensional coordinates of each sample from the GTZAN dataset, as resulting from an MS-CR representation trained on 5 sources [11]. In the zoomed-in panes, we overlay the strongest topic model terms in $z_t$, for various types of learning sources.

6 Conclusion

In this paper, we have investigated the effect of different strategies to learn music representations with deep networks, considering multiple learning sources and different network architectures with varying degrees of shared information. Our main research questions are how the number and combination of learning sources (RQ1), and different configurations of the shared architecture (RQ2)
affect effectiveness of the learned deep music representation. As a consequence, we conducted an experiment training 425 neural network models with different combinations of learning sources and architectures.

After an extensive empirical analysis, we can summarize our findings as follows:

- **RQ1** The number of learning sources positively affects the effectiveness of a learned deep music representation, although representations based on a single learning source will already be effective in specialized cases (e.g. BPM and the Ballroom dataset).

- **RQ2** In terms of architecture, the amount of shared information has a negative effect on performance: larger models with less shared information (e.g. MS-CR@2, MSS-CR) tend to outperform models where sharing is higher (e.g. MS-CR@6, MS-SR@FC), all of which outperform the base model (SS-R).

Our findings give various pointers to useful future work. First of all, ‘generality’ is difficult to define in the music domain, maybe more so than in CV or NLP, in which lower-level information atoms may be less multifaceted in nature (e.g. lower-level representations of visual objects naturally extend to many vision tasks, while an equivalent in music is harder to pinpoint). In case of clear task-specific data skews, practitioners should be pragmatic about this.

Also, we only investigated one special case of transfer learning, which might not be generalized well if one consider the adaptation of the pre-trained network for the further fine-tuning with respect to their target dataset. Since there are various choices to make, which will bring substantial amount of the variability, we decided to leave the aspects for the further future works. We believe open-sourcing the models we trained throughout this work will be helpful for such follow-up works. Another limitation of current work is the selective set of label types in the learning sources. For instance, there are also a number of MIR related tasks that are using a time-variant labels such as the automatic music transcription, segmentation, beat tracking and chord estimation. We believe that such tasks should be investigated as well in the future to build a more complete overview of MTDTL problem.

Finally, in our current work, we still largely considered MTDTL as a ‘black box’ operation, trying to learn how MTDTL can be effective. However, the original reason for starting this work was not only to yield an effective general-purpose representation, but one that also would be semantically interpretable according to different semantic facets. We showed some early evidence our representation networks may be capable of picking up such facets; however, considerable future work will be needed into more in-depth analysis techniques of what the deep representations actually learned.

**Acknowledgements** This work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative. We further thank the CDR for having provided their album-level genre annotations for our experiments. We thank Keunwoo Choi for the discussion and all the help regarding the implementation of his work. We also thank David Tax for the valuable inputs and discussion. Finally, we thank editors and reviewers for their effort and constructive help to make improve this work.

**Conflict of interest:** Jaehun Kim, Julián Urbano, Cynthia C. S. Liem and Alan Hanjalic state that there are no conflicts of interest.

**References**

1. Michael A. Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future
2. Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. doi: 10.1023/A:1007379606734.

3. Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013. doi: 10.1109/TPAMI.2013.50.

4. Wu Liu, Tao Mei, Yongdong Zhang, Cherry Che, and Jiebo Luo. Multi-task deep visual-semantic embedding for video thumbnail selection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3707–3715, 2015. doi: 10.1109/CVPR.2015.7298994.

5. Joachim Bingel and Anders Søgaard. Identifying beneficial task relations for multi-task learning in deep neural networks. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, 2(2016):164–169, 2017.

6. Sijin Li, Zhi-Qiang Liu, and Antoni B Chan. Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 482–489, 2014.

7. Wenlu Zhang, Rongjian Li, Tao Zeng, Qian Sun, Sudhir Kumar, Jieping Ye, and Shuizhang Ji. Deep model based transfer and multi-task learning for biological image analysis. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 1475–1484, 2015. doi: 10.1145/2783258.2783304.

8. Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaou Tang. Facial landmark detection by deep multi-task learning. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI*, pages 94–108, 2014. doi: 10.1007/978-3-319-10599-4\_7.

9. Lukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Lion Jones, and Jakob Uszkoreit. One model to learn them all. *CoRR*, abs/1706.05137, 2017. URL http://arxiv.org/abs/1706.05137

10. Jen-Hao Rick Chang, Chun-Liang Li, Barnabás Póczos, and B. V. K. Vijaya Kumar. One network to solve them all - solving linear inverse problems using deep projection models. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 5889–5898, 2017. doi: 10.1109/ICCV.2017.627.

11. Jason Weston, Samy Bengio, and Philippe Hamel. Multi-Tasking with Joint Semantic Spaces for Large-Scale Music Annotation and Retrieval. *Journal of New Music Research*, (November 2012):37–41, 2011. ISSN 0929-8215. doi: 10.1080/09298215.2011.603834.

12. Yusuf Aytar, Carl Vondrick, and Antonio Torralba. Soundnet: Learning sound representations from unlabeled video. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 892–900, 2016.

13. Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010*, pages 339–344, 2010.

14. Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Conference on Machine Learning, ICML*
One Deep Music Representation to Rule Them All?

2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012.

15. Jan Schlüter and Sebastian Böck. Improved musical onset detection with convolutional neural networks. In IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014, pages 6979–6983, 2014. doi: 10.1109/ICASSP.2014.6854953.

16. Keunwoo Choi, György Fazekas, and Mark B. Sandler. Automatic tagging using deep convolutional neural networks. In Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016, pages 805–811, 2016.

17. Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., pages 2643–2651, 2013.

18. Pritish Chandna, Marius Miron, Jordi Janer, and Emilia Gómez. Monoaural audio source separation using deep convolutional neural networks. In Latent Variable Analysis and Signal Separation - 13th International Conference, LVA/ICA 2017, Grenoble, France, February 21-23, 2017, Proceedings, pages 258–266, 2017. doi: 10.1007/978-3-319-53547-0\_25.

19. Il-Young Jeong and Kyogu Lee. Learning temporal features using a deep neural network and its application to music genre classification. pages 434–440, 2016.

20. Yoonchang Han, Jae-Hun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. IEEE/ACM Trans. Audio, Speech & Language Processing, 25(1):208–221, 2017. doi: 10.1109/TASLP.2016.2632307.

21. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014. URL [http://arxiv.org/abs/1409.1556](http://arxiv.org/abs/1409.1556).

22. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

23. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pages 1–9, 2015. doi: 10.1109/CVPR.2015.7298594.

24. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. CoRR, abs/1301.3781, 2013. URL [http://arxiv.org/abs/1301.3781](http://arxiv.org/abs/1301.3781).

25. Sander Dieleman, Philemon Brakel, and Benjamin Schrauwen. Audio-based music classification with a pretrained convolutional network. In Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011, pages 669–674, 2011.

26. Keunwoo Choi, György Fazekas, Mark B. Sandler, and Kyunghyun Cho. Transfer learning for music classification and regression tasks. In Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017, pages 141–149, 2017.

27. Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Transfer learning by supervised pre-training for audio-based music classification. In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October
27-31, 2014, pages 29–34, 2014.

28. Dawen Liang, Minshu Zhan, and Daniel P. W. Ellis. Content-aware collaborative music recommendation using pre-trained neural networks. In Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015, pages 295–301, 2015.

29. Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3994–4003, 2016.

30. Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011, pages 591–596, 2011.

31. Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006, pages 153–160, 2006.

32. Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008, pages 1096–1103, 2008. doi: 10.1145/1390156.1390294.

33. Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, COLORADO UNIV AT BOULDER DEPT OF COMPUTER SCIENCE, 1986.

34. Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. Neural Computation, 18(7):1527–1554, 2006. doi: 10.1162/neco.2006.18.7.1527.

35. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems, December 8-13 2014, Montreal, Quebec, Canada, pages 2672–2680, 2014.

36. Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. pages 3279–3286, 2015. doi: 10.1109/CVPR.2015.7298948.

37. Relja Arandjelovic and Andrew Zisserman. Look, listen and learn. In IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, pages 609–617, 2017. doi: 10.1109/ICCV.2017.73.

38. Yu-Siang Huang, Shu-Yu Chou, and Yi-Hsuan Yang. Similarity embedding network for unsupervised sequential pattern learning by playing music puzzle games. CoRR, abs/1709.04384, 2017. URL http://arxiv.org/abs/1709.04384.

39. Gerard Salton and Michael McGill. Introduction to Modern Information Retrieval. McGraw-Hill Book Company, 1984. ISBN 0-07-054484-0.

40. Paul Lamere. Social Tagging and Music Information Retrieval. Journal of New Music Research, 37(2):101–114, 2008. ISSN 0929-8215. doi: 10.1080/09298210802479284.

41. Philippe Hamel, Matthew E. P. Davies, Kazuyoshi Yoshii, and Masataka Goto. Transfer learning in mir: Sharing learned latent representations for music audio classification and similarity. In Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013, Curitiba, Brazil, November 4-8, 2013, pages 9–14, 2013.
42. Edith Law, Burr Settles, and Tom Mitchell. Learning to Tag using Noisy Labels. *European Conference on Machine Learning*, 2010.

43. Thomas Hofmann. Probabilistic latent semantic analysis. In *UAI ’99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 289–296, 1999.

44. Jan Schlüter. Learning to pinpoint singing voice from weakly labeled examples. pages 44–50, 2016.

45. Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin W. Wilson. CNN architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, pages 131–135, 2017. doi: 10.1109/ICASSP.2017.7952132.

46. Honglak Lee, Peter T. Pham, Yan Largman, and Andrew Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pages 1096–1104, 2009.

47. Eric J. Humphrey and Juan Pablo Bello. Rethinking automatic chord recognition with convolutional neural networks. In *11th International Conference on Machine Learning and Applications, ICMLA, Boca Raton, FL, USA, December 12-15, 2012. Volume 2*, pages 357–362, 2012. doi: 10.1109/ICMLA.2012.220.

48. Toru Nakashika, Christophe Garcia, and Tetsuya Takiguchi. Local-feature-map integration using convolutional neural networks for music genre classification. In *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, pages 1752–1755, 2012.

49. Karen Ullrich, Jan Schlüter, and Thomas Grill. Boundary detection in music structure analysis using convolutional neural networks. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, pages 417–422, 2014.

50. Karol J. Piczak. Environmental sound classification with convolutional neural networks. In *25th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2015, Boston, MA, USA, September 17-20, 2015*, pages 1–6, 2015. doi: 10.1109/MLSP.2015.7324337.

51. Andrew J. R. Simpson, Gerard Roma, and Mark D. Plumbley. Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network. In *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*, pages 429–436, 2015. doi: 10.1007/978-3-319-22482-4\_50.

52. Huy Phan, Lars Hertel, Marco Maß, and Alfred Mertins. Robust audio event recognition with 1-max pooling convolutional neural networks. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 3653–3657, 2016. doi: 10.21437/Interspeech.2016-123.

53. Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *14th International Workshop on Content-Based Multimedia Indexing, CBMI 2016, Bucharest, Romania, June 15-17, 2016*, pages 1–6, 2016. doi: 10.1109/CBMI.2016.7500246.
54. Bartłomiej Stasiak and Jędrzej Monko. Analysis of time-frequency representations for musical onset detection with convolutional neural network. In Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, FedCSIS 2016, Gdańsk, Poland, September 11-14, 2016., pages 147–152, 2016. doi: 10.15439/2016F558.

55. Hong Su, Hui Zhang, Xueliang Zhang, and Guanglai Gao. Convolutional neural network for robust pitch determination. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016, pages 579–583, 2016. doi: 10.1109/ICASSP.2016.7471741.

56. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. Commun. ACM, 60(6):84–90, 2017. doi: 10.1145/3065386.

57. Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4–9, 2014, pages 6964–6968, 2014. doi: 10.1109/ICASSP.2014.6854950.

58. Áaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. page 125, 2016.

59. Navdeep Jaitly and Geoffrey E. Hinton. Learning a better representation of speech soundwaves using restricted boltzmann machines. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic, pages 5884–5887, 2011. doi: 10.1109/ICASSP.2011.5947700.

60. Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. CoRR, abs/1703.01789, 2017. URL http://arxiv.org/abs/1703.01789.

61. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, pages 448–456, 2015.

62. Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel, pages 807–814, 2010.

63. Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

64. Juhan Nam, Jorge Herrera, Malcolm Slaney, and Julius O. Smith. Learning sparse feature representations for music annotation and retrieval. In Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012, pages 565–570, 2012.

65. Keunwoo Choi, George Fazekas, Kyunghyun Cho, and Mark B. Sandler. A comparison on audio signal preprocessing methods for deep neural networks on music tagging. CoRR, abs/1709.01922, 2017. URL http://arxiv.org/abs/1709.01922.

66. Monika Dörfler, Thomas Grill, Roswitha Bammer, and Arthur Flexer. Basic filters for convolutional neural networks: Training or design? CoRR, abs/1709.02291, 2017. URL http://arxiv.org/abs/1709.02291.

67. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

68. Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in
One Deep Music Representation to Rule Them All?

69. Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and douard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2012. ISSN 1532-4435. doi: 10.1007/s13398-014-0173-7.

70. Brian Mcfee, Colin Raffel, Dawen Liang, Daniel P W Ellis, Matt Mcvicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and Music Signal Analysis in Python. *PROC. OF THE 14th PYTHON IN SCIENCE CONF*, 2015.

71. Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. In *18th International Society for Music Information Retrieval Conference*, 2017.

72. George Tzanetakis and Perry R. Cook. Musical genre classification of audio signals. *IEEE Trans. Speech and Audio Processing*, 10(5):293–302, 2002. doi: 10.1109/TSA.2002.800560.

73. Corey Kereliuk, Bob L. Sturm, and Jan Larsen. Deep learning and music adversaries. *IEEE Trans. Multimedia*, 17(11):2059–2071, 2015. doi: 10.1109/TMM.2015.2478068.

74. Fabien Gouyon, Anssi Klapuri, Simon Dixon, M. Alonso, George Tzanetakis, C. Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Trans. Audio, Speech & Language Processing*, 14(5):1832–1844, 2006. doi: 10.1109/TSA.2005.858509.

75. Ugo Marchand and Geoffroy Peeters. Scale and shift invariant time/frequency representation using auditory statistics: Application to rhythm description. In *26th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2016, Vietri sul Mare, Salerno, Italy, September 13-16, 2016*, pages 1–6, 2016. doi: 10.1109/MLSP.2016.7738904.

76. Juan J. Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera. A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. In *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, pages 559–564, 2012.

77. Mohammad Soleymani, Michael N. Caro, Erik M. Schmidt, Cheng-Ya Sha, and Yi-Hsuan Yang. 1000 songs for emotional analysis of music. In *Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia, CrowdMM@ACM Multimedia 2013, Barcelona, Spain, October 22, 2013*, pages 1–6, 2013. doi: 10.1145/2506364.2506365.

78. Óscar Celma. *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer, 2010. ISBN 978-3-642-13286-5. doi: 10.1007/978-3-642-13287-2.

79. Bob L. Sturm. The state of the art ten years after a state of the art: Future research in music information retrieval. *Journal of New Music Research*, 43(2):147–172, 2014. doi: 10.1080/09298215.2014.894533.

80. Bob L. Sturm. The "horse" inside: Seeking causes behind the behaviors of music content analysis systems. *Computers in Entertainment*, 14(2):3:1–3:32, 2016. doi: 10.1145/2967507.

81. J Posner, J A Russell, and B S Peterson. The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. *Dev Psychopathol*, 17(03):715–734, 2005. ISSN 1469-2198. doi: 10.1017/S0954579405050340.

82. Douglas C. Montgomery. *Design and Analysis of Experiments*. Wiley, 8th edition, 2012.

83. Peter Goos and Bradley Jones. *Optimal Design of Experiments: A Case Study Approach*. Wiley, 2011.
84. Geoffrey E. Hinton. Connectionist learning procedures. *Artif. Intell.*, 40(1-3):185–234, 1989. doi: 10.1016/0004-3702(89)90049-0.
85. Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative Filtering for Implicit Feedback Datasets Yifan. *IEEE International Conference on Data Mining*, pages 263–272, 2008. ISSN 15504786. doi: 10.1109/ICDM.2008.22.
86. Andrew Gelman and Jennifer Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Press, Cambridge University, 2006.
87. Shayle R. Searle, George Casella, and Charles E. McCulloch. *Variance components*. Wiley, 2006.
88. Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.