MIDI2vec: Learning MIDI Embeddings for Reliable Prediction of Symbolic Music Metadata

Pasquale Lisena [a,*], Albert Meroño-Peñuela [b] and Raphaël Troncy [a]
[a] *EURECOM, Sophia Antipolis, France*
E-mails: pasquale.lisena@eurecom.fr, raphael.troncy@eurecom.fr
[b] *King's College London, United Kingdom*
E-mail: albert.merono@kcl.ac.uk

Abstract. An important problem in large symbolic music collections is the low availability of high quality metadata, which is essential for various information retrieval tasks. Traditionally, systems have addressed this by relying either in costly human annotations or in rule-based systems at limited scale. Recently, embedding strategies have been exploited for representing latent factors in graphs of connected nodes. In this work, we propose MIDI2vec, a new approach for representing MIDI files as vectors based on graph embedding techniques. Our strategy consists of representing the MIDI data as a graph, including the information about tempo, time signature, programs and notes. Next, we run and optimise node2vec for generating embeddings using random walks in the graph. We demonstrate that the resulting vectors can successfully be employed for predicting the musical genre and other metadata such as the composer, the instrument or the movement. In particular, we conduct experiments using those vectors as input to a Feed-Forward Neural Network and we report good comparable accuracy scores in the prediction with respect to other approaches relying purely on symbolic music, avoiding feature engineering and producing highly scalable and reusable models with low dimensionality. Our proposal has real-world applications in automated metadata tagging for symbolic music, for example in digital libraries for musicology, datasets for machine learning, and knowledge graph completion.

Keywords: music, metadata, metadata prediction, graph embeddings, neural networks

1. Introduction

High-quality metadata is a prerequisite in many music information retrieval (MIR) tasks, like accessing symbolic music collections, music recommender systems and discovery [1]. Historically, the availability of this metadata has depended on manual human annotation labour, which is typically costly. Consequently, several systems have been proposed that automatically analyse music in order to annotate high-level features, some being abstract enough to be close to traditional metadata [2]. Most of this systems target the so-called *symbolic representation* of music: this representation explicitly describes the notes and their properties – timbre, tempo, velocity, etc. – on individual tracks for the different instruments, as opposed to the digital audio which encodes a sampled musical signal (i.e. a recording). Due to the need of high-quality annotated datasets, these systems have many real-world applications. For example, digital libraries for musicology can use them for automatically tagging metadata, lowering manual annotation costs and improving results of music information retrieval systems. In recently proposed music knowledge graphs [3], these systems could be used to complete missing information in the graph. Another application is in machine learning for music, which needs large amounts of music data annotations that could be provided by such systems, for applications such as data programming [4] or weak supervision [5] or even music recommender systems based on similarity.

*Corresponding author. E-mail: pasquale.lisena@eurecom.fr.

In this work, we focus on symbolic music in MIDI format [6] due to its high availability on the Web and its popularity in tasks like music generation [7] and music knowledge graphs [3]. This popularity has also given birth to large MIDI datasets that need, besides high-quality annotations, scalable approaches producing them. Despite their natural fit as a transcription format for music, MIDI files alone have a number of limitations, all concerning the fact that track information (instruments, vocals, metadata) is not standardized and often only implicit [8]. One way of addressing this is to express MIDI information using RDF and ontologies [3], allowing for the re-use and standardization of missing MIDI metadata at Web scale. The Lakh MIDI dataset [8] is another important MIR benchmark used in e.g. automatic audio transcription; however, only 31,034 of its 176,581 MIDI files (17.57%) are aligned with metadata from the Million Song Dataset [9].

Reconstructing this kind of high-level metadata from symbolic musical content, automatically and at scale, is a challenging task due to the existing semantic gap between the desired metadata and low-level music descriptors [10]. One way of addressing this is by analysing the content of the symbolic notation for predicting higher-level metadata, i.e. identifying symbolic patterns in melody, harmony, rhythm, structure, etc. that are characteristic of a certain genre or composer. So far, research has mainly focused on genre [11, 12], emotion, and composer [13] classification mainly using supervised machine learning techniques. However, traditional machine learning algorithms are limited by the need to perform some feature selection in the so-called feature engineering process [14].

In order to overcome this, embedding-based methods have been proposed. Embeddings are mappings that transform the symbolic representations of discrete variables (elements which cannot be naturally ordered such as words or songs) into numeric vectors. Each dimension of an embedding vector represents a latent feature which has been automatically learned. Avoiding more expensive representations such as the one-hot encoding, embedding vectors are useful at reducing the dimensionality of the input data of neural networks, a typical choice to address a learning task such as automatic classification. Such embedding-based methods have been successfully applied to textual [15] and graph data [16] and notably also on MIDI data for the task of automated music generation through Recurrent Neural Networks (RNN) [17, 18] and self-learning techniques such as Variational Autoencoders (VAE) [7]. Graph embeddings have become a widely used and effective way to represent graph information in a way neural networks can easily process it [19].

Some latent, hidden features that lie in the data might be hard, if not impossible, for a human engineer to discover and model [20]; some music information tends to be ambiguous and loosely defined (e.g. "Allegro", "Prelude"), and therefore a more fuzzy representation can constitute an advantage; besides, current feature-engineered methods have pitfalls in scalability with respect to the dataset size and the number of metadata classes to predict [21]. However, feature engineering is still generally faster than learning latent features from data, since it is based on given knowledge; and it is easier to debug because engineered features are human-understandable. Moreover, embeddings can certainly overfit models as well [22]. Therefore, one of the aims of this work is to gain a better understanding of how feature-based and embedding-based techniques compare and perform at scale in the task of symbolic music metadata prediction.

In this paper, we propose MIDI2vec, a method for representing MIDI data as vector-space embeddings for automated metadata classification. First, we express MIDI files as graphs, assigning unique identifiers to specific characteristics and their values such as tempo, time signature, programs (instruments), and notes. Therefore, two MIDI files will be connected in the graph if they share the same resources (e.g. same instruments, chords, tempo, etc.). Second, we use graph embedding techniques – and in particular the *node2vec* algorithm [16] – to traverse these MIDI graphs with random walks, and represent the information of the traversed paths as numeric vectors. We assume that these traversals will encode not only MIDI information that is relevant for a given song, but also additional neighbouring information of similar songs that can be relevant for metadata classification.[1] In other words, we assume the distributional semantics hypothesis [23] over MIDI features: notes or groups of notes used in similar contexts will tend to have similar meanings, and in particular, will be associated with similar features, even for high-level metadata such as genre, composer or instrument. More specifically, our contributions are:

[1] We leave the interesting alternative of traversing these MIDI files sequentially, i.e. following the trail of temporal event occurrence instead of contextual co-occurrence, as future work.

– the conceptualisation of relevant symbolic features (pitch, timbre, tempo, time signature) of MIDI space into *graph space*;
– the systematic application of a well-known graph embedding generation method to generate *MIDI embeddings*;
– the use of such learned embeddings to predict metadata for three datasets, achieving comparable accuracy to symbolic feature-based approaches without the need of feature engineering, scaling to Web-size datasets, and with one order of magnitude less dimensions.

To the best of our knowledge, this is the first time that graph embedding approaches are used for representing a whole symbolic music track, and for reliably predicting symbolic music metadata.

The rest of the paper is organised as follows. In Section 2, we survey related work. In Section 3 we briefly introduce graph embeddings. In Section 4, we describe our strategy to extract relevant symbolic data from MIDI files, to represent them as graphs, and to use these graphs to build MIDI embeddings. In Section 5, we run an experiment to predict genre and other metadata on three different datasets using the MIDI embeddings. Finally, we conclude and outline some future works in Section 6.

2. Related Work

The extraction of high-level metadata from musical content is a long-standing goal of the MIR community [1], and one of the purposes of the Essentia library [2]. This is a first task to fulfil towards an automatic reconstruction of music metadata, although the semantic gap [10] between content and metadata shows that purely bottom-up approaches are hard. Nevertheless, automating the generation of high-quality metadata would be a great benefit to tasks like music knowledge graph completion and music emotion detection, for which MIDI has already been used [3, 24].

In [25], a comprehensive study surveys techniques for genre classification based on symbolic music. Although the performance scores of the state-of-the-art set the baseline to outperform, many are computed on different sets of classes, on monophonic MIDI, or on genre-specific datasets (e.g. folk music). Nonetheless, the survey finds a large number of methods based on machine learning. For example, the unsupervised nearest neighbours (NN) and k-nearest neighbours (kNN) is applied to genre prediction from MIDI in [11]. This work is further extended in [12] with linear discriminant classifiers (LDN) and by combining MIDI and audio features. Different data sources – audio, symbolic music, lyrics – are instead combined in [21]. However, these classical machine learning approaches suffer from the need for feature selection, which is costly and can overfit models [14].

Recent developments in neural networks have boosted work in vector space-based music metadata classification, using vectors computed from the audio signal. Some examples are genre-agnostic key classification [26] and jazz solo instrument classification [27]. However, these approaches still tackle the classification of mostly content-based features (e.g. timbre), and not high-level metadata (e.g. genre).

Both feature engineering – common in pre-deep machine learning and purely symbolic approaches – and vector-space based methods have advantages and limitations. Model overfitting might happen in both [14, 22]. Because it is based on provided knowledge, feature engineering is faster than learning features from data – a process than can be computationally expensive – and is easily understandable to humans due to its intrinsic symbolic representation. On the other hand, vector space representations have the advantage of capturing latent features that might be hard or impossible for humans to describe symbolically; a more fuzzy representation can constitute an advantage when music information is ambiguous and loosely defined (e.g. “Allegro”, “Prelude”). [20]; and scale very well to large datasets and number of classification classes [21].

While recent MIR research relies mostly on audio analysis for metadata prediction, symbolic notation is largely used for automated music generation with these kinds of models. An example is MusicVAE [7], a hierarchical variational autoencoder (VAE) that learns a latent space of musical sequences. Similarly, in [17] MIDI embeddings are employed for automatic music generation, representing all the notes played together at regular time steps. In [18], MIDI files are used for learning a set of embeddings representing different aspects of a pitch, during the training of an RNN for music generation. *BachProp* [28], an approach for music score generation that relies on an architecture which combines Gated-Recurrent Units (GRU), receives in input MIDI notes in the one-hot encoding format.

Vector embedding similarities on various semantic descriptors are applied in music recommendation in [29]. These vector space representations, which tie re-

lated artists, works and performances closer, eventually surface terminologies and ultimately linked vocabularies for music metadata [30].

While embedding-based approaches provide different modes of interactive musical creation, they do not require feature selection and can be mapped to other latent spaces – e.g. texts [31] or documents (the homonym *Midi2Vec*[2]) – and none so far address the task of metadata classification specifically. In a more related work towards an embedding-based symbolic music metadata classification, MIDI-glove[3] produces embeddings of notes from monophonic MIDI, but its consideration of MIDI note values leaves out some information such as timing and rhythm, therefore producing representations of a single feature (pitch) instead of the whole MIDI content.

3. Graph Embeddings

Graph embeddings are the result of the transposition of word embedding techniques – notably word2vec [15] and GloVe [32] – to networks. According to [33], a graph can be defined as a set $G = (V, E)$, where V is a set of vertices (or nodes) and E is the set of edges, represented as pairs of directly connected vertices. Graph embedding algorithms produce a mathematical representation – consisting of a set of vectors – of the content of the graph, which is much more compact than other kinds of representation (e.g. adjacency matrix) and consequently easier and faster to process with Machine Learning. The effectiveness of these techniques makes them very popular in different applications, from classification to recommendation, with an interesting number of algorithms developed for their computation. An extensive survey has been realised by [34].

In 2014, Perozzi et al. published **Deep Walk** [35]. The core idea of this work consists in the use of random walks in the graph in order to generate sequences of nodes. The number and the length of link paths between two nodes impacts on the probability of those two nodes to be selected together in the random walk. In other words, the more two nodes share connections[4] in the graph and the fewer edges compose those connections, the more those nodes will appear together in several walks. According to the intuition of the authors, we can deal with nodes in sequences as they are words in sentences, so it is possible to apply word embedding models, and by extension, the distributional semantics hypothesis, to those sequences. The transition probabilities between nodes replace the one between words in the embedding computation. The result is a vector space in which distances in the graph are kept.

DeepWalk has been extended by **node2vec** [16], with the inclusion of two parameters P and Q, which rule on the generation of random walks. In particular, the parameter P impacts on the probability that the random walk immediately revisits the previous node. The parameter Q controls the probability that the random walk moves towards increasingly further away nodes, enabling to discover peripheral parts of the graph. In other words, higher values of P promote random walks that explore a local neighbourhood around the starting node, while high values of Q encourage walks that cover wider areas of the graph. *Node2vec* can be also applied to weighted graphs, in which the weight of an edge affects the probability that it participates to the walk.

Other notable embedding-based techniques have been proposed for representing nodes in a graph, such as rdf2vec [19], entity2vec [36] and graph2vec [37] for graph embeddings, and many others[5].

4. Learning MIDI Embeddings

The MIDI format does not present a graph structure, but it consists of a time-based linear succession of events, called *MIDI messages*, detailed in the specification [6]. Some examples are *Note On* and *Note Off* for representing played notes, *Program Change* for setting the instrument, and *MTC Quarter Frame Message* for specifying the playing speed according to the MIDI Time Code (MTC) protocol. This last information impacts on the duration of the interval between two Song Position Pointers (SPP), which identify the time at which the message occurs, expressed in MIDI beats from the beginning of the song (commonly referred to as *ticks*). Some of the MIDI messages are referred to a specific channel (a single device emitting

[2] https://github.com/TaylorPeer/Midi2Vec

[3] https://github.com/brangerbriz/midi-glove

[4] Two nodes are connected if exists one or more paths of edges between them, of which the two edges represent respectively the first and the last node involved. A connection can consist of a single edge; in this case, we can speak about directly connected nodes.

[5] A regularly updated list of software for embedding generation is available at https://github.com/MaxwellRebo/awesome-2vec

music, in other words a single instruments), while others can apply to the whole MIDI. Because of this structure, we first need to convert the MIDI into a graph, on which embeddings can be computed afterwards.

4.1. MIDI to Graph

We propose a preliminary conversion of a MIDI file into a graph. As shown in Figure 1, a *MIDI node* (the circle) represents the MIDI file and will be connected to nodes representing different parts of the MIDI content (i.e. tempo, programs, time signature, notes). A MIDI node can be linked to one or more nodes for each type.

In the context of graph embeddings computation, the practice is to take into account only connections between entities, that are nodes represented by identifiers. Literal values (text, numbers, etc.) are normally ignored [19] or some shrewdness is applied such as the use of contiguity windows [38]. In fact, literals can increase uncontrollably the number of nodes, in particular in presence of continuous values or entity-specific textual annotation. The effect is a very sparse graph[6], causing an exponential increment of the computation time and poor performance [39]. In our case, the crucial information represented as continuous data (e.g. the tempo) can not be excluded from the embeddings. We opted for partitioning the continuous values in ranges, in order to insert their information in the graph, while limiting at the same time the number of nodes. We provide some details for each type of node in the following.

Tempo, computed in *bpm* (beats per minute). This value is computed from the MIDI tempo field (in microseconds per beat), according to the formula:

$$Tempo_{bpm} = 60000000/Tempo_{midi} \quad (1)$$

The continuous values are then discretised in partitions, each one representing a range of 10 bpm. Example: `tempo-11` represents the range of values 110 ± 5 bpm.

Programs, representing the timbre of the channels, among the 128 different standard programs[7]. Example: `program-0` is the Acoustic Grand Piano.

Time signature is the measure of how many beats are contained in each measure. It is represented as the concatenation of numerator and denominator. Example: $\frac{4}{4}$ is represented as `ts-4/4`.

Notes, representing the pitches in the MIDI. The information about duration and co-occurrence of notes (e.g. in a chord) are not directly represented in the MIDI file. The duration is extracted by comparing successive *NoteOn* and *NoteOff* events sharing the same pitch and located on the same channel. Co-occurring notes can be detected by comparing the same category of events among all channels, selecting the ones with overlapping *Song Position Pointers* (SPP). To include this information in the graph while limiting the number of nodes and edges, we extract all groups of notes starting (i.e. with a *NoteOn* message) at the same SPP. A tolerance of 10ms is applied for considering two notes simultaneous, in order to overcome eventual small differences due to MIDI recording. Each group is connected to:

- the maximum duration of the notes in the group, discretised in classes of 100ms. Example: the id `duration-3` represents the range 300 ± 50 ms.
- their average velocity. Example: `velocity-1`.
- all the pitches, identified by their standard MIDI numbers. Example: `note-57` is A-3.

Each group has an identifier which is deterministically computed from its content using a hash function. These groups are then linked to the relative MIDI node.

MIDI2vec does not encode any information concerning time, in particular about sequences of notes occurring in the same channel. This information is undoubtedly relevant and crucial in music representation. Nevertheless, we decide to not include the time dimension in this experiment for two main reasons. First, the inclusion of order and sequentiality in a graph representation is not very common and few works addressed this topic so far [40, 41]. Second, this would require to make some choices, among them the number of consecutive notes to be grouped and the opportunity of representing pauses in the graph or not. For this reason, we decided to consider the encoding of subsequent notes for future work.

The connections between nodes are mostly of type *many-to-many*, so that two involved nodes are potentially part of other instance of the same connection. Taking as an example the connections between Tempo and MIDI types, this means that a specific Tempo node may be linked to different MIDI – pieces sharing the same tempo – and a specific MIDI may be linked to

[6] A graph is considered *dense* or *sparse* if its number of edges is close or far, respectively, to the number of all potential edges connecting each pair of vertices [33].

[7] The full list of MIDI programs is available at https://jazz-soft.net/demo/GeneralMidi.html

different tempos – representing a tempo change in the track. Some connections can be instead of type *one-to-many*: this is the case of the Group of Notes, linked to exactly one Duration which is, in turn, connected to several Group of Notes. Two MIDI nodes are never directly connected. Tempo, programs, and time signature might change within a file; we register all of them as equal nodes.

Formally, we realise a graph $G = (V, E)$, where $V = M \cup C \cup A$. M corresponds to the set of MIDI files. C corresponds to the first level of MIDI content, including tempo, programs, time signature, and the set $N \in C$ referring to groups of notes. A corresponds to the attributes of groups of notes, in particular to duration, velocity, and pitch. The edges in $E = E_M \cup E_N$ can belong to two types. An edge $(m, c) \in E_M$ indicates that the MIDI $m \in M$ has $c \in C$ as part of its content; this category also includes edges (m, n) linking a MIDI m to a note group $n \in N \in C$. While $(n, a) \in E_N$ indicates that the group of notes $n \in N$ has the attribute $a \in A$.

In Figure 2, an excerpt of an example graph is provided, in particular showing the connections between MIDI files through complete note groups, single notes, duration, etc. In this kind of graph, two MIDI tracks sharing multiple chords (or other elements) will have more probability to appear in the same random walk. This representation aims to track at the same time the presence of specific chords and of quick and long notes, which can respectively characterise a more virtuous or lyrical composition.

The graph is saved in the edgelist format, which includes all couples of connected nodes. In the edgelist, each line of text represent an edge and contains the identifiers of the two involved nodes separated by blank spaces. In practice, the MIDI files are read and messages are sequentially converted and appended to the edgelist. This edgelist is the output of this first conversion and feeds the second part of the approach, described in the next section.

4.2. Graph to Vectors

Embeddings are computed on the output graph of the previous process with the *node2vec* algorithm. As more extensively written in Section 3, the algorithm simulates random walks on the graph and computes the transition probabilities between nodes, which are mapped into the vector space. In other words, two MIDI files sharing programs, tempos, note groups are more likely to be part of the same random walk and consequently are more likely to be close in the computed embedding space.

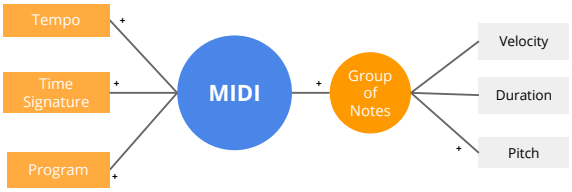


Fig. 1. Schema of the graph generated from MIDI. The + indicates edges representing connections of type many-to-many. The colours represent different groups of nodes: MIDI M (blue), Content C (orange) among which Notes N have a round shape, and Attributes A (grey).

In practice, each node in the graph is selected as the starting node for a random walk, occupying its first slot. The second slot will be occupied by one of the nodes directly linked to the first node, according to the probability function. Iteratively, every slot of the random walk will be occupied by one of the neighbours of the previous one. The number of walks to be produced for each node and their length are given parameters. These walks are then processed by *word2vec* as they are sentences.

Following this procedure, a 100-dimensions embedding vector is computed for each node (vertex) of the input graph. Each dimension of the vector cannot be attributed to a specific feature of the described item – for example, the tempo – but it rather represents latent features learned by the embedding algorithm. We apply a post-processing step in order to keep only the vectors $m \in M$ representing the MIDI files, excluding consequently all the nodes that refer to MIDI messages and attributes.

Such obtained vectors can be then used in input to any algorithm, in tasks such as classification, clustering, and others. In the experiment which will be detailed in the following section, we will use such generated vectors in input to a neural network for classification. In particular, all vectors used in our experiment have been computed using the following configuration of *node2vec*: walk length = 10, number of walks = 40, window size = 5, number of iterations = 5, p = 0.1, and q=0.1. We also publish as open-source the library for producing MIDI embeddings at https://git.io/midi2vec.

5. Evaluation

We evaluate this strategy in relation to three different goals, involving three different MIDI datasets.

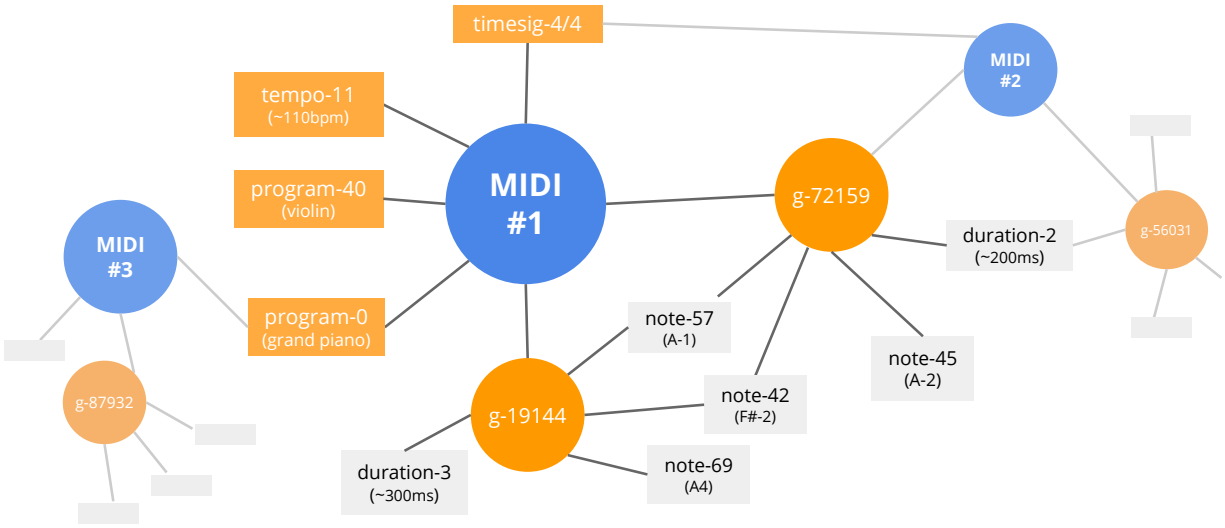


Fig. 2. MIDI graph example, showing possible connections between MIDI #1, #2, and #3.

These goals are detailed in next sections and consist respectively in predicting the genre, some high-quality metadata, and some user-defined tags.

For each goal, we perform an experiment which relies on a common procedure. MIDI embeddings are generated on the dataset using MIDI2vec. A Feed-Forward Neural Network receives the MIDI embeddings as input (100 dimensions) in batches of size 32. The network is detailed in Figure 3. The set of labels used for training and testing changes according to each experiment. However, it is worth reminding the reader that those labels have not been used in the embedding task, and consequently, they are not directly included in the embedding information. The neural network consists of 3 dense layers. The hidden layers count 100 neurons each and use *ReLU*[8] as activation function. The output layer uses a sigmoid[9] as activation function and has a number of neurons equal to the dimension of the vocabulary of labels, which is represented with one-hot encoding. We performed 10-fold cross-validation for training the neural network and we provide as final score the average of the accuracy[10] computed on every fold.

For the first two goals (Section 5.1 and 5.2), a further experiment requires a preliminary splitting of the dataset in 10 equal folds, in order to alternatively use 1 of them as test set and the remaining 9 as training set, implementing a complete 10-fold cross-validation (CCV). The embeddings are generated using exclusively the training set, while the vectors representing the MIDI files in the test set are computed *a posteriori* as the mean of the embeddings of their sibling elements in the graph (tempos, programs, note groups, time signatures). Even if this approach is not commonly applied and not equivalent to the result of an embedding learning, we include this simplification with the purpose of demonstrating how graph embedding information is generalisable to unseen data. In the context of this experiment, the reported accuracy refers to the predictions on the test set generated by the neural network trained on the training set.

Currently, the library deals with the unpitched notes in MIDI Channel 10 (reserved to percussion by specification) as they have a pitch. When we ignore Channel 10 and use only programs representing pitched in-

[8] A rectified linear unit (ReLU) is a classic activation function in Deep Learning networks, which return 0 when the input is negative or the input value itself when it is positive. ReLU is widely used because of its simplicity and its empirically demonstrated fast convergence.

[9] The sigmoid function transforms the input x according to the formula:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

It exists between 0 and 1, so it is widely used when probabilities are requested in output. Its step curve shape gives it a behaviour similar to the Heaviside step, but derivable for any input.

[10] The definition of *accuracy* is "the closeness of agreement between a test result and the accepted reference value", i.e. the true value (ISO 5725-1).

struments[11], we empirically observe that the average scores remains substantially unchanged (less than 1% of difference), while the standard deviation is around 2% higher. For this reason, we report here only the performances obtained considering note events from all channels.

These experiments are available as notebooks at https://git.io/midi-embs. Furthermore, all models (embeddings) learned for each dataset are also published for supporting research reproducibility at http://www.doremus.org/midi/.

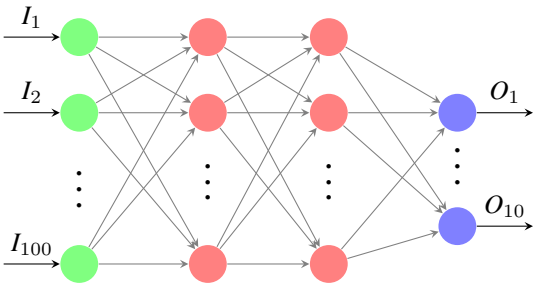


Fig. 3. Scheme of the neural network.

5.1. Genre Prediction

In [21], McKay et al. perform a genre classification task on a contextually published SLAC Dataset[12], which contains 250 MIDI files classified according to a two levels taxonomy. The first level includes 5 genre labels (Blues, Classical, Jazz, Rap, Rock), while the second one further specialises each genre by 2 sub-genres, for a total of 10 sub-genre labels. The dataset is perfectly balanced among the classes. Figure 4 shows a breakdown of the notes, instruments, tempo and time signature found in MIDIs of the SLAC dataset.

We perform a 5-class genre classification experiment as well as a 10-class experiment on the same dataset. In [21], the authors use different inputs for predicting the genre: symbolic music (S) –which is the MIDI content–, lyrics (L), audio (A), cultural features (C) (tags extracted from the Web) and the multi-modal combination of all of these features (SLAC). In particular, the symbolic information is organised around 111 features (1021 dimensions). Their work has been

[11] Musical instruments can be classified as *pitched*, producing recognisable notes in the musical scale (e.g. the piano), or *unpitched*, producing sounds of indefinite pitch (e.g. the cymbals).

[12] SLAC dataset: http://jmir.sourceforge.net/Codaich.html

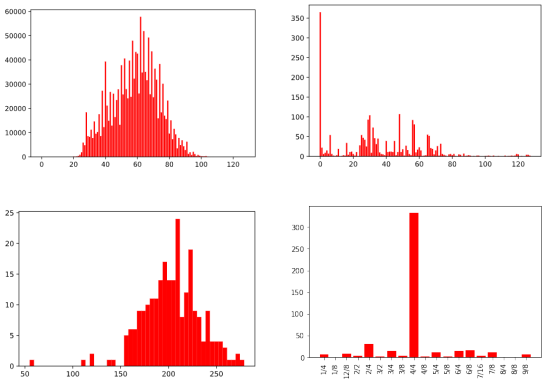


Fig. 4. From left to right and top to bottom, notes, instruments, tempo and time signature of the MIDIs in the SLAC dataset (249 files). The average note is B3 (we have omitted all drum events in channel 10); the most frequent instruments (peaks) are acoustic grand piano, string ensemble 1, and distortion guitar. The average tempo is 203.16 bpm, estimated with [42]. The most common time signature is 4/4.

extended and improved in a more recent paper [43] including, among others, features about chords and simultaneous notes, for a total of 172 features (1497 dimensions). We will compare our approach with these works, taking into account these 5 variants of features being used. The results are reported in Table 1.

Approach		**5 classes**	**10 classes**
McKay et al. 2010 [21]	S	85%	66%
	L	69%	43%
	A	84%	68%
	C	100%	86%
	SLAC	99%	85%
McKay et al. 2018 [43]		93.2%	77.6%
MIDI2vec + NN	ALL	**86.4%** (5.4%)	**67.2%** (7.8%)
	*N	81.6% (7.6%)	62.4% (9.9%)
	*P	79.6% (6.8%)	61.6% (8.6%)
	*T	27.2% (9.5%)	18.8% (9.2%)
	*TS	25.6% (9.2%)	15.2% (4.4%)
	*300	79.2% (7.0%)	57.2% (12.5%)
	CCV	76.8% (9.4%)	55.2% (6.5%)

Table 1

Accuracy of the genre classification. The reported values are the average (and standard deviation) of the cross-fold validation. In *N, *P, *T, *TS the embeddings have been computed on the sole notes, programs, tempos and time signature, while ALL includes all of them and *300 uses only the first 300 note-groups. Under CCV, the results of the complete cross-fold validation.

Our approach slightly outperforms [21] when only symbolic data are used in input (S), with an accuracy of 86% for 5-classes and 67% for 10-classes prediction.

In addition, our method outperforms also other variants, namely lyrics (in both classes) and audio (in the 5-classes). The improvements made in [43] increase these scores of a few percentage points. We believe that the combination of melodic and chords features was crucial in this case and worth to investigate in future work.

The same Table 1 shows also the accuracy scores obtained with different variations of the complete model (ALL); these variations compute the embeddings on the sole notes nodes (*N), program nodes (*P), tempo nodes (*T), and time signature nodes (*TS). None of these single features reach the accuracy score of their combination. It is not surprising that *N reaches the higher accuracy among those models, having been computed on a more populated graph – the number of *NoteOn*/*NoteOff* events is higher than any other kind of event. Moreover, this study proves the absence of correlation between mono-dimensional features (e.g. tempo) and the classes. Finally, we trained the embeddings on all features, but taking into account only the first 300 note groups (*300). The experiment shows that reducing the number of vertex in the graph causes lower accuracy scores.

Given the close results between the two best variations (ALL and *N), we studied their statistical significance, in order to understand if these two variations are likely to have the same accuracy mean. In order to do so, we extracted a t statistic computed on a 10x10-fold cross-fold validation, according to [44]. Applying this statistic to a Student's t-test, we obtain their p-values, comparing them with the common significance level $\alpha = 0.05$. For the 5 classes classification, the p-value of 0.048 suggests its statistical significance, while this is not confirmed with $p = 0.079$ for the 10 classes case. However, the repeated experiments show always better results for ALL when looking at the average of each 10-fold shuffle, while may happen that *N has punctual higher scores on single folds.

In the complete cross-fold validation (CCV) experiment, the accuracy scores are around 10% lower. This decrease is mostly due to the difference in computing the vectors of the nodes in the train set (embedding algorithm) and the test set (average of other nodes). However, the results are consistent respect to ALL, suggesting that the system is learning relevant features rather than coincidentally building a smart hashing on the content.

Figure 6 shows the confusion matrix between the real and the predicted values (configuration ALL). Even if there are no strong patterns, we can state that *Blues* is the genre that attract more negative predictions. This is confirmed by what we see in Figure 5, which contains a 2D visualisation of the vector space realised using the *t-Distributed Stochastic Neighbor Embedding* (t-SNE) algorithm[13] [45]. The final result is obtained by minimising the differences in this probability when computed on the low-dimensional space with respect to the high-dimensional one. In this figure, items of the same genre look closer in the space, with the *Blues* tracks occupying the central part of the graph, partially overlapping with the area of other genres. Figure 6b confirms that sub-genres belonging to the same parent genre are easier to be confused.

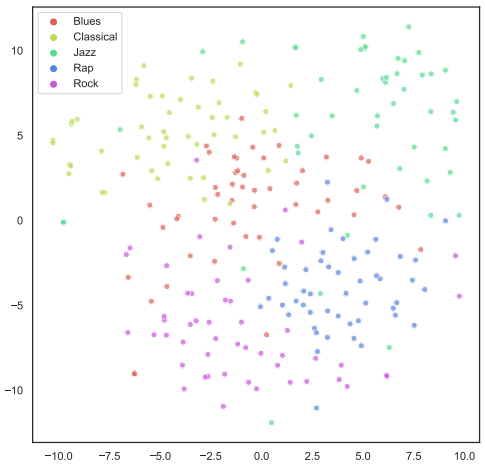


Fig. 5. 2D representation of the embedding space learned by midi2vec from the SLAC dataset.

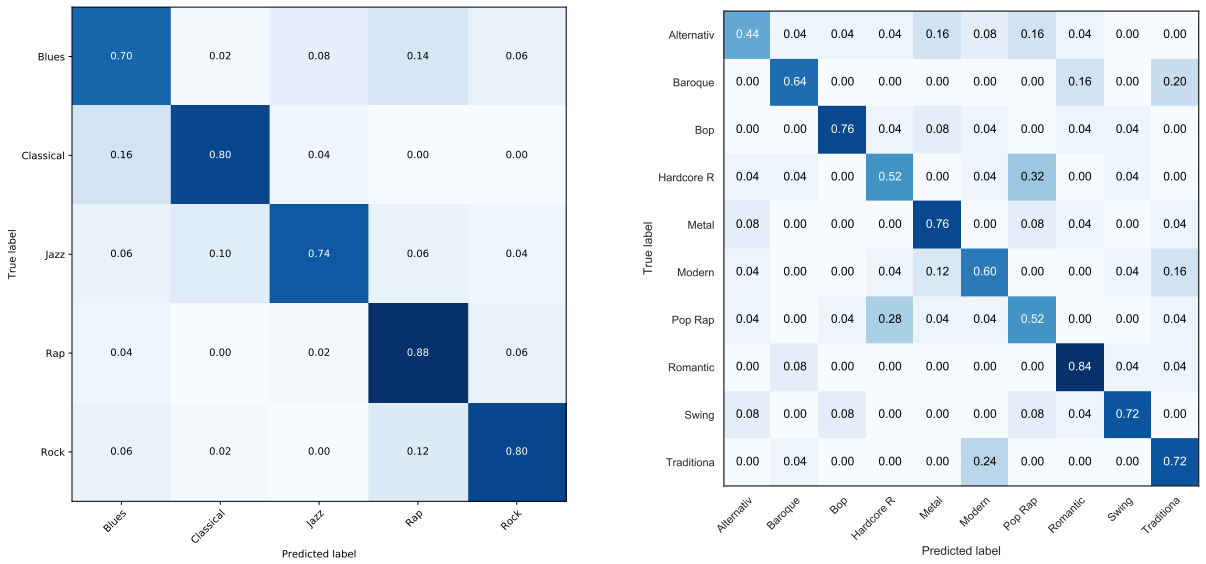
5.2. Metadata Prediction

This task consists in predicting a set of metadata from the MIDI, namely the *composer*, the *genre*, the *instrument* and the *movement*.

We started by downloading a corpus of 438 MIDI files from MuseData[14]. Those files refer to 139 classi-

[13] Similarly to Principal Components Analysis (PCA), t-SNE maps a high-dimensional space into a low-dimensional one. The algorithm computes the probability that a point A would choose point B as its neighbour, according to a Gaussian probability distribution centred at A.

[14] The MuseData dataset is available on the old musedata website: http://old.musedata.org



(a) Genre prediction

(b) Sub-genre prediction

Fig. 6. Confusion matrices of midi2vec predictions for the SLAC dataset.

cal music compositions, and each file can represent a specific movement. Figure 8 shows a breakdown of the notes, instruments, tempo and time signature found in MIDIs of the MuseData dataset.

MuseData provides also some metadata, like the composer name, the scholar catalogue number, a label for the movement. In order to obtain further information (i.e. the genre and the played instruments), we have interlinked each composition against the DOREMUS knowledge base [46], a dataset specialised in classical music metadata.

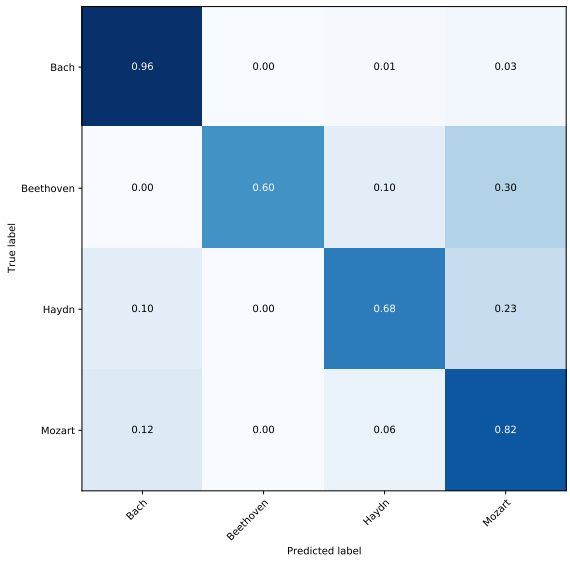
The interlinking process consists of three successive steps:

- interlinking of the composer through the exact match on the full name. This limits the candidates for the composition interlinking to the sole compositions of the interlinked composer;
- interlinking of the composition through the exact match on the catalogue number;
- if no catalogue number match is found, the titles are involved in the process. A title can often contain other kinds of information, such as key, instruments, opus number, etc. For example, the title “Symphony No. 3 in E-flat Major" contains the order number and the key. For this reason, titles are tokenised through empirical methods based on regular expressions to separate the different parts of the string, used as input of the Extended Jaccard Measure. [47]

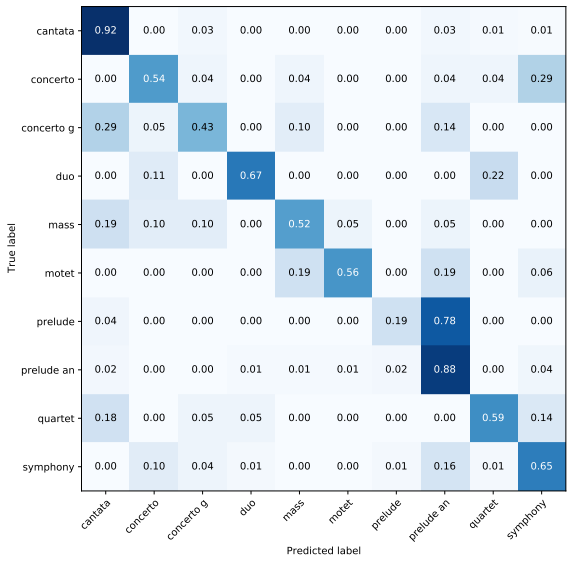
Every composition can be linked to more than one MIDI file, in the case of works made of multiple movements. The movement labels have been cleaned by removing the order number, the key, the instruments and eventual comments in parentheses. For example, *“1. Allegro in E Major"* becomes simply *“Allegro"*.

The interlinking gives access to precise metadata, mostly coming from controlled vocabularies [30], in particular composers (4 classes, i.e. Bach, Beethoven, Haydn, and Mozart), genres (10 classes), and instruments. For this last dimension, given the very large number of possibilities, we decided to reduce the number of classes to 6, including piano `P`, instrument (other than piano, including also small instrument ensembles) `I`, voice `V`, orchestra `O`, orchestra with voice `O+V`, and orchestra with instrumental soloist `O+S`. For instrument prediction, we excluded from the input, 21 MIDI with unknown instrumentation and 3 others which did not fall into any of the previous classes, having a final source dataset of 414 items.

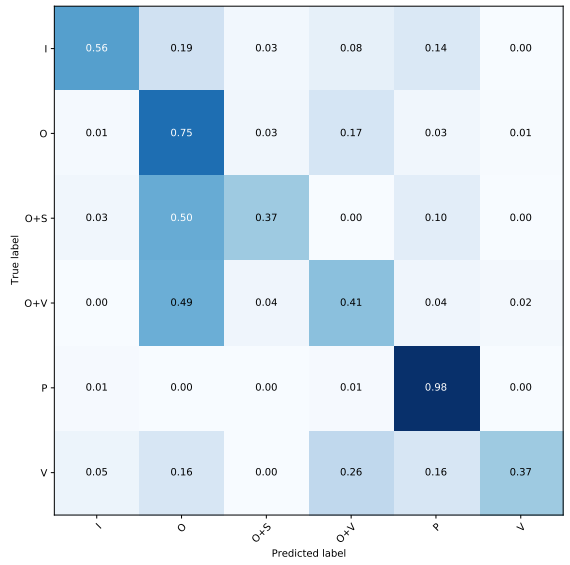
Furthermore, we consider also the movement label as a feature to predict, considering only those which were occurring more than 10 times. Those labels include tempos (*Allegro*) and musical forms (*Prelude*), for a total of 9 distinct classes on 335 MIDI files. Some of these categories are loosely defined, but we consider



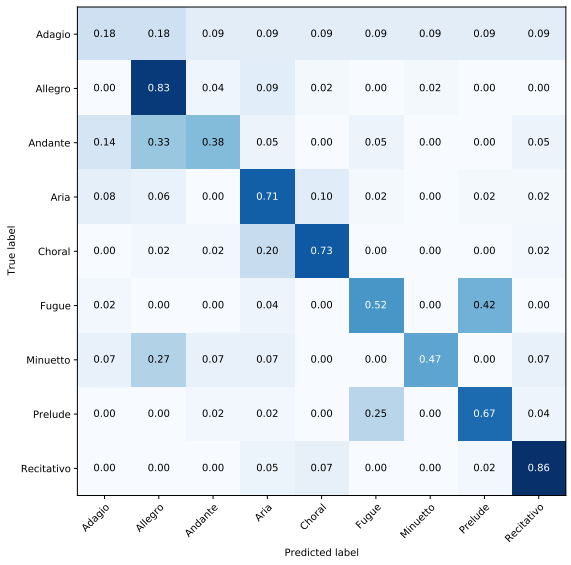
(a) Composer prediction



(b) Genre prediction



(c) Instrument prediction



(d) Movement label prediction

Fig. 7. Confusion matrices of midi2vec predictions for the MuseData dataset. For the instrument predictions (c), the labels are Instrument, Orchestra, Orchestra + Soloist instrument, Orchestra + Voice, Piano, Voice.

them as-is since ambiguity is part of music information and therefore also part of the task. The dataset is not balanced among classes and has a strong presence of Bach works (76% of the total).

The final accuracy (average of all the fold scores) is reported in Table 2. The best results are achieved for composer and genre prediction, and good results can be observed for all metadata. Looking at the confusion matrices:

– For the composers, the best results belong to Bach (the most present in the dataset). The two Austrian composers Mozart and Haydn are not surprisingly quite confused with one another, belonging both to the Classicism, differently

feature	n. items	n. classes	midi2vec	midi2vec CCV	jSymbolic
composer	438	4	90.4% (5.8%)	88.9% (15.0%)	78.7% (4.9%)
genre	438	10	71.3% (6.4%)	58.0% (16.6%)	37.9% (9.3%)
instrument	414	6	65.1% (17.5%)	48.6 (19.5%)	46.1% (8.6%)
movement	335	9	68.3% (12.7%)	54.9 (22.7%)	32.4 (6.6%)

Table 2

For each kind of metadata feature, the table reports the number of items, the number of distinct classes, the average (and standard deviation) accuracy score for midi2vec, midi2vec with complete cross validation, jSymbolic.

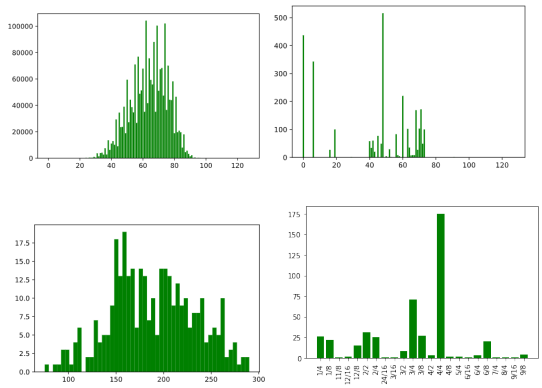


Fig. 8. From left to right and top to bottom, notes, instruments, tempo and time signature of the MIDIs in the MuseData dataset (439 files). The average note is E4 (we have omitted all drum events in channel 10); the most frequent instruments (peaks) are string ensemble 1, the acoustic grand piano, and the harpsichord. The average tempo is 188.96 bpm, estimated with [42]. The most common time signature is 4/4, with 3/4 also relatively frequent.

from Beethoven (Classic-Romantic) and Bach (Baroque) [48]. The score for Beethoven reflect its under-representation (only 10 tracks) in the dataset (Figure 7a);

- The genres are much more specific, with respect to the ones investigated in Section 5.1. As a consequence, the greatest confusion occurs between couples of very similar genres, such as [*concerto*, *symphony*] and [*prelude*, *prelude and fugue*] (Figure 7b). Those genre groups are overlapping also in the t-SNE visualisation in Figure 9;
- While the instrument prediction has great results in identifying works for orchestra, piano solo or small ensemble of instruments, it reveals some unreliable classification for voice-only pieces, probably due to the under-representation of the class in the dataset. In the same way, the approach is not able to distinguish compositions for orchestra, orchestra and voice, and orchestra and soloist, all classified under the class `O` (Figure 7c);
- Even if the movement labels include heterogeneous meaning, the network correctly predicts 7 over 10 items. Some confusion patterns can be spotted. The *Fugue* tag is often predicted as *Prelude*, on the other hand proving a correct genre prediction. The classes representing tempos (e.g. *Adagio* or *Tempo di Minuetto*) are often confused with the most represented class among them (*Allegro*). Some confusion is visible also between the two tags related to singing, *Aria* and *Choral* (Figure 7d).

Also in this case, we loose some accuracy (around 12-15%) when applying the complete cross-validation strategy. In order to have a comparison, we replicated the classification experiment described in [43], applying to Musedata a SVM classifier trained on the feature vectors computed by jSymbolic[15]. The results show how the features extracted from midi2vec are more capable to discern overlapping classes – e.g. genres of classical music, movement labels.

All those results should be analysed with a grain of salt, given the absence of balance between classes in the dataset.

5.3. Tag Prediction

The Lakh MIDI Dataset (LMD)[16] is one of the biggest collections of MIDI which have been realised for research purposes [9]. An `LMD-matched` subset contains 31,034[17] MIDI aligned to entries of the Million Song Dataset, providing a set of metadata about the tracks, the albums and the artists. Figure 10 shows a breakdown of the notes, instruments, tempo and time signature found in MIDIs of the Lakh dataset.

[15] We also used the jSymbolic vectors in combination with a Neural Network, but obtaining worse performance scores.

[16] Lakh MIDI Dataset: https://colinraffel.com/projects/lmd/

[17] The LMD website declares that `LMD-matched` includes 45,129 MIDIs. However, only 31,034 of them have metadata within a HDF5 file.

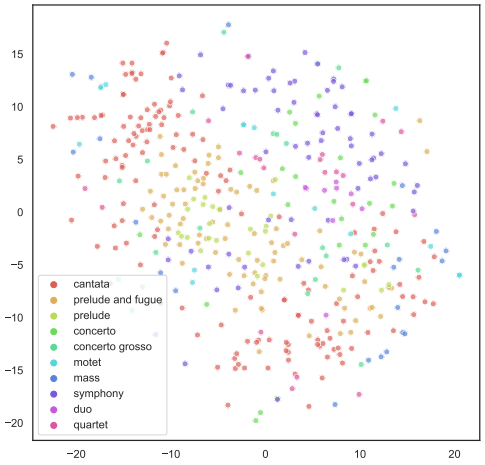


Fig. 9. 2D representation of the embedding space learned by midi2vec from the MuseData dataset.

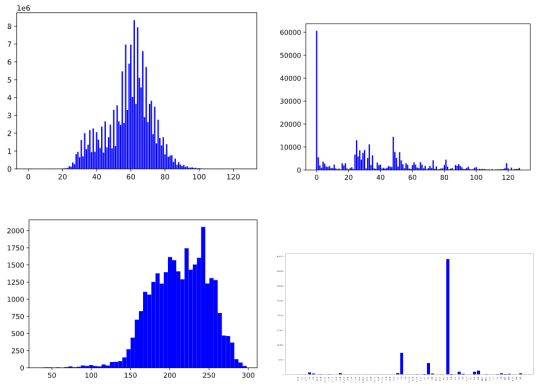


Fig. 10. From left to right and top to bottom, notes, instruments, tempo and time signature of the MIDIs in the Lakh dataset (31,036 files). The average note is B3 (we have omitted all drum events in channel 10); the most frequent instruments (peaks) are the acoustic grand piano, string ensemble 1, and acoustic guitar (steel). The average tempo is 213.15 bpm, estimated with [42]. The most common time signature is 4/4; 2/4 is also relatively frequent.

We extracted from `LMD-matched` two kinds of tags, coming respectively from MusicBrainz[18] and The Echo Nest[19]. The former group is a mix of terms which may refer to genres – i.e. *classic pop* – or to nationalities – *British* – while the latter group is more homogeneous in representing genres. Both kinds of tags refer more to the artist rather than to the exact track.

Differently from the experiment in Section 5.2, the size of the dataset allows to further filter the data in order to extract a balanced dataset. In particular, we select all distinct classes which are represented by at least 50 instances. For each of these classes, we randomly select 50 instances. Table 3 shows the accuracy for the predictions, measured through 10-fold cross-validation, together with the number of classes (distinct tags) against which we run the classification. In addition, in the table are reported the accuracy scores obtained by an SVM classifier build on top of jSymbolic. The comparison of this results reveals that latent features can largely boost the performance in tag classification, with evident benefit in real world scenarios like automatic tag prediction.

The confusion matrices are shown in Figure 11. Among the most wrongly predicted MusicBrainz classes, we find a strong presence of nationality tags (*British*, *Italian*, *UK*, etc.). The consistent number of classes do not let us detect patterns in the confusion matrix, in which the best values are however concentrated in the diagonal of corrected predictions. In order to overcome this issue, we report in Table 4 the most frequent wrong predictions. For MusicBrainz tags, the list includes loosely defined genres (*easy listening*, *ccm*), one evident error (line 6.), together with couples of tags which can be considered similar or overlapping (1., 3., 7.). Looking at EchoNest tags, all pairings are meaningful, involving similar or related genres. This data gives us more confidence that the neural network is learning music-relevant features, which are well represented through graph embeddings.

6. Conclusion and Future Work

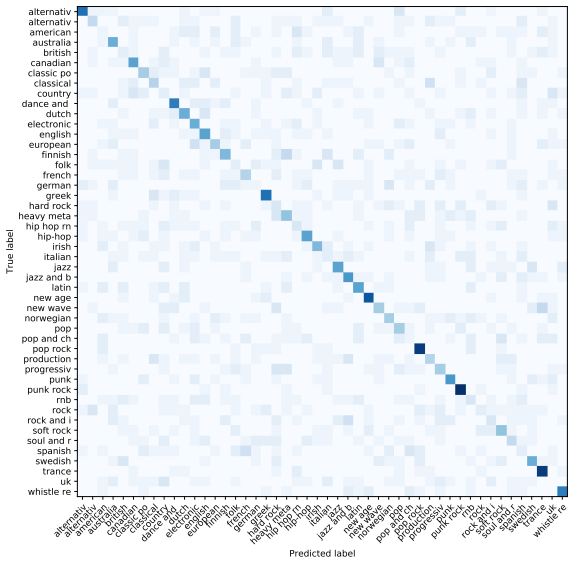
In this paper, we hypothesise that symbolic music content in MIDI files, and its embedding representation in vector space, are a powerful tool for automated metadata classification tasks. Traditionally, applications of machine learning to this problem have encountered limitations in feature selection, and more recent embedding-based techniques have been only used for other tasks (e.g. music generation) or on different data (e.g. music metadata). In this paper, we propose MIDI2vec, a method to represent MIDI content as a graph and, subsequently, in a vector space through learning graph embeddings. We gather evidence that our hypothesis holds: MIDI2vec embed-

[18] MusicBrainz: https://musicbrainz.org/
[19] The Echo Nest: http://the.echonest.com/

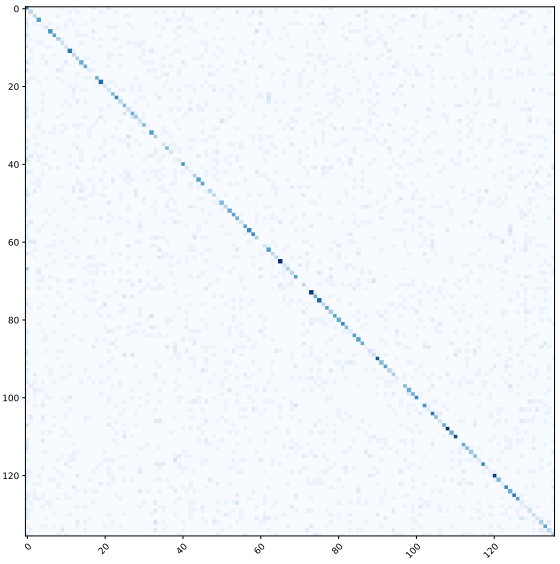
feature	n. items	n. classes	midi2vec	jSymbolic
MusicBrainz	2400	48	39.7% (2.8%)	3.7% (1.4%)
EchoNest	6800	136	32.5% (1.9%)	1.4% (0.4%)

Table 3

For each kind of tag feature, the table reports the number of items, the number of distinct classes, the average (and standard deviation) accuracy score for midi2vec and jSymbolic.



(a) MusicBrainz Tag prediction



(b) The Echo Nest Tag prediction

Fig. 11. Confusion matrices of midi2vec predictions for the Lakh dataset.

	tag	ground truth	predicted	%
MusicBrainz	1.	ballad	folk rock	12
	2.	blues-rock	ccm	12
	3.	classic rock	british invasion	12
	4.	cool jazz	ccm	10
	5.	easy listening	classic rock	12
	6.	flamenco	british invasion	12
	7.	folk rock	ballad	10
EchoNest	8.	orchestra	requiem	10
	9.	progressive trance	hard trance	10
	10.	ragtime	jazz	10
	11.	requiem	orchestra	10
	12.	techno	tech house	10

Table 4

Most frequent prediction errors for tag classification, with the percentage of cases over the total of the class

dings were successful in metadata classification, obtaining comparable performances to state-of-art methods based on feature extraction from symbolic music, with the added advantages of scalability, automating feature engineering, and reducing the required dimensions by one order of magnitude.

We plan on improving this work in various ways. Even if experiments revealed that the impact of unpitched notes in Channel 10 is minimal, we intend to assign a separated branch of the graph to percussion notes in the future, on order to distinguish them from the other ones while still taking them into account, in order to try to improve the overall performance.

Being transformed into a flat graph, the MIDI content loses in MIDI2vec all time-based information, with the only exception of simultaneity. Given the importance of melodic patterns in a music piece, future work would investigate how this work can deal with note sequences. We plan to investigate a few different strategies. First, sequences of notes may be encoded similarly to the simultaneous notes groups and included in MIDI2vec as a fifth node type. A second strategy may rely upon the inclusion in the graph embedding process of sequence embeddings like Sequence Graph Transform (SGT), which is capable to

short- and long- term sequence features [40]. It is possible to think of MIDI file as a *temporal graph*, in which the information (currently played notes, tempo, playing instruments, etc.) is evolving over time, and apply to such a graph temporal node embeddings strategies [41]. Finally, another study may combine MIDI2vec with other feature extraction techniques – e.g. the previously cited [43] – in an ensemble system, in order to exploit the best of the two methods.

A MIDI ontology and a corpus of over 300 thousand MIDI in RDF format have been presented in [49]. Despite being an interesting target for MIDI2vec, the extraction of crucial information (like the duration of a note) from the dataset is hard. In the current version, the ontology faithfully reproduces the event structure of the MIDI files, while significant edges – e.g. among simultaneous notes or consecutive events – are missing. Still, the MIDI2vec approach does not exploit edges between consecutive groups of notes, while they may potentially impact on the performances. We plan to extend or map the MIDI ontology in order to solve this issue and enable MIDI2vec for working on such corpus, e.g. to perform link discovery and knowledge graph completion. Moreover, it would be interesting to extend this approach to other symbolic music notation formats, namely MusicXML.

According to some intuition from other works in the genre classification field [12], the computation should not necessarily involve the full length of the track. Experiments with different time spans or sample sizes among the graph edges can help in detecting a trade-off between the performances and the embedding computation time. Recent approaches for including literal values in graph embeddings [50, 51] could be included in MIDI2vec, in order to avoid any arbitrary choice that value-partitioning implies. Finally, we will use MIDI2vec in more applied contexts, such as the task of knowledge graph completion in knowledge bases with incomplete metadata entries [3].

References

[1] M.A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes and M. Slaney, Content-Based Music Information Retrieval: Current Directions and Future Challenges, *Proceedings of the IEEE* **96**(4) (2008), 668–696. doi:10.1109/JPROC.2008.916370.

[2] D. Bogdanov, N. Wack, E. Gómez Gutiérrez, S. Gulati, P. Herrera Boyer, O. Mayor, G. Roma Trepat, J. Salamon, J.R. Zapata González and X. Serra, Essentia: An audio analysis library for music information retrieval, in: *14th International Society for Music Information Retrieval Conference (ISMIR)*, Curitiba, Brazil, 2013.

[3] A. Meroño-Peñuela, R. Hoekstra, A. Gangemi, P. Bloem, R. de Valk, B. Stringer, B. Janssen, V. de Boer, A. Allik, S. Schlobach et al., The MIDI Linked Data Cloud, in: *16th International Semantic Web Conference (ISWC)*, Springer, Vienna, Austria, 2017, pp. 156–164.

[4] A. Ratner, C.D. Sa, S. Wu, D. Selsam and C. Ré, Data Programming: Creating Large Training Sets, Quickly, 2017.

[5] A. Ratner, S.H. Bach, H. Ehrenberg, J. Fries, S. Wu and C. Ré, Snorkel: Rapid training data creation with weak supervision, in: *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 11, NIH Public Access, 2017, p. 269.

[6] MIDI Manufacturers Association, The Complete MIDI 1.0 Detailed Specification, Technical Report, MIDI Manufacturers Association, Los Angeles, CA, USA, 1996-2014, https://www.midi.org/specifications/item/the-midi-1-0-specification.

[7] A. Roberts, J. Engel, C. Raffel, C. Hawthorne and D. Eck, A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music, in: *35 th International Conference on Machine Learning (ICML)*, Stockholmsmässan, Sweden, 2018.

[8] C. Raffel and D.P.W. Ellis, Extracting Ground Truth Information from MIDI Files: A MIDIfesto, in: *ISMIR 2016*, 2016.

[9] C. Raffel, Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching, PhD thesis, Columbia University, 2016.

[10] O. Celma and X. Serra, FOAFing the Music: Bridging the Semantic Gap in Music Recommendation, *Web Semantics: Science, Services and Agents on the World Wide Web* **6**(4) (2008), 250–256. doi:10.1016/j.websem.2008.09.004.

[11] C. McKay and I. Fujinaga, Automatic Genre Classification Using Large High-Level Musical Feature Sets, in: *5th International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, 2004. http://ismir2004.ismir.net/proceedings/p095-page-525-paper240.pdf.

[12] Z. Cataltepe, Y. Yaslan and A. Sonmez, Music Genre Classification Using MIDI and Audio Features, *EURASIP Journal on Advances in Signal Processing* **2007**(1) (2007), 036409. doi:10.1155/2007/36409.

[13] Z. Fu, G. Lu, K.M. Ting and D. Zhang, A Survey of Audio-Based Music Classification and Annotation, *IEEE Transactions on Multimedia* **13**(2) (2011), 303–319. doi:10.1109/TMM.2010.2098858.

[14] I. Guyon and A. Elisseeff, An Introduction to Variable and Feature Selection, *Journal of Machine Learning Research* **3**(Mar) (2003), 1157–1182.

[15] T. Mikolov, K. Chen, G. Corrado and D. Jeffrey, Efficient Estimation of Word Representations in Vector Space, in: *1st International Conference on Learning Representations, (ICLR), Workshop Track*, Scottsdale, AZ, USA, May 2-4, 2013, 2013. http://arxiv.org/abs/1301.3781.

[16] A. Grover and J. Leskovec, node2vec: Scalable Feature Learning for Networks, in: *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, San Francisco, CA, USA, 2016. doi:10.1145/2939672.2939754.

[17] A. Huang and R. Wu, Deep Learning for Music, *Computing Research Repository (CoRR)* **abs/1606.04930** (2016).

[18] Y. Yan, E. Lustig, J. VanderStel and Z. Duan, Part-invariant Model for Music Generation and Harmonization, in: *19th Inter-*

national Society for Music Information Retrieval Conference (ISMIR), Paris, France, 2018.

[19] P. Ristoski, J. Rosati, T. Di Noia, R. De Leone and H. Paulheim, RDF2Vec: RDF Graph Embeddings and Their Applications, *Semantic Web Journal* **10**(4) (2019), 721–752. doi:10.3233/SW-180317.

[20] X. Wilcke, P. Bloem and V. De Boer, The knowledge graph as the default data model for learning on heterogeneous knowledge, *Data Science* **1**(1–2) (2017), 39–57.

[21] C. McKay, J. Burgoyne, J. Hockman, J.B.L. Smith, G. Vigliensoni and I. Fujinaga, Evaluating the Genre Classification Performance of Lyrical Features Relative to Audio, Symbolic and Cultural Features, in: *11th International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, 2010.

[22] L. Weng, Are Deep Neural Networks Dramatically Overfitted?, *lilianweng.github.io/lil-log* (2019). http://lilianweng.github.io/lil-log/2019/03/14/are-deep-neural-networks-dramatically-overfitted.html.

[23] Z.S. Harris, Distributional Structure, *WORD* **10**(2–3) (1954), 146–162.

[24] R. Abboud and J. Tekli, MUSE Prototype for Music Sentiment Expression, in: *IEEE International Conference on Cognitive Computing (ICCC)*, San Francisco, CA, USA, 2018, pp. 106–109. doi:10.1109/ICCC.2018.00023.

[25] D.C. Corrêa and F.A. Rodrigues, A survey on symbolic data-based music genre classification, *Expert Systems with Applications* **60** (2016), 190–210. doi:10.1016/j.eswa.2016.04.008. http://www.sciencedirect.com/science/article/pii/S095741741630166X.

[26] F. Korzeniowski and G. Widmer, Genre-Agnostic Key Classification With Convolutional Neural Networks, in: *19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018. http://ismir2018.ircam.fr/doc/pdfs/7_Paper.pdf.

[27] J.S. Gomez, J. Abeßer and E. Cano, Jazz solo instrument classification with convolutional neural networks, source separation, and transfer learning, in: *19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[28] F. Colombo, J. Brea and W. Gerstner, Learning to Generate Music with BachProp, in: *16th Sound and Music Computing Conference (SMC)*, Malaga, Spain, 2019, pp. 380–386.

[29] P. Lisena and R. Troncy, Combining Music Specific Embeddings for Computing Artist Similarity, Suzhou, China, 2017. http://www.eurecom.fr/publication/5361.

[30] P. Lisena, K. Todorov, C. Cecconi, F. Leresche, I. Canno, F. Puyrenier, M. Voisin and R. Troncy, Controlled Vocabularies for Music Metadata, in: *19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018. http://ismir2018.ircam.fr/doc/pdfs/68_Paper.pdf.

[31] R. van der Weerdt, Generating Music from Text: Mapping Embeddings to a VAE's Latent Space, Master's thesis, University of Amsterdam, 2018.

[32] J. Pennington, R. Socher and C.D. Manning, GloVe: Global Vectors for Word Representation, in: *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. doi:10.3115/v1/D14-1162.

[33] R. Diestel, *Graph Theory*, Graduate Texts in Mathematics, Vol. 173, Springer, 2005. doi:10.1007/978-3-662-53622-3.

[34] P. Goyal and E. Ferrara, Graph embedding techniques, applications, and performance: A survey, *Knowledge-Based Systems* **151** (2018), 78–94. doi:10.1016/j.knosys.2018.03.022. http://www.sciencedirect.com/science/article/pii/S0950705118301540.

[35] B. Perozzi, R. Al-Rfou and S. Skiena, DeepWalk: Online Learning of Social Representations, in: *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, New York, NY, USA, 2014, pp. 701–710. ISBN 978-1-4503-2956-9. doi:10.1145/2623330.2623732.

[36] E. Palumbo, D.M.G. Rizzo, R. Troncy and E. Baralis, entity2rec: Property-specific Knowledge Graph Embeddings for Item Recommendation, *Expert Systems With Applications* (2020).

[37] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu and S. Jaiswal, graph2vec: Learning Distributed Representations of Graphs, in: *13th International Workshop on Mining and Learning with Graphs (MLG)*, 2017.

[38] M. Kejriwal and P. Szekely, Neural Embeddings for Populated Geonames Locations, in: *16th International Semantic Web Conference (ISWC)*, Springer International Publishing, Vienna, Austria, 2017, pp. 139–146. ISBN 978-3-319-68204-4. doi:10.1007/978-3-319-68204-4_14.

[39] J. Pujara, E. Augustine and L. Getoor, Sparsity and Noise: Where Knowledge Graph Embeddings Fall Short, in: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2017, pp. 1751–1756. doi:10.18653/v1/D17-1184. http://aclweb.org/anthology/D17-1184.

[40] C. Ranjan, S. Ebrahimi and K. Paynabar, Sequence Graph Transform (SGT): A Feature Extraction Function for Sequence Data Mining, *arXiv preprint arXiv:1608.03533* (2016).

[41] U. Singer, I. Guy and K. Radinsky, Node Embedding over Temporal Graphs, in: *28th International Joint Conference on Artificial Intelligence, (IJCAI)*, IJCAI Organization, 2019, pp. 4605–4612.

[42] C. Raffel and D.P. Ellis, Intuitive Analysis, Creation and Manipulation of MIDI Data with `pretty_midi`, in: *15th International Conference on Music Information Retrieval (ISMIR), Late Breaking and Demo Papers*, 2014, pp. 84–93.

[43] C. McKay, J.E. Cumming and I. Fujinaga, jSymbolic 2.2: Extracting Features from Symbolic Music for use in Musicological and MIR Research, in: 19*th International Conference on Music Information Retrieval (ISMIR)*, Paris, France, 2018. http://ismir2018.ircam.fr/doc/pdfs/26_Paper.pdf.

[44] R.R. Bouckaert and E. Frank, Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms, in: *Advances in Knowledge Discovery and Data Mining*, H. Dai, R. Srikant and C. Zhang, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 3–12. ISBN 978-3-540-24775-3.

[45] L. van der Maaten and G. Hinton, Visualizing data using t-SNE, *Journal of machine learning research* **9**(Nov) (2008), 2579–2605.

[46] M. Achichi, P. Lisena, K. Todorov, R. Troncy and J. Delahousse, DOREMUS: A Graph of Linked Musical Works, in: *17th International Semantic Web Conference (ISWC)*, Monterey, CA, USA, 2018. http://www.eurecom.fr/publication/5565.

[47] A.N. Tigrine, Z. Bellahsene and K. Todorov, Light-Weight Cross-Lingual Ontology Matching with LYAM++, in: *On the Move to Meaningful Internet Systems Conferences (OTM)*, Springer International Publishing, Rhodes, Greece, 2015, pp. 527–544. ISBN 978-3-319-26148-5. doi:10.1007/978-3-319-26148-5_36.

[48] C. Rosen, *The Classical Style: Haydn, Mozart, Beethoven*, WW Norton & Company, 1997. ISBN 0393317129.

[49] A. Meroño-Peñuela, M. Daquino and E. Daga, A Large-Scale Semantic Library of MIDI Linked Data, in: *5th International Conference on Digital Libraries for Musicology (DLfM)*, Paris, France, 2018.

[50] M. Cochez, M. Garofalo, J. Lenßen and M.A. Pellegrino, A First Experiment on Including Text Literals in KGloVe, in: *4th Workshop on Semantic Deep Learning (SemDeep)*, Monterey, CA, USA, 2018.

[51] A. Kristiadi, M. Asif Khan, D. Lukovnikov, J. Lehmann and A. Fischer, Incorporating Literals into Knowledge Graph Embeddings, *Computing Research Repository (CoRR)* **abs/1802.00934** (2018). http://arxiv.org/abs/1802.00934.