One deep music representation to rule them all? A comparative analysis of different representation learning strategies

Kim, Jaehun; Urbano, Julián; Liem, Cynthia C.S.; Hanjalic, Alan

DOI
10.1007/s00521-019-04076-1

Publication date
2019

Document Version
Final published version

Published in
Neural Computing and Applications

Citation (APA)
Kim, J., Urbano, J., Liem, C. C. S., & Hanjalic, A. (2019). One deep music representation to rule them all? A comparative analysis of different representation learning strategies. *Neural Computing and Applications*, *32 (2020)*(4), 1067-1093. https://doi.org/10.1007/s00521-019-04076-1

Neural Computing and Applications (2020) 32:1067–1093
https://doi.org/10.1007/s00521-019-04076-1

DEEP LEARNING FOR MUSIC AND AUDIO

One deep music representation to rule them all? A comparative analysis of different representation learning strategies

Jaehun Kim[1] · Julián Urbano[1] · Cynthia C. S. Liem[1] · Alan Hanjalic[1]

Received: 7 December 2017 / Accepted: 12 February 2019 / Published online: 4 March 2019

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, their representation is unlikely to be informative for all possible future tasks. In this paper, we present the results of our investigation of what are the most important factors 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 into how to approach the design of methods for learning widely deployable deep data representations in the music domain.

Keywords Representation learning · Music Information Retrieval · Multitask 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. As a consequence, in many cases, while music-related tasks are well understood by humans, it often is hard to pinpoint and describe where

✉ Jaehun Kim
J.H.Kim@tudelft.nl

[1] Multimedia Computing Group, Department of Intelligent Systems, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, Netherlands

the truly 'relevant' information is in the music data used for the tasks, and how this properly can be translated into numeric representations that should be used for prediction. While research into such proper translations can be conducted per individual task, it is likely that informative factors in music data will be shared across tasks. As a consequence, when seeking to identify informative factors that are not explicitly restricted to a single task, multitask learning (MTL) is a promising strategy. In MTL, 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 multitask learning.

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 *multitask 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 MTDTL, 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 effectiveness 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 into *how to choose the optimal multitask network architecture* under 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, an investigation is still needed on where in the layered architecture branching should ideally happen—if a branching strategy would turn out beneficial in the first place.

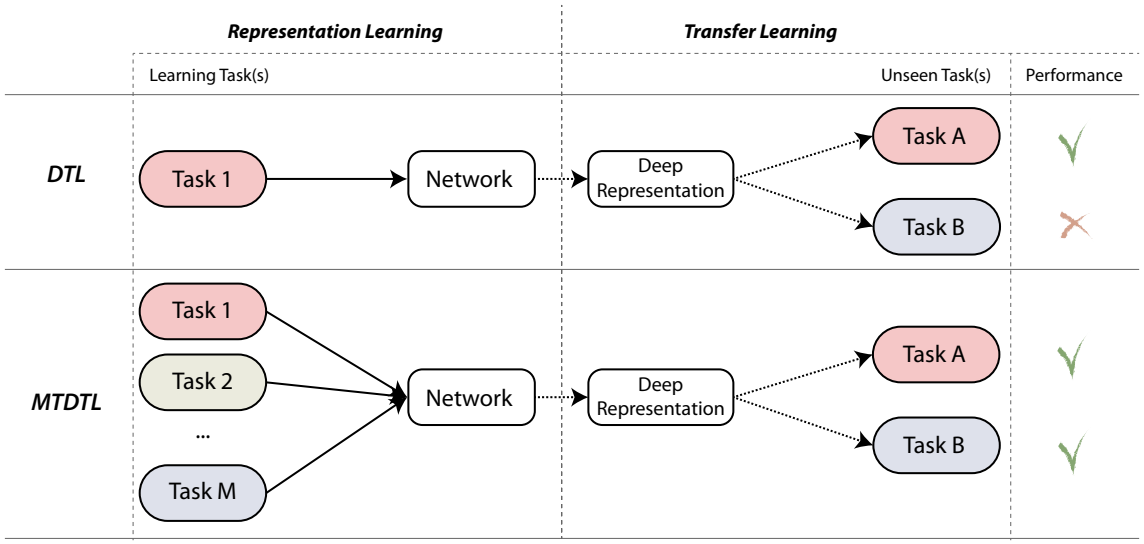


Fig. 1 Simplified illustration of the conceptual difference between traditional deep transfer learning (DTL) based on a single learning task (above) and multitask based deep transfer learning (MTDTL) (below). The same color used for a learning and an target task indicates that the tasks have commonalities, 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 (color figure online)

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 the 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 a 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.

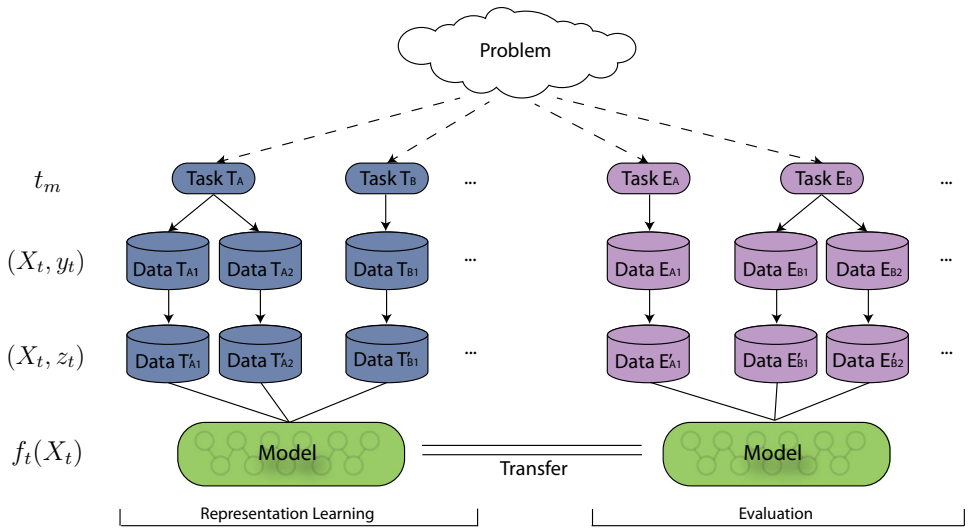
The scientific contribution of this work can be summarized as follows:

- We provide insight into the effectiveness of various deep representation learning strategies under the multitask 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.

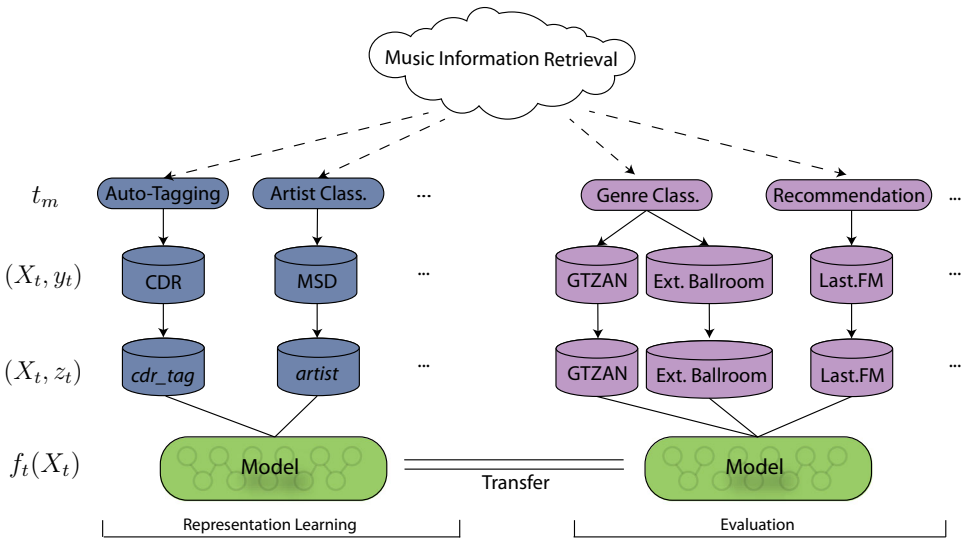
The rest of this work is presented as follows: a formalization of this problem, as well as the global outline of how learning will be performed based on different learning tasks from different sources, will be presented in Sect. 2. Detailed specifications of the deep architectures we considered for the learning procedure will be discussed in Sect. 3. Our strategy to *evaluate* the effectiveness of different representation network variants by employing various *target datasets* will be the focus of Sect. 4. Experimental results will be discussed in Sect. 5, after which general conclusions will be presented in Sect. 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 Sect. 2.1. Subsequently in Sect. 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 Sect. 2.3, ensuring that the learning sources are more regularized, such that their comparative analysis is clearer.



(a) Multi-Task Transfer Learning in General Problem Domain



(b) Multi-Task Transfer Learning in Music Information Retrieval Domain

Fig. 2 Schematic overview of what this work investigates. The upper scheme illustrates a general problem solving framework in which multitask transfer learning is employed. The tasks $t \in \{t_0, t_1, \ldots, t_M\}$ are derived from a certain problem domain, which is 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 $\mathcal{L}$ for given dataset $\mathcal{D}_t = \{ (x_t^{(i)}, y_t^{(i)}) \mid i \in \{1, \ldots, I\} \}$, 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:

$$\hat{\theta} = \arg\min \; \mathbb{E}_{\mathcal{D}_t} \mathcal{L}(y_t, f_t(x_t; \theta)) \tag{1}$$

where $x_t \in \mathbb{R}^d$ is, traditionally, a hand-crafted d-dimensional feature vector and θ 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:

$$\hat{\theta} = \arg\min \mathbb{E}_{\mathcal{D}_t}\mathcal{L}(y_t, f_t(X_t;\theta)). \tag{2}$$

In MTL, in the process of learning the network model f, different tasks will need to be solved in parallel. In the 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 MTL scenario as follows:

$$\hat{\theta}^s, \hat{\theta}^* = \arg\min \mathbb{E}_{t\in\mathcal{T}}\mathbb{E}_{\mathcal{D}_t}\mathcal{L}(y_t, f_t(X_t;\theta^s,\theta^t)) \tag{3}$$

Here, $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ is a given set of tasks to be learned and $\theta^* = \{\theta^1, \theta^2, \ldots, \theta^m\}$ indicates a set of model parameters θ^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 θ^s and θ^t, respectively. Updates for all parameters can be achieved through standard back-propagation. Further specifics on network architectures and training configurations will be given in Sect. 3.

Given the formalizations above, the first step in our framework is to select a suitable set $\mathcal{T}$ 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 as possible. 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:

$$\hat{\theta}^s, \hat{\theta}^* = \arg\min \mathbb{E}_{t\in\mathcal{T}}\mathbb{E}_{\mathcal{D}_t}\mathcal{L}(z_t, f_t(X_t;\theta^s,\theta^t)) \tag{4}$$

where $z_t \in \mathbb{R}^k$ is a k-dimensional vector that represents a reduced learning label for a specific task t. Each z_t will be obtained through task-specific factor extraction methods, as described in Sect. 2.3.

2.2 Learning sources

In 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 a resource for our learning experiments is the Million Song Dataset (MSD) [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-s audio previews can be obtained for the songs in the dataset. These 30-s 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 [31, 32], Restricted Boltzmann Machines (RBM) [33], Deep Belief Networks (DBN) [34] and Generative Adversarial Networks (GAN) [35]. As another example within this category, variants of the Siamese networks for similarity learning can be considered [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 formulated as follows:

$$\hat{\theta}^{self}, \hat{\theta}^s = \arg\min \mathbb{E}_{X_l, X_r \sim \mathcal{D}_{self}}\mathcal{L}(y_{self}, f_{self}(X_l, X_r;\theta^{self},\theta^s)) \tag{5}$$

$$y_{self} = \begin{cases} 1, & \text{if } X_l \text{ and } X_r \text{ sampled from same track} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

  where X_l and X_r are a pair of randomly sampled short music snippets (taken from the 30-s 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_{self}. It is composed of one or more

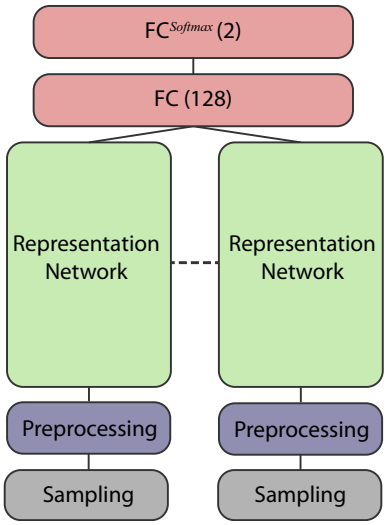


Fig. 3 Siamese architecture adopted for the *self* learning task. For further details of the representation network, see Sect. 3.1 and Fig. 4

fully connected layers and one output layer with softmax activation. A 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 Sect. 3.

- ***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 [39].
- ***Combination.*** Finally, learning labels can be derived from combinations of the above categories. In our experiment, we used a 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 unbalanced datasets in a single learning procedure, to maximize the coverage of the dataset. However, on the other hand, if one uses an unbalanced number of samples across different

[1] https://labrosa.ee.columbia.edu/millionsong/lastfm.

[2] https://www.muziekweb.nl/.

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 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 the 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 the regularization of the scale of the objective function between different tasks involved in the learning, when the resulting factors have the same 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 the 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 learning, we choose a convolutional neural network (CNN) architecture inspired by [21], as described in Fig. 4 and Table 3.

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 the use of 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, global spatial pooling is adopted as the last pooling layer of the cascading convolution blocks, which is known to effectively summarize the spatial dimensions both in the 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 zero-padding such that the input and the output have the same spatial shape. As for the regularization, we choose to apply dropout [63] on the `fc-`

Table 1 Properties of learning sources

Identifier	Category		Data	Dimensionality	Preprocessing
self	Algorithm	Self	MSD—Track	1	
bpm		Feature	MSD—BPM	1	GMM
year	Annotation	Metadata	MSD—Year	1	GMM
tag		Crowd	MSD—Tag	174,156	pLSA
taste		Crowd	MSD—Taste	949,813	pLSA
cdr_tag		Professional	CDR—Tag	367	pLSA
lyrics		Professional	MSD—Lyrics	5000	pLSA, TF-IDF
artist		Combination	MSD—Artist and 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`

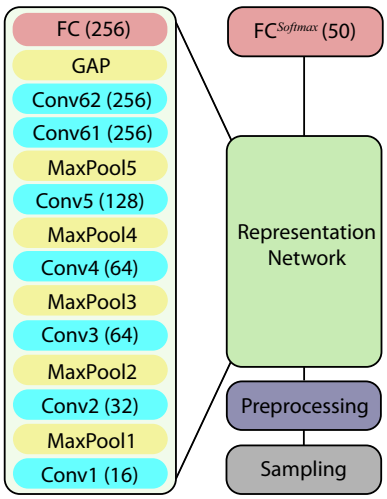


Fig. 4 Default CNN architecture for supervised single-source representation learning. Details of the representation network are presented at the left of the global architecture diagram. The numbers inside the parentheses indicate either the number of filters or the number of units with respect to the type of layer

`feature` layer. We added $L2$ regularization across all the parameters with the same weight $\lambda = 10^{-6}$.

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 CNN 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 frequency band of the 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 s in length; for computational efficiency, we randomly crop 2.5 s 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 the number of channels, and following dimensions mean time steps and mel-bins, respectively. Along with the computational efficiency, a number of previous works in MIR field reported that using a 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 [20, 57, 60]. For the *self* case, we generate batches with equal numbers of songs for both membership categories in y_{self}.

Table 3 Configuration of the base CNN

Layer	Input shape	Weight shape	Sub-sampling	Activation
conv1	2 × 216 × 128	2 × 16 × 5 × 5	2 × 1	ReLU
max-pool1	16 × 108 × 128		2 × 2	
conv2	16 × 54 × 64	16 × 32 × 3 × 3		ReLU
max-pool2	32 × 54 × 64		2 × 2	
conv3	32 × 27 × 32	32 × 64 × 3 × 3		ReLU
max-pool3	64 × 27 × 32		2 × 2	
conv4	64 × 13 × 16	64 × 64 × 3 × 3		ReLU
max-pool4	64 × 13 × 16		2 × 2	
conv5	64 × 6 × 8	64 × 128 × 3 × 3		ReLU
max-pool5	128 × 6 × 8		2 × 2	
conv61	128 × 3 × 4	128 × 256 × 3 × 3		ReLU
conv62	256 × 3 × 4	256 × 256 × 1 × 1		ReLU
gap	256			
fc-feature	256	256 × 256		ReLU
dropout	256			
fc-output	256	Learning source specific		Softmax

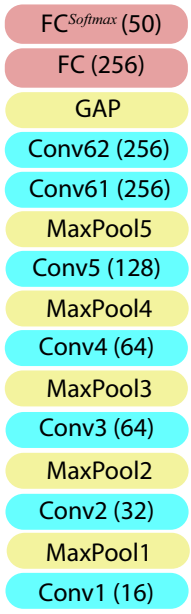
conv and max-pool indicate a 2-dimensional convolution and max-pooling layer, respectively. We set the stride size with 2 on the time dimension of conv1, to compress dimensionality at the early stage. Otherwise, all strides are set as 1 across all the convolution layers. gap corresponds to the global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

3.2 Multi-source architectures with various degrees of shared information

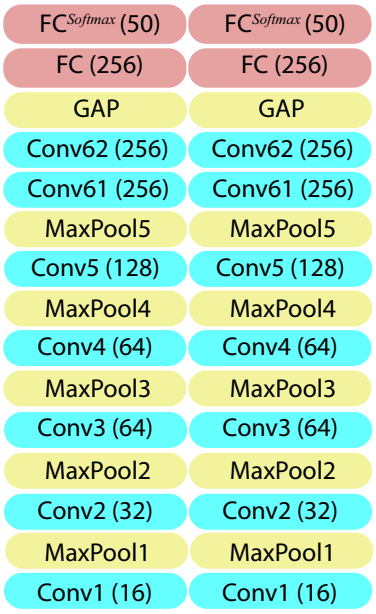
When learning a music representation based on various available learning sources, different strategies can be taken regarding the choice of architecture. We will investigate the following setups:

- 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 Sect. 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.
- If multiple perspectives on the same content, as reflected by the multiple learning labels, should also be reflected in the learned representation, one can learn *SS-R* representations for each learning source and simply concatenate them afterward. With d dimensions per source and m sources, this leads to a $d \times m$ ***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 ***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 (*MS-CR@2*, Fig. 5c), the fourth convolution layer (*MS-CR@4*, Fig. 5d), and the sixth convolution layer (*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 **Multi-Source Shared Representation (MS-SR)** (or, more specifically, *MS-SR@FC*) is learned, as illustrated in Fig. 5f. As the representation is obtained from the fc-feature layer, no concatenation takes place here, and a d-dimensional representation is obtained.

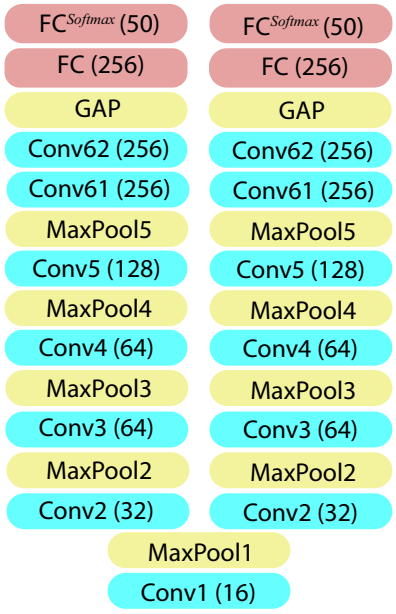
A summary of these different representation learning architectures is given in Table 4. Beyond the strategies we choose, further approaches can be thought of to connect



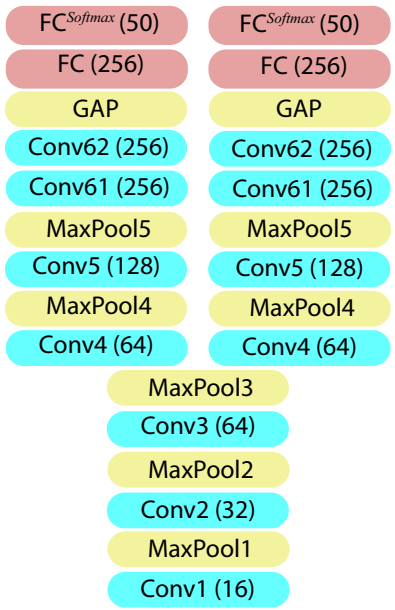
(a) SS-R: Base setup.



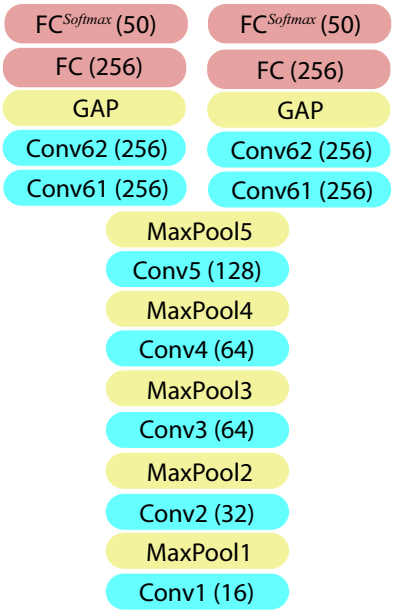
(b) MSS-CR: Concatenation of multiple independent SS-R networks.



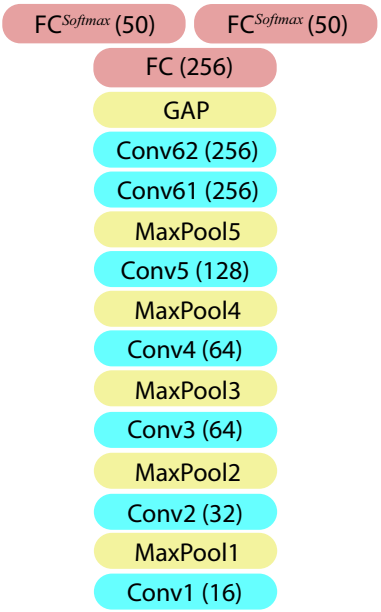
(c) MS-CR@2: network branches to source-specific layers from 2nd convolution layer.



(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.

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)

representations learned for different learning sources in neural network architectures. For example, for different tasks, 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.

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

3.3 MTL training procedure

Algorithm 1: Training a Multi-Source CNN

```
Initialize Θ: {θ^t, θ^s} randomly;
for epoch in 1...N do
    for iteration in 1...L do
        Pick a learning source t randomly;
        Pick batch of samples from learning source t;
        (X_l, X_r) for self;
        X otherwise;
        Derive learning label z_t;
        Sub-sample chunk X* from track X;
        Forward-pass:;
        L(y_self, Θ, X_l*, X_r*) =Eq. 5 for self;
        L(z_t, Θ, X*) =Eq. 2 otherwise;
        Backward-pass: ∇(Θ);
        Update model: Θ ← Θ − ε∇(Θ);
```

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_t) 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 Θ 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.

3.4 Implementation details

We used *PyTorch* [68] to implement the CNN models and parallel data serving. For the evaluation of 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 GPUs.

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 Sect. 4.1, followed in Sect. 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 Sect. 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, the 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:

- ***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 compared 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,

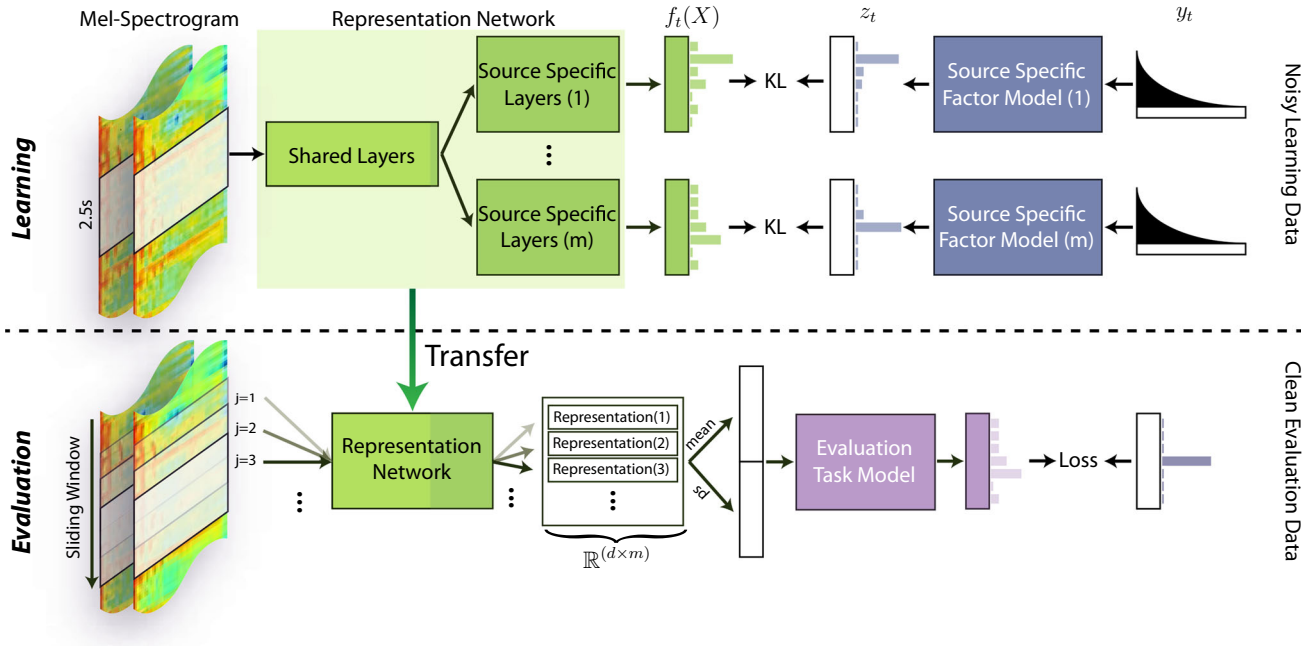


Fig. 6 Overall system framework. The first row of the figure illustrates the learning scheme, where the representation learning is happening by minimizing the KL divergence between the network inference $f_t(X)$ and the preprocessed learning label z_t. The preprocessing is conducted by the blue blocks 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 (color figure online)

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 [71]. 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 literature, 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 the genre. We split the dataset to make sure that observations from the same music track are not split into training and test sets.

As a 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. In similar fashion to the approach taken in [26], 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 an 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`.[3] After cross-matching, we collected 30 s previews of all track using the `Spotify API`.[4] We found that there is a substantial amount of missing mapping information between the SpotifyID and MBID in the `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 an 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 Sect. 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 Coefficient (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 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 Sect. 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*, *cdr_tag,* and *artist*.

[3] https://musicbrainz.org/.

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

Table 5 Properties of target datasets used in our experiments

Task	Data		#Tracks	#Class	Split method
Classification	FMA [71]	Genre	25,000	16	Artist Filtered [71]
Classification	GTZAN [72]	Genre	1000	10	Artist Filtered [73]
Classification	Ext. Ballroom [74, 75]	Genre	3390	13	N/A
Classification	IRMAS [76]	Instrument	6705	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

Because of time constraints, we sampled the Lastfm dataset as described in Sect. 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)

- 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 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 unbalance 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 1230 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 [83], 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 2. From the set of still remaining runs $\mathcal{A}$, a subset $\mathcal{O}$ is selected such that the expected unbalance in the augmented design $\mathcal{B} \cup \{o\}$ is minimal. In this case, the unbalance of design is defined as the maximum unbalance found between the levels of any factor, except for those already exhausted.[7] From $\mathcal{O}$, a second subset $\mathcal{P}$ is selected such that the expected aliasing in the augmented design is minimal, here defined as the maximum absolute aliasing between main effects.[8] Finally, a run p is selected at random from $\mathcal{P}$, the corresponding representation is learned, and the algorithm iterates again after updating $\mathcal{A}$ and $\mathcal{B}$.

Following this on-demand methodology, we managed to run another 352 experimental runs from all the 1230 possible.

[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 unbalanced 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 section 15.2 in [82] for details on unbalanced designs.

[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 sections 8.5 and 9.5 in [82] for details on aliasing.

[7] For instance, let a design have 20 runs for *SS-R*, 16 for *MS-SR@FC*, and 18 for all other representations. The unbalance in the representation factor is thus $20 - 16 = 4$. The total unbalance of the design is defined as the maximum unbalance found across all factors.

[8] See section 2.3.7 in [83] for details on how to compute an alias matrix.

Algorithm 2: Sequential generation of experimental runs.

```
Initialize A with all possible 1,230 runs to execute;
Initialize B ← ∅ for the set of already executed runs;
while time allows do
    Select O ⊆ A s.t. ∀o ∈ O, the unbalance in B ∪ {o} is minimal;
    Select P ⊆ O s.t. ∀p ∈ P, the aliasing in B ∪ {p} is minimal;
    Select p ∈ P at random;
    Update A ← A − {p};
    Update B ← B ∪ {p};
    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 $42 \times 7 = 2975$ data points. Figure 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 Sect. 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 Sect. 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 into an 80% training set and 20% test set. For every dataset, we repeated the training and evaluation for 5 times, using different train/test splits. In most of our evaluation cases, validation will take place on the test set; in case of the recommendation problem, the test set represents a set of tracks to be held out from each user during model training, and re-inserted for validation. In all cases, we will extract representations from evaluation dataset audio as detailed in Sect. 4.4.1, and then learn relatively simple models based on them, as detailed in Sect. 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.

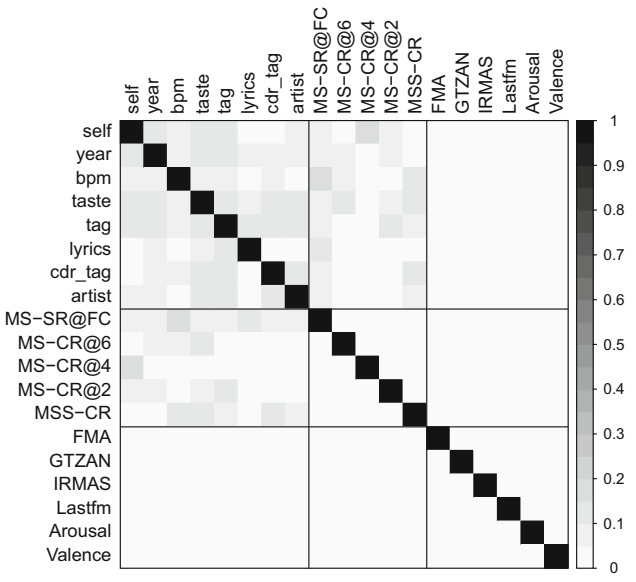


Fig. 7 Aliasing among main effects in the final experimental design

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 s. Based on this, we apply the same preprocessing transformations as discussed in Sect. 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 Sect. 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 s; therefore, we aggregate information in feature vectors over multiple time slices by taking their *mean* and *standard deviation* values. As a result, we get 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 the multilayer

perceptron (MLP) model [84]. More specifically, the MLP model has two hidden layers, whose dimensionality is 256. As for the nonlinearity, we choose ReLU [62] for all nodes, and the model is trained with ADAM optimization technique [67] for 200 iterations. In the 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 $\mathcal{L}$ is defined as

$$\hat{U}, \hat{V}, \hat{W} = \arg\min \ ||P - UV^T||_C + \frac{\lambda^V}{2}||V - XW|| + \frac{\lambda^U}{2}||U|| + \frac{\lambda^W}{2}||W|| \quad (7)$$

where $P \in \mathbb{R}^{u \times i}$ is a binary matrix indicating whether there is interaction between users u and items i, $U \in \mathbb{R}^{u \times r}$ and $V \in \mathbb{R}^{i \times r}$ are r dimensional user factors and item factors for the low-rank approximation of P. P is derived from the original interaction matrix $R \in \mathbb{R}^{u \times i}$, which contains the number of interaction from users u to items i, as follows:

$$P_{u,i} = \begin{cases} 1, & \text{if } R_{u,i} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

$W \in \mathbb{R}^{d \times r}$ 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:

$$C = 1 + \alpha R \quad (9)$$

where α 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 Sect. 5.1, we will first compare the performance across the *SS-R*s, 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 Sect. 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 Sect. 5.3, we discuss the effectiveness of different representations in MIR. Finally, we present some initial evidence for multi-faceted semantic explainability of the proposed MTDTL in Sect. 5.5.[9]

5.1 Single-source and multi-source representation

Figure 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 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-R*s 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* run within the training setup we applied. Specifically, in 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.

[9] For the reproducibility, we release all relevant materials including code, models and extracted features at https://github.com/eldrin/MTLMusicRepresentation-PyTorch.

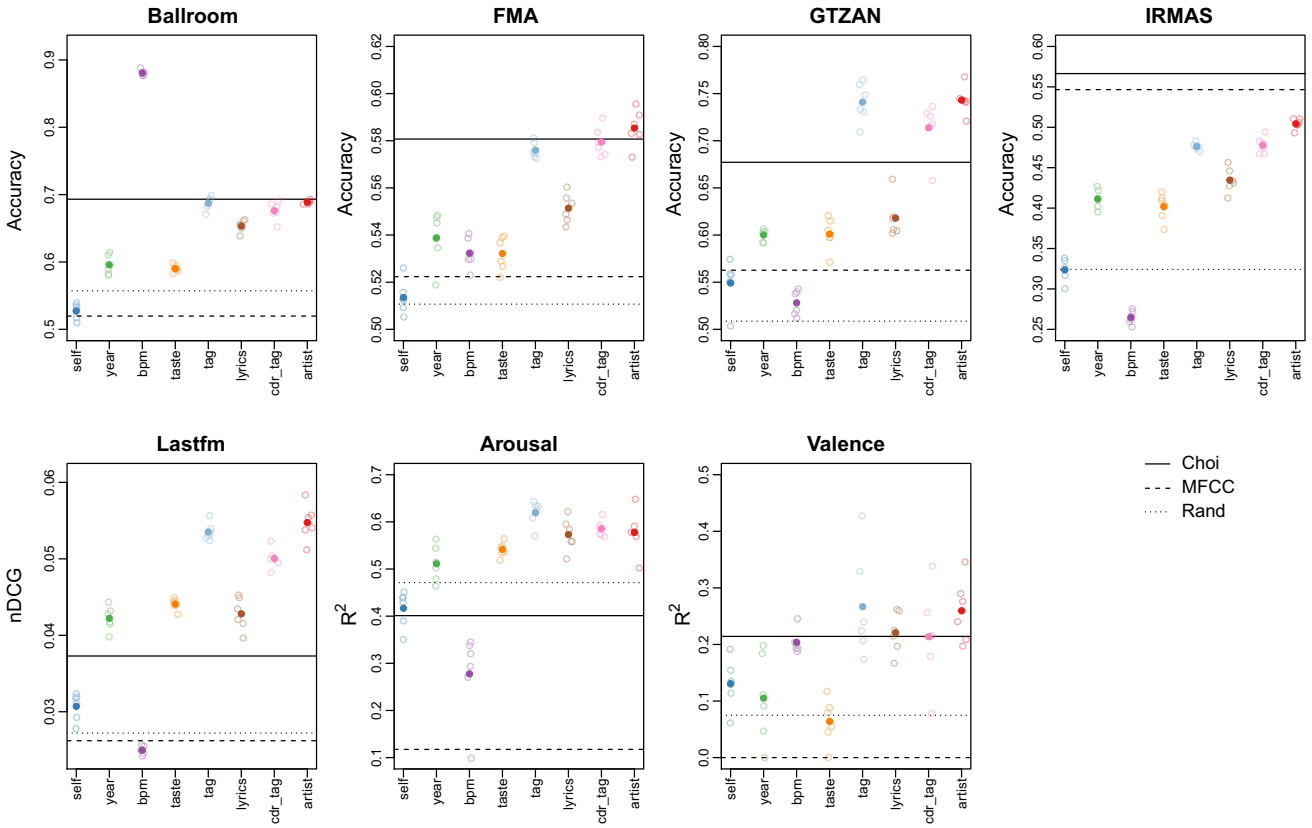


Fig. 8 Performance of single-source representations. Each point indicates the performance of a representation learned from a single source. Solid points indicate the average performance per source. The baselines are illustrated as horizontal lines

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. In general, we can see that these *SS-R* obtain the lowest scores, followed by *MS-SR@FC*, except for the IRMAS dataset. 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 to improve 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 a better insight of the effects of the dataset, architecture strategies and number and type of learning sources, we further analyzed the results using a hierarchical or multi-level linear model on all observed scores [86]. The advantage of 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-R*s and to $n^* = n - 2$ for the other strategies. This way, the intercepts of the linear model will represent the average

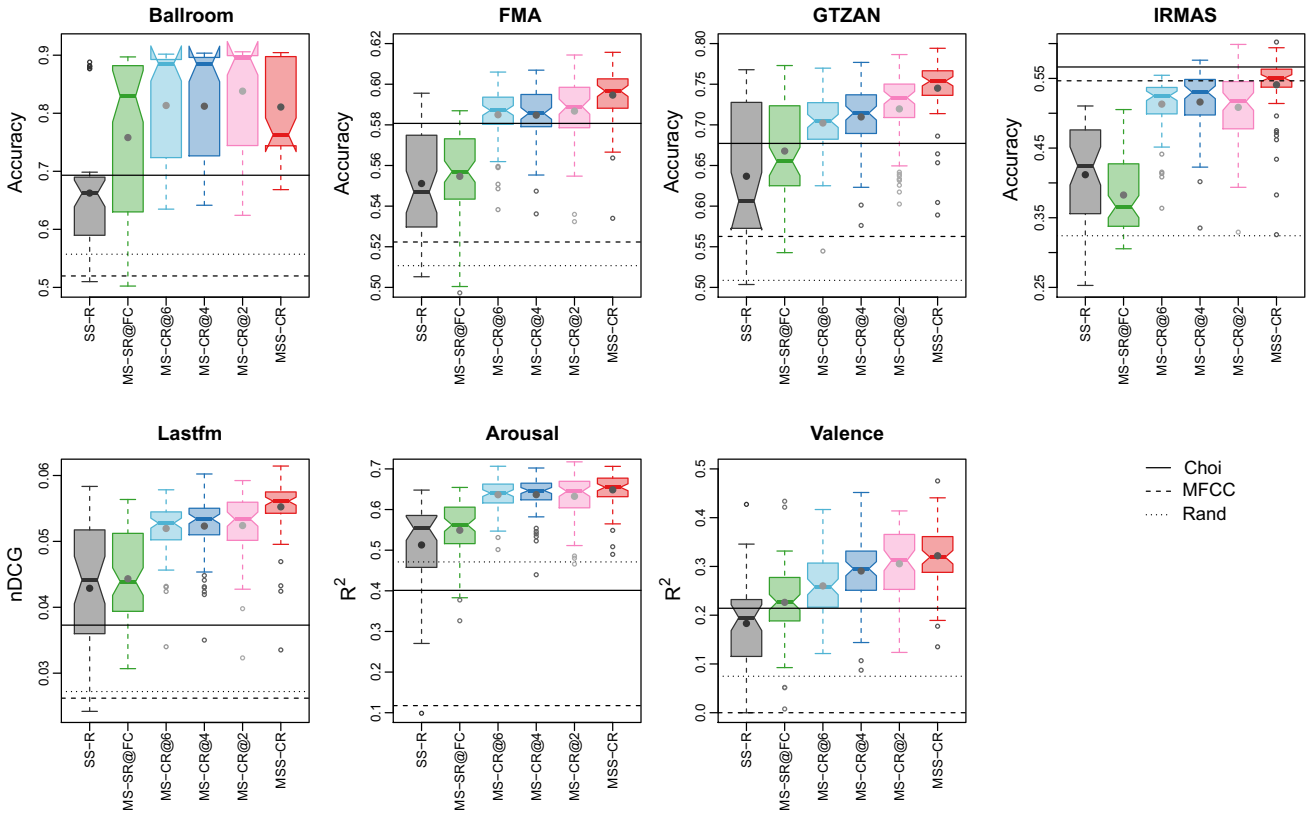


Fig. 9 Performance by representation strategy. Solid points represent the mean per representation. The baselines are illustrated as horizontal lines

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:

$$y_i^* = \beta_{0r[i]d[i]} + \beta_{1r[i]d[i]} \cdot n_i^* + e_i \quad e_i \sim N(0, \sigma_e^2) \tag{10}$$

$$\beta_{0rd} = \beta_{0r} + u_{0rd} \quad u_{0rd} \sim N(0, \sigma_{0r}^2) \tag{11}$$

$$\beta_{1rd} = \beta_{1r} + u_{1rd} \quad u_{1rd} \sim N(0, \sigma_{1r}^2), \tag{12}$$

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 β_{0r} of the representation, and a random effect u_{0rd} which allows for random within-dataset variation.[10] This way, we separate the effects of interest (i.e., each β_{0r}) from the dataset-specific variations (i.e., each u_{0rd}). The effect of the number of sources is similarly defined as the sum of a fixed representation-specific coefficient β_{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*.

[10] 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, σ_{0r}^2 (6 in total).

Figure 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 *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 the size of the network, 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:

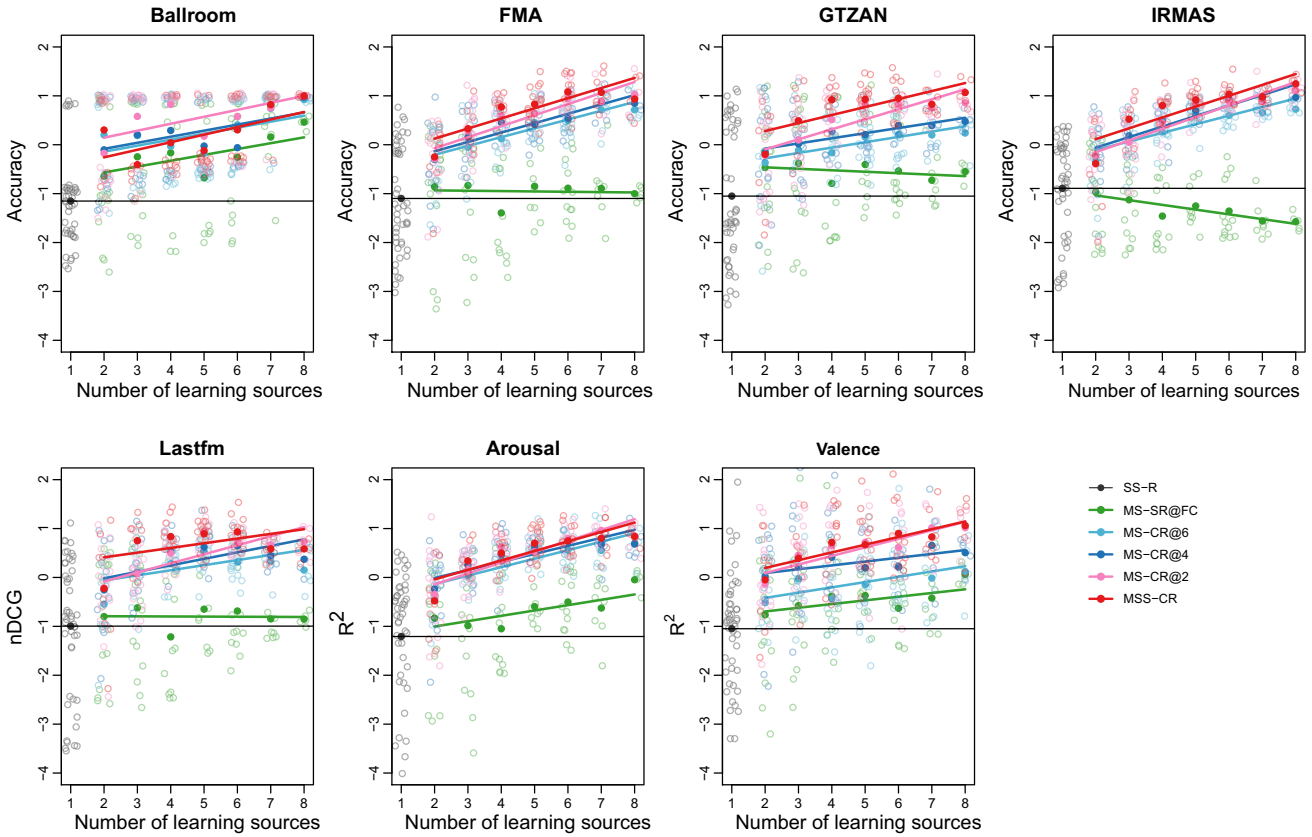


Fig. 10 (Standardized) performance by the number of learning sources. Solid points represent the mean per architecture and number of sources. The black horizontal line marks the mean performance of the *SS-R* representations. The colored lines show linear fits (color figure online)

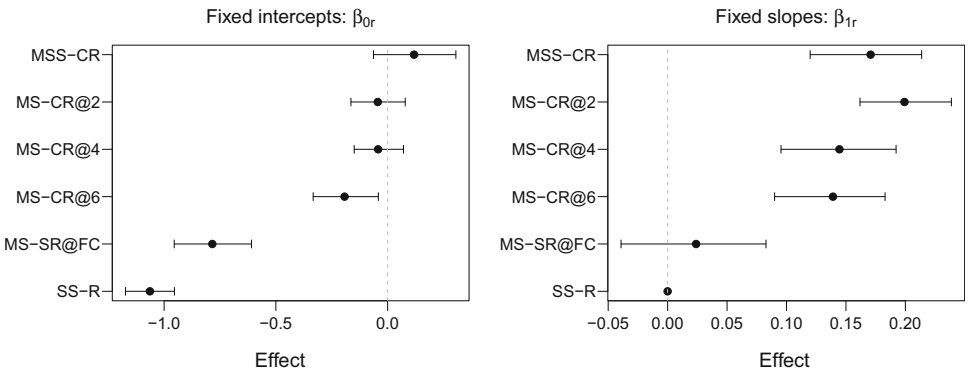


Fig. 11 Fixed effects and bootstrap 95% confidence intervals estimated for the first analysis model. The left plot depicts the effects of the representation strategy (β_{0r} intercepts), and the right plot shows the effects of the number of sources (β_{1r} slopes)

$$y_i^* = \beta_0 + \beta_{1d[i]} \cdot r_i^* + \beta_{2d[i]} \cdot n_i^* + \beta_{3d[i]} \cdot r_i^* \cdot n_i^* + e_i \quad e_i \sim N(0, \sigma_e^2) \tag{13}$$

$$\beta_{1d} = \beta_{10} + u_{1d} \quad u_{1d} \sim N(0, \sigma_1^2) \tag{14}$$

$$\beta_{2d} = \beta_{20} + u_{2d} \quad u_{2d} \sim N(0, \sigma_2^2) \tag{15}$$

$$\beta_{3d} = \beta_{30} + u_{3d} \quad u_{3d} \sim N(0, \sigma_3^2). \tag{16}$$

In contrast to the first model, there is no representation-specific fixed intercept but an overall intercept β_0. The effect of the network size is similarly modeled as the sum of an overall fixed slope β_{10} and a random dataset-specific effect u_{1d}. Likewise, this model includes the main effect of the number of sources (fixed effect β_{20}), as well as its interaction with the network size (fixed effect β_{30}). Figure 12 shows the fitted coefficients, confirming the statistically positive effect of the size of 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,

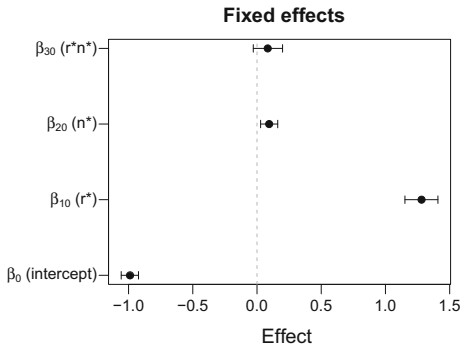


Fig. 12 Fixed effects and bootstrap 95% confidence intervals estimated for the second analysis model, depicting the overall intercept (β_0), the slope of the network size (β_{10}), the slope of the number of sources (β_{20}), and their interaction (β_{30})

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.

Figure 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 versus 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. Figure 13 presents similar

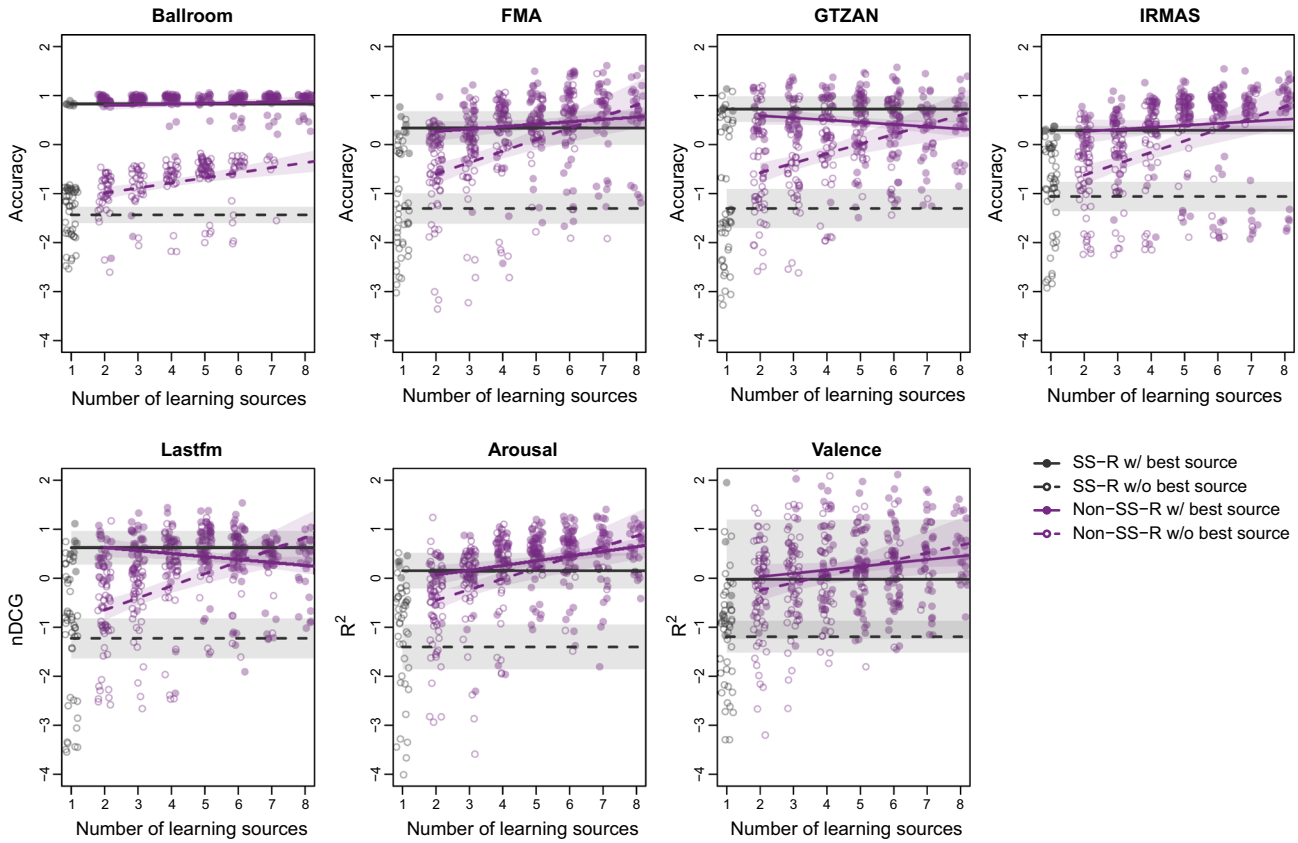


Fig. 13 (Standardized) performance by number of learning sources. Solid points mark representations including the source performing best with *SS-R* in the dataset; empty points mark representations without it. Solid and dashed lines represent linear fits, respectively; dashed areas represent 95% confidence intervals (color figure online)

Table 6 Variance components (as percent of total) of the learning sources, within each of the target datasets, and for non-*SS-R* representations

	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

Largest per-dataset in bold face

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 the largest variance are not necessarily the sources that obtain the best results by themselves in an *SS-R* representation (see Fig. 8). We examined this relationship further by calculating the correlation between variance components and (standardized) performance of the corresponding *SS-R*s. The Pearson correlation is 0.38, meaning that there is a mild association. Figure 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 more general representation based on multiple sources is a much simpler way to proceed, which still yields comparable or better results.

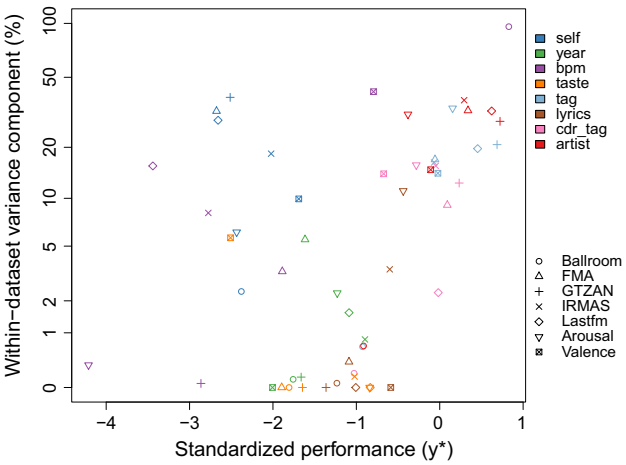


Fig. 14 Correlation between (standardized) *SS-R* performance and variance component (color figure online)

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* representations 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.

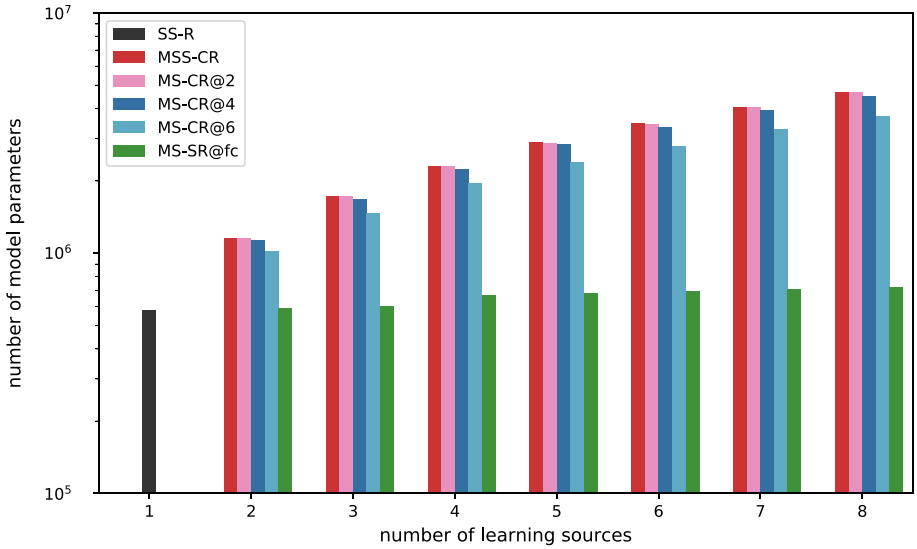


Fig. 15 Number of network parameters by number of learning sources

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 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.[11]

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 the 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

[11] Note that as soon as a pre-trained representation network model will be adapted to an 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.

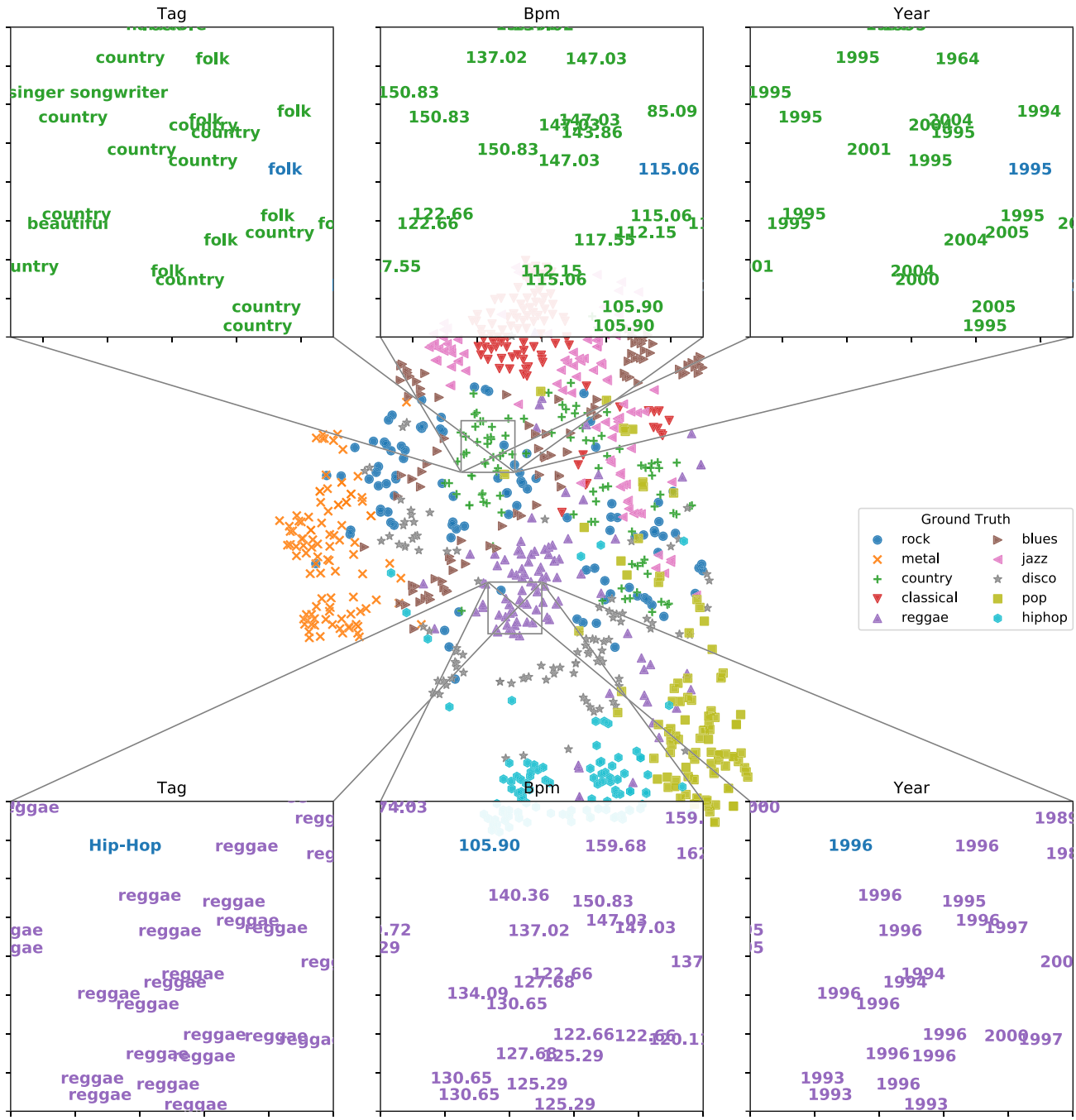


Fig. 16 Potential semantic explainability of DTMTL music representations. Here, we provide a visualization using t-SNE [88], plotting 2-dimensional coordinates of each sample from the GTZAN dataset, as resulting from an *MS-CR* representation trained on 5 sources. In the zoomed-in panes, we overlay the strongest topic model terms in z_t, for various types of learning sources. The specific model used in the visualization is the 232th model from the experimental design we introduce in Sect. 4.3, which is performing better than 95% of other models on GTZAN target dataset

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 considers the adaptation of the pre-trained network for further fine-tuning with respect to their target dataset. Since there are various choices to make, which will bring a substantial amount of variability, we decided to leave the aspects for 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 time-variant labels such as 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 improve this work.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Casey MA, Veltkamp RC, Goto M, Leman M, Rhodes C, Slaney M (2008) Content-based music information retrieval: current directions and future challenges. Proc IEEE 96(4):668–696. https://doi.org/10.1109/JPROC.2008.916370
2. Caruana R (1997) Multitask learning. Mach Learn 28(1):41–75. https://doi.org/10.1023/A:1007379606734. ISSN: 1573-0565
3. Bengio Y, Courville AC, Vincent P (2013) Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell 35(8):1798–1828. https://doi.org/10.1109/TPAMI.2013.50. ISSN: 0162-8828
4. Liu W, Mei T, Zhang Y, Che C, Luo J (2015) Multi-task deep visual-semantic embedding for video thumbnail selection. In: IEEE conference on computer vision and pattern recognition CVPR, Boston, MA, USA, pp 3707–3715. https://doi.org/10.1109/CVPR.2015.7298994
5. Bingel J, Søgaard A (2017) Identifying beneficial task relations for multi-task learning in deep neural networks. In: Proceedings of the 15th conference of the European chapter of the association for computational linguistics, vol 2. Association for Computational Linguistics, Valencia, Spain, pp 164–169
6. Li S, Liu Z-Q, Chan AB (2015) Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network. Int J Comput Vis 113(1):19–36. https://doi.org/10.1007/s11263-014-0767-8. ISSN: 1573-1405
7. Zhang W, Li R, Zeng T, Sun Q, Kumar S, Ye J, Ji S (2015) 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 KDD, Sydney. ACM, NSW, Australia, pp 1475–1484. https://doi.org/10.1145/2783258.2783304. ISBN: 978-1-4503-3664-2
8. Zhang Z, Luo Z, Loy CC, Tang X (2014) Facial landmark detection by deep multi-task learning. In: Computer vision—ECCV 13th European conference, proceedings, part VI. Springer, Zurich, Switzerland, pp 94–108. https://doi.org/10.1007/978-3-319-10599-4_7
9. Kaiser L, Gomez AN, Shazeer N, Vaswani A, Parmar N, Jones L, Uszkoreit J (2017) One model to learn them all. arXiv:abs/1706.05137
10. Rick Chang J-H, Li C-L, Póczos B, Vijaya Kumar BVK (2017) One network to solve them all—solving linear inverse problems using deep projection models. In: IEEE international conference on computer vision, ICCV. IEEE Computer Society, Venice, Italy, pp 5889–5898. https://doi.org/10.1109/ICCV.2017.627
11. Weston J, Bengio S, Hamel P (2011) Multi-tasking with joint semantic spaces for large-scale music annotation and retrieval. J New Music Res 40(4):337–348. https://doi.org/10.1080/09298215.2011.603834
12. Aytar Y, Vondrick C, Torralba A (2016) Soundnet: Learning sound representations from unlabeled video. In: Advances in neural information processing systems 29: annual conference on neural information processing systems. Barcelona, Spain, pp 892–900
13. Hamel P, Eck D (2010) Learning features from music audio with deep belief networks. In: Proceedings of the 11th international society for music information retrieval conference, ISMIR. Utrecht, Netherlands, pp 339–344
14. Boulanger-Lewandowski N, Bengio Y, Vincent P (2012) 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. Omnipress, Edinburgh, Scotland, UK
15. Schlüter J, Böck S (2014) Improved musical onset detection with convolutional neural networks. In: IEEE international conference

on acoustics, speech and signal processing, ICASSP. IEEE, Florence, Italy, pp 6979–6983. https://doi.org/10.1109/ICASSP.2014.6854953
16. Choi K, Fazekas G, Sandler MB (2016) Automatic tagging using deep convolutional neural networks. In: Proceedings of the 17th international society for music information retrieval conference, ISMIR. New York City, USA, pp 805–811
17. van den Oord A, Dieleman S, Schrauwen B (2013) Deep content-based music recommendation. In: Advances in neural information processing systems 26 NIPS. Lake Tahoe, NV, USA, pp 2643–2651
18. Chandna P, Miron M, Janer J, Gómez E (2017) Monoaural audio source separation using deep convolutional neural networks. In: Latent variable analysis and signal separation—13th international conference, LVA/ICA, Proceedings. Grenoble, France, pp 258–266. https://doi.org/10.1007/978-3-319-53547-0_25. ISBN: 978-3-319-53547-0
19. Jeong I-Y, Lee K (2016) Learning temporal features using a deep neural network and its application to music genre classification. In: Proceedings of the 17th international society for music information retrieval conference, ISMIR. New York City, USA, pp 434–440
20. Han Y, Kim J-H, Lee K (2017) Deep convolutional neural networks for predominant instrument recognition in polyphonic music. IEEE/ACM Trans Audio Speech Lang Process 25(1):208–221. https://doi.org/10.1109/TASLP.2016.2632307. ISSN: 2329-9290
21. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: 3th international conference on learning representations, ICLR, San Diego, CA, USA
22. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: IEEE conference on computer vision and pattern recognition, CVPR. IEEE Computer Society, Las Vegas, NV, USA, pp 770–778. https://doi.org/10.1109/CVPR.2016.90
23. Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: IEEE conference on computer vision and pattern recognition, CVPR. IEEE Computer Society, Boston, MA, USA, pp 1–9. https://doi.org/10.1109/CVPR.2015.7298594
24. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems 26 NIPS. Lake Tahoe, NV, USA, pp 3111–3119
25. Dieleman S, Brakel P, Schrauwen B (2011) Audio-based music classification with a pretrained convolutional network. In: Proceedings of the 12th international society for music information retrieval conference, ISMIR. University of Miami, Miami, FL, USA. pp 669–674. ISBN: 9780615548654
26. Choi K, Fazekas G, Sandler MB, Cho K (2017) Transfer learning for music classification and regression tasks. In: Proceedings of the 18th international society for music information retrieval conference, ISMIR. Suzhou, China, pp 141–149
27. van den Oord A, Dieleman S, Schrauwen B (2014) Transfer learning by supervised pre-training for audio-based music classification. In: Proceedings of the 15th international society for music information retrieval conference, ISMIR. Taipei, Taiwan, pp 29–34
28. Liang D, Zhan M, Ellis DPW (2015) Content-aware collaborative music recommendation using pre-trained neural networks. In: Proceedings of the 16th international society for music information retrieval conference, ISMIR. Málaga, Spain, pp 295–301
29. Misra I, Shrivastava A, Gupta A, Hebert M (2016) Cross-stitch networks for multi-task learning. In: IEEE conference on computer vision and pattern recognition. CVPR. IEEE Computer Society, Las Vegas, NV, USA, pp 3994–4003
30. Bertin-Mahieux T, Ellis DPW, Whitman B, Lamere P (2011) The million song dataset. In: Proceedings of the 12th international society for music information retrieval conference, ISMIR. University of Miami, Miami, FL, USA. pp 591–596
31. Bengio Y, Lamblin P, Popovici D, Larochelle H (2006) Greedy layer-wise training of deep networks. In: Advances in neural information processing systems 19. NIPS. MIT Press, Vancouver, BC, Canada, pp 153–160
32. Vincent P, Larochelle H, Bengio Y, Manzagol P-A (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on machine learning ICML. ACM, Helsinki, Finland, pp 1096–1103. https://doi.org/10.1145/1390156.1390294
33. Smolensky P (1986) Information processing in dynamical systems: Foundations of harmony theory. Technical report, University of Colorado, Boulder, Department of Computer Science
34. Hinton GE, Osindero S, Teh Y-W (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527
35. Goodfellow I, Pouget-Abadie J, Mirza M, Bing X, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems 27. NIPS. Curran Associates Inc., Montreal, QC, Canada, pp 2672–2680
36. Han X, Leung T, Jia Y, Sukthankar R, Berg AC (2015) Matchnet: unifying feature and metric learning for patch-based matching. In: IEEE conference on computer vision and pattern recognition, CVPR. IEEE Computer Society, Boston, MA, USA, pp 3279–3286. https://doi.org/10.1109/CVPR.2015.7298948
37. Arandjelovic R, Zisserman A (2017) Look, listen and learn. In: IEEE international conference on computer vision, ICCV. IEEE Computer Society, Venice, Italy, pp 609–617. https://doi.org/10.1109/ICCV.2017.73
38. Huang Y-S, Chou S-Y, Yang Y-H (2018) Generating music medleys via playing music puzzle games. In: Proceedings of the thirty-second conference on artificial intelligence, AAAI. AAAI Press, New Orleans, LA, USA, pp 2281–2288
39. Salton G, McGill M (1984) Introduction to modern information retrieval. McGraw-Hill Book Company, New York City. ISBN: 0-07-054484-0
40. Lamere P (2008) Social tagging and music information retrieval. J New Music Res 37(2):101–114. https://doi.org/10.1080/09298210802479284. ISSN: 0929-8215
41. Hamel P, Davies MEP, Yoshii K, Goto M (2013) 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. Curitiba, Brazil, pp 9–14
42. Law E, Settles B, Mitchell TM (2010) Learning to tag from open vocabulary labels. In: Machine learning and knowledge discovery in databases, European conference, ECML PKDD, Proceedings. Part II. Springer, Barcelona, Spain, pp 211–226
43. Hofmann T (1999) Probabilistic latent semantic analysis. In: UAI: proceedings of the fifteenth conference on uncertainty in artificial intelligence. Morgan Kaufmann, Stockholm, Sweden, pp 289–296
44. Schlüter J (2016) Learning to pinpoint singing voice from weakly labeled examples. In: Proceedings of the 17th international society for music information retrieval conference, ISMIR. New York City, USA, pp 44–50
45. Hershey S, Chaudhuri S, Ellis DPW, Gemmeke JF, Jansen A, Moore RC, Plakal M, Platt D, Saurous RA, Seybold B, Slaney M, Weiss RJ, Wilson KW (2017) CNN architectures for large-scale audio classification. In: IEEE international conference on acoustics, speech and signal processing, ICASSP. IEEE, New

Orleans, LA, USA, pp 131–135. https://doi.org/10.1109/ICASSP.2017.7952132
46. Lee H, Pham PT, Largman Y, Ng AY (2009) Unsupervised feature learning for audio classification using convolutional deep belief networks. In: Advances in neural information processing systems 22. NIPS. Curran Associates Inc, Vancouver, BC, Canada, pp 1096–1104
47. Humphrey EJ, Bello JP (2012) Rethinking automatic chord recognition with convolutional neural networks. In: 11th international conference on machine learning and applications, ICMLA. IEEE, Boca Raton, FL, USA, pp 357–362. https://doi.org/10.1109/ICMLA.2012.220
48. Nakashika T, Garcia C, Takiguchi T (2012) Local-feature-map integration using convolutional neural networks for music genre classification. In: INTERSPEECH, 13th annual conference of the international speech communication association. ISCA, Portland, OR, USA, pp 1752–1755
49. Ullrich K, Schlüter J, Grill T (2015) Boundary detection in music structure analysis using convolutional neural networks. In: Proceedings of the 16th international society for music information retrieval conference, ISMIR. Málaga, Spain, pp 417–422
50. Piczak KJ (2015) Environmental sound classification with convolutional neural networks. In: 25th IEEE international workshop on machine learning for signal processing, MLSP. IEEE, Boston, MA, USA, pp 1–6. https://doi.org/10.1109/MLSP.2015.7324337
51. Simpson AJR, Roma G, Plumbley MD (2015) 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, Proceedings. Springer, Liberec, Czech Republic, pp 429–436. https://doi.org/10.1007/978-3-319-22482-4_50. ISBN: 978-3-319-22482-4
52. Phan H, Hertel L, Maaß M, Mertins A (2016) Robust audio event recognition with 1-max pooling convolutional neural networks. In: INTERSPEECH 17th annual conference of the international speech communication association. ISCA, San Francisco, CA, USA, pp 3653–3657. https://doi.org/10.21437/Interspeech.2016-123
53. Pons J, Lidy T, Serra X (2016) Experimenting with musically motivated convolutional neural networks. In: 14th international workshop on content-based multimedia indexing, CBMI. IEEE, Bucharest, Romania, pp 1–6. https://doi.org/10.1109/CBMI.2016.7500246
54. Stasiak B, Monko J (2016) Analysis of time-frequency representations for musical onset detection with convolutional neural network. In: Proceedings of the federated conference on computer science and information systems, FedCSIS. Gdańsk, Poland, pp 147–152. https://doi.org/10.15439/2016F558
55. Su H, Zhang H, Zhang X, Gao G (2016) Convolutional neural network for robust pitch determination. In: IEEE international conference on acoustics, speech and signal processing, ICASSP. IEEE, Shanghai, China. pp 579–583. https://doi.org/10.1109/ICASSP.2016.7471741
56. Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84–90. https://doi.org/10.1145/3065386
57. Dieleman S, Schrauwen B (2014) End-to-end learning for music audio. In: IEEE international conference on acoustics, speech and signal processing, ICASSP. IEEE, Florence, Italy, pp 6964–6968. https://doi.org/10.1109/ICASSP.2014.6854950
58. van den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior AW, Kavukcuoglu K (2016) Wavenet: a generative model for raw audio. In: The 9th ISCA speech synthesis workshop, SSW. ISCA, Sunnyvale, CA, USA, p 125
59. Jaitly N, Hinton GE (2011) Learning a better representation of speech soundwaves using restricted boltzmann machines. In: IEEE international conference on acoustics, speech, and signal processing, ICASSP. IEEE, Prague, Czech Republic, pp 5884–5887. https://doi.org/10.1109/ICASSP.2011.5947700
60. Lee J, Park J, Kim KL, Nam J (2017) Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. In: 14th sound and music computing conference, SMC, Espoo, Finland
61. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd international conference on machine learning, ICML. JMLR, Inc, Lille, France, pp 448–456
62. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning ICML. Omnipress, Haifa, Israel, pp 807–814
63. Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
64. Nam J, Herrera J, Slaney M, Smith JO (2012) Learning sparse feature representations for music annotation and retrieval. In: Proceedings of the 13th international society for music information retrieval conference, ISMIR. FEUP Edições, Porto, Portugal, pp 565–570
65. Choi K, Fazekas G, Sandler MB, Cho K (2018) A comparison of audio signal preprocessing methods for deep neural networks on music tagging. In: 26th European signal processing conference. EUSIPCO. IEEE, Roma, Italy, pp 1870–1874
66. Dörfler M, Grill T, Bammer R, Flexer A (2018) Basic filters for convolutional neural networks applied to music: training or design? Neural Comput Appl https://doi.org/10.1007/s00521-018-3704-x. ISSN: 1433-3058
67. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: 3th International conference on learning representations, ICLR, San Diego, CA, USA
68. Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A (2017) Automatic differentiation in PyTorch. In: NIPS-W
69. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay D (2012) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830. https://doi.org/10.1007/s13398-014-0173-7.2. ISSN: 15324435
70. McFee B, Raffel C, Liang D, Ellis DPW, McVicar M, Battenberg M, Nieto O (2015) librosa: audio and music signal analysis in python. In: Kathryn H, James B (eds) Proceedings of the 14th python in science conference SciPy. Austin, TX, USA, pp 18 – 24. https://doi.org/10.25080/Majora-7b98e3ed-003
71. Defferrard M, Benzi K, Vandergheynst P, Bresson X (2017) FMA: a dataset for music analysis. In: Proceedings of the 18th international society for music information retrieval conference, ISMIR. Suzhou, China, pp 316–323
72. Tzanetakis G, Cook PR (2002) Musical genre classification of audio signals. IEEE Trans Speech Audio Process 10(5):293–302. https://doi.org/10.1109/TSA.2002.800560. ISSN: 1063-6676
73. Kereliuk C, Sturm BL, Larsen J (2015) Deep learning and music adversaries. IEEE Trans Multimed 17(11):2059–2071. https://doi.org/10.1109/TMM.2015.2478068. ISSN: 1520-9210
74. Fabien G, Anssi K, Simon D, Alonso M, George T, Uhle C, Pedro C (2006) An experimental comparison of audio tempo induction algorithms. IEEE Trans Audio Speech Lang Process 14(5):1832–1844. https://doi.org/10.1109/TSA.2005.858509. ISSN: 1558-7916
75. Marchand U, Peeters G (2016) 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. IEEE, Salerno, Italy, pp 1–6. https://doi.org/10.1109/MLSP.2016.7738904
76. Bosch JJ, Janer J, Fuhrmann F, Herrera P (2012) 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. FEUP Edições, Porto, Portugal, pp 559–564
77. Soleymani M, Caro MN, Schmidt EM, Sha C-Y, Yang Y-H (2013) 1000 songs for emotional analysis of music. In: Proceedings of the 2nd ACM international workshop on crowdsourcing for multimedia CrowdMM@ACM multimedia. ACM, Barcelona, Spain, pp 1–6. https://doi.org/10.1145/2506364.2506365. ISBN: 978-1-4503-2396-3
78. Òscar C (2010) Music recommendation and discovery–the long tail, long fail, and long play in the digital music space. Springer, Berlin. https://doi.org/10.1007/978-3-642-13287-2. ISBN: 978-3-642-13286-5
79. Sturm BL (2014) The state of the art ten years after a state of the art: future research in music information retrieval. J New Music Res 43(2):147–172. https://doi.org/10.1080/09298215.2014.894533
80. Sturm BL (2016) The "Horse" inside: seeking causes behind the behaviors of music content analysis systems. Comput Entertain 14(2):3:1–3:32. https://doi.org/10.1145/2967507
81. Jonathan P, Russell James A, Peterson Bradley S (2005) The circumplex model of affect: an integrative approach to affective neuroscience, cognitive development, and psychopathology. Dev Psychopathol 17(3):715734. https://doi.org/10.1017/S0954579405050340. ISSN: 1469-2198
82. Montgomery DC (2012) Design and analysis of experiments, 8th edn. Wiley, Hoboken
83. Goos P, Jones B (2011) Optimal design of experiments: a case study approach, 1st edn. Wiley, Hoboken
84. Hinton GE (1989) Connectionist learning procedures. Artif Intell 40(1):185–234. https://doi.org/10.1016/0004-3702(89)90049-0. ISSN: 0004-3702
85. Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: Proceedings of the 8th IEEE international conference on data mining (ICDM). IEEE Computer Society, Pisa, Italy, pp 263–272. https://doi.org/10.1109/ICDM.2008.22
86. Gelman A, Hill J (2006) Data analysis using regression and multilevel/hierarchical models. Cambridge University Press, Cambridge
87. Searle SR, Casella G, McCulloch CE (2006) Variance components. Wiley, Hoboken
88. van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9(November):2579–2605