# Music-Graph2Vec: An Efficient and Effective Method for Embedding Pitch Segment

No Author Given

No Institute Given

**Abstract.** Learning low-dimensional continuous vector representation for short pitch segment extracted from songs is key to melody modeling that can be utilized in many music investigations, such as genre classification, emotion classification, and music retrieval, and so on. The skip-gram version of Word2Vec is ubiquitous, and widely used approach for music pitch segment embedding, but it poorly scales to large data sets due to its extremely long training time. In this paper, we propose a novel efficient graph-based embedding method, named Music-Graph2Vec, to tackle this concern. This approach converts music files into graphs, extracts the rhythmic sequence through random walking, and trains the rhythmic embedding model using skip-gram. Experimental results demonstrate that Music-Graph2Vec outperforms Word2Vec in training rhythmic embedding, with the advantage of being 55 times faster on the top-MAGD dataset, with the same accuracy for Word2Vec in terms of music genre classification.

**Keywords:** Genre Classification · Music Graph · MIDI · Random Walk

Music analyzing, including tasks such as emotion classification [7], genre classification [21], music retrieval [23] and music recommendation system [25] and so on, has attracted lots of attention in both academia and industry. Recently, since music embedding techniques [9, 5, 3, 24], especially for the phenomenal progress on the deep learning technique, can capture meaningful pitch and harmonic relationships in music, these embedding techniques are widely carried out in abovementioned music applications. More precisely, music embedding technique [5], in spired of Word2Vec, first divides the long music sequence into the pitch information of symbolic music slices, or "pitch segments", applies the skip-gram [19] to gain the embedding for these pitch segments, and then aggregates the slices embedding together to present a music.

However, these music embedding techniques scale poorly to large-scale data sets because they suffer huge time from training the pitch segment embedding. For example, training music pitch segment embeddings using Word2Vec on the top-MAGD dataset [26], which only contains 1.41GB of MIDI data, took up to **2133.7** seconds. To tackle this concern, we propose a novel graph-based approach, Music-Graph2Vec, which only needs **38.9** seconds (**55** times faster than Word2Vec) with the same accuracy on the same data set.

Our idea is to provide a sufficiently abstract representation of a melodic line that actually makes sense from a musical point of view. Melodies are generally studied from a pitch sequence/contour point of view. We take as a starting point the interval structure, by which we mean the network of connections between pitches. We remark that melodies use only a subset of all possible connections, and with different frequencies. The graph is used to model such relationships. A node in a graph is a pitch segment, and an edge between two nodes describes the co-occurrence between two pitch segments. Therefore, based on the music graph, we propose Music-Graph2Vec, which converts music from sequences into graphs and samples from the graphs to train music embeddings. This approach ensures efficiency and effective when processing large-scale music corpus.

The Music-Graph2Vec approach consists of three steps. First, we construct a graph of music by using co-occurrence information between different pitches. Second, we perform random walks on the constructed music graph and sample music pitch sequences. Third, we apply Skip-Gram[19] to process the sampled music sequences and obtain the pitch segment embeddings. Skip-Gram utilizes contextual information from neighboring pitch segments in the sequences to learn meaningful embeddings that capture the underlying semantic relationships between pitch segments.

Our method significantly reduces the time complexity of training pitch segment embeddings. The reason can explained that there are only 4096 nodes in the music graph, because of only 12 types of music pitch nodes (C, C#, D, D#, E, F, F#, G, G#, A, A#, B) with $2^{12} = 4096$. Meanwhile, with the music corpus increasing, these newly added music melodies only change the edge weights of the pitch segment co-occurrence graph, but the size of nodes do not have any alteration. Thus, it does not affect the Music-Graph2Vec's performance.

**Contributions:**

- We offer a detailed study on the pitch segment co-occurrence graph with consideration of different edge weights and nodes weights to describe a music.
- We present the Music-Graph2Vec approach, which is based on the pitch segment co-occurrence graphto calculate the pitch segment embedding. An effective sampling strategy is developed to generate the music slice sequences, and the Skip-Gram algorithm is used to calculate the pitch segment embedding on these music sequences.
- Extensive experiments are conducted to evaluate the efficiency and the accuracy of the proposed algorithm. Experimental results show that the Music-Graph2Vec algorithm is capable of outperforming Word2Vec by 55 times on 1.41GB of MIDI data with the same accuracy in genre classification task.

**RoadMap:** The rest of the paper is organized as follows: In section 1, we describe previous work. In section 2 we introduce the overview of music embedding. Section 3 presents our Music-Graph2Vec method. In section 4, we present experimental results on three datasets and comparison with our baseline method. Summary and conclusion are made in section 5.

## 1 Related Work

### 1.1 Symbolic Music Encoding

There are two primary ways to represent music: as audio and as symbolic music. In the case of symbolic music, two main approaches are commonly used for encoding: piano-roll-based and MIDI-based. Piano-roll-based methods [13, 4] typically involve encoding music as a 2-dimensional binary matrix, where one dimension corresponds to pitches and the other corresponds to time steps. Each element in the matrix indicates whether the pitch is played at a given time step. However, this method has a limitation in that it divides notes into fixed intervals, which can be inefficient, particularly for longer notes. MIDI, or Musical Instrument Digital Interface, is a technical standard used for transferring digital instrument data. In the field of symbolic music, many research works [20, 12] encode music pieces based on MIDI events. It utilizes a symbolic representation that is transmitted serially and indicates note-on and note-off events, allowing for a high temporal sampling rate. The loudness of each note is encoded using a discrete quantity known as velocity, which is named after the speed at which a piano key is pressed. While MIDI encodes note timing and duration, it does not account for qualities such as timbre. Instead, MIDI events are used to trigger the playback of audio samples. MIDI-based methods offer greater flexibility and versatility when encoding symbolic music compared to piano-roll-based methods. Therefore, this paper focuses on MIDI-based symbolic music analysis.

### 1.2 Symbolic Music Embeddings Training

Inspired by Word2Vec [19] in natural language processing (NLP), previous studies on symbolic music understanding have learned music embeddings by predicting a music symbol based on its neighboring symbols. Huang and Liang [11] and Madjiheurem and Renault [16] viewed chords as words in NLP and used the word2vec model to learn chord representations. Chuan et al. [5] divided music pieces into non-overlapping slices with a fixed duration and trained embeddings for each slice. Hirai et al. [10] clustered musical notes into groups, treating these groups as words for representation learning. However, the Word2Vec-based approaches mentioned above only use relatively simple methods to train music embeddings, which have limited performance and require more time to train. Inspired by the graph theory, we remark that music melodies use only a subset of all possible connections, and with different frequencies, and build the pitch segment co-occurrence graph, in which a node is a pitch segment, and an edge between two nodes describes the co-occurrence between two pitch segments. Based on the pitch segment co-occurrence graph, we propose Music-Graph2Vec, which samples from the pitch segment co-occurrence graph to train music embeddings. This approach ensures efficiency and effective when processing large-scale music corpus.

## 2 Background

Both music and language consist of sequential events that follow a set of rules or grammar, albeit with more flexibility in music. Moreover, both domains elicit expectations about upcoming events, such as words or chords, and unexpected deviations trigger cognitive responses, such as the N400 or Early Right Anterior Negativity (ERAN) event-related potentials (ERPs) [2, 14]. In NLP area, grammar rules impose constraints on word order, while in music, rules of style and music theory influence the arrangement of musical elements, such as tones or chords. However, unlike NLP area, music allows for simultaneously expressing multiple elements. This unique characteristic means that the fundamental units of music are music slices comprising one or more tones and lasting for a duration determined by time signature and music density. Therefore, when applying embedding techniques, it is crucial to carefully consider these features and employ appropriate technical approaches to address challenges such as computational time and memory consumption while ensuring the accuracy and effectiveness of the analysis.

According to the approach proposed by Chinghua Chuan et al. [5], they slice each music into equally sized slices, where each slice is represented by a list of pitch names contained within that slice, and each pitch name only count once in each slice. Specifically, the music is divided into non-overlapping equal-length slices; each slice is spanned one beat (one pitch segment). Noted that the duration of one beat may vary across different musical compositions. Thus the slice lengths are estimated based on the beat duration calculated using the MIDIToolBox [27]. Within these slices, all the pitches, including chord tones, non-chord tones, and embellishments, are recorded. Additionally, the songs are not transposed to the same key, and each pitch within a slice is represented by its corresponding pitch name. For example, both C4 and C5 are encoded as "C". After extracting the pitch names in each beat, we encode them to 12-bit binary numbers as a vector for each beat. The 12 bits in the binary number represent whether the corresponding pitch names (C, C#, D, ..., B) are present in the current beat, with each bit indicating the presence or absence of a specific pitch name from low to high.

Fig. 1 presents an example of segmenting a 7-bar excerpt from the Italian folk song "Bella Ciao". We have listed the pitch names that appear in each beat and the corresponding vector generated based on each beat. For instance, in the fourth beat of the second bar, the pitches E and F# are present. A vector of 000001010000, equivalent to 80 in decimal representation. In convenience, in the following sections, we will refer to the pitch information extracted from each beat as "pitch segments". In subsequent experiments, we will use the decimal representation of the vector corresponding to each pitch segment in multiple instances.

Chinghua Chuan et al. [5] approach employ the Skip-Gram method to process music vectors, specifically training the vector space with pitch segments so that each pitch segment can be represented as a vector in the space. In this vector space, pitch segments that occur in similar musical contexts tend to have similar

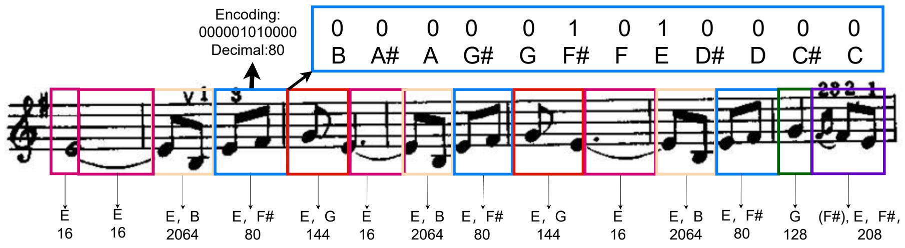

Fig. 1: This is an example of music segmentation. It is an excerpt from the Italian folk song "Bella Ciao", consisting of seven bars. Each box represents one beat, and the subscripts indicate the pitch names that appear in that beat along with their corresponding code. For example, in the second bar, second beat, there are two notes - E and F# - so its encoding is 000001010000, which represents the decimal value 80. The note on the right top of the figure shows that the encoding of pitch segment E and F# is 000001010000. In the last beat, note F# appears twice, but it is only considered as a single count. Therefore, the last pitch segment is (E, F#, G), encoding 000011010000(208).

meanings, and thus, pitch segments that appear in the same context are geometrically close to each other. The underlying principle of the Skip-Gram model is to acquire high-dimensional vectors that bring the vectors of pitch segments which co-occur in the same contextual environment closer together in the vector space through training. This is achieved by utilizing a sliding window approach, where each pitch segment is treated as the central pitch segment, while the surrounding pitch segments are considered target pitch segments. Subsequently, a neural network is employed to predict the target pitch segments that appear within a given musical context. During training, the Skip-Gram model learns two distinct matrices of pitch segment vectors: the central pitch segment vector matrix and the target pitch segment vector matrix. The central pitch segment vector matrix denotes the position of each pitch segment in the vector space. In contrast, the target pitch segment vector matrix is employed to compute the probability of predicting the target pitch segments. By employing iterative optimization of the loss function throughout the model training process, the Skip-Gram model endeavors to minimize the disparity between the predicted target pitch segments and the actual target pitch segments found in the training data. This allows the model to acquire comprehensive vector representations for each pitch segment.

The time complexity of generating pitch segment embeddings using the Skip-Gram model is $O(|N|\log(|V|)$, where $|N|$ represents the total number of pitch segment and $|V|$ represents the number of unique pitch segment types (e.g., the maximum value is $2^{12}$, 4096.) that occur in the corpus. This implies that as the corpus size increases, the time complexity of generating pitch segment embeddings using Skip-gram also grows with the trend of $|N|\log(|V|)$. Thus, processing a large amount of data using the Skip-Gram method needs substantial time. To address this issue, we propose the Music-Graph2Vec method, which offers significant advantages in reducing training time while maintaining comparable classification accuracy for the final trained pitch segment embedding model.

## 3 Music-Graph2Vec

Although the top-MAGD dataset [26] contains 18,061,246 pitch segments, there are only 4,096 types of pitch segments. Therefore, we convert the original pitch segment sequences in music into a pitch segment co-occurrence graph and then sampling from these graphs to train the pitch segment embedding model.

Fig. 2 illustrates the complete process of our Music-Graph2Vec approach. The Music-Graph2Vec approach consists of three steps. The first step involves constructing a pitch segment co-occurrence graph using the co-occurrence information between different pitch segments. The second step is to perform random travelling on the pitch segment graph and sample sequences of pitch segments. In the third step, the sampled sequences obtained from the pitch segment co-occurrence graph are used to train the embedding model using the Skip-Gram algorithm to generate pitch segment embeddings.

### 3.1 Pitch Segment Graph Construction

The first step involves constructing the pitch segment graph based on the co-occurrence information between different pitch segments. Fig. 3 shows an example of a pitch segment graph generated from 7 musical phrases of the Italian folk "Bella ciao". In the graph, the label in the nodes represent an encoded vector value (see 1). During constructing the pitch segment graph, each pitch segment is treated as a node, and edges represent the co-occurrence relationships between adjacent pitch segments. The direction of the edges indicates the order of occurrence of the pitch segments, and the edge weights represent the number of times adjacent pitch segments co-occur in the given order. If nodes $v$ and $x$ are adjacent, the weight $W_{v,x}$ of the edge between them is calculated as follows:

$$W_{v,x} = \sum co - occurrence(v, x) \tag{1}$$

The term $\sum co - occurrence(v, x)$ represents the total count of occurrences where pitch segment $v$ and pitch segment $x$ appear in sequence across all music files. For example, in Fig. 3, pitch segments (E, G) and (E) co-occur in sequence twice, during the fifth and the sixth beat, as well as during beats nine and ten. Therefore, the edge weight between node 144 (pitch segments (E, G)) and node 16 (pitch segments (G)) should be 2 (see red frame in Fig. 3).

Furthermore, each node has a weight representing the probability of occurrence for the corresponding pitch segment across all files. The weight $PW_v$ for node $v$ is calculated using the following formula:

$$PW_v = \frac{n_v}{\sum_k n_k} \tag{2}$$

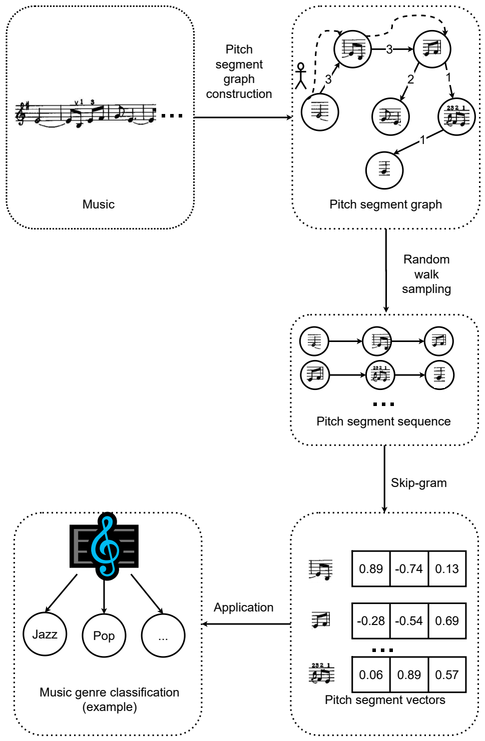

Fig. 2: The framework of Music-Graph2Vec. The first step is to construct the pitch segment graph based on the co-occurrence information between different pitch segments. The second step is to do random walks on the pitch segment graph and sample pitch segment sequences. The third step is to apply Skip-Gram on the sampled pitch segment sequences, resulting in pitch segment embeddings. Then, we can apply the segment embeddings in application (e.g., music genre classification).

Where $n_v$ represents the number of occurrences of pitch segment $v$ across all files, and $\sum_k n_k$ represents the total number of occurrences of all pitch segments. For example, in Fig. 3, the pitch segment (E, F#) appears three times in the 4th, 8th, and 12th beats (see blue frame in Fig. 2). Therefore, the weight of node 80 (the pitch segment (E, F#)), should be $\frac{3}{14} = 0.21$, as shown in Fig. 3.

### 3.2 Pitch Segment Sampling

The second step involves random travelling on the pre-constructed pitch segment graph and sampling pitch segment sequences. Fig. 4 presents an example where $total\_walks = 1$ and $walk\_length = 3$. We choose node 16 as the starting point. In one of the random walking processes, we start from node 16 and arrive

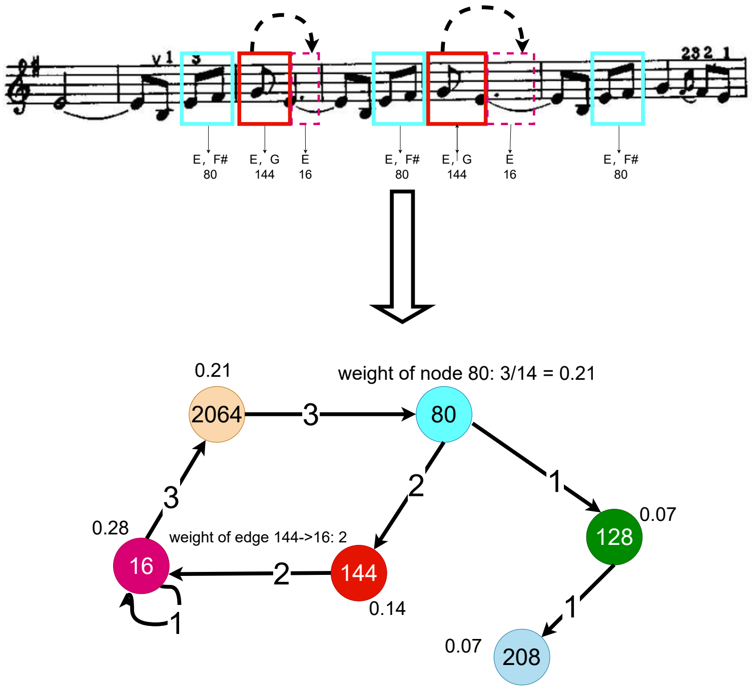

Fig. 3: An example of pitch segment graph. This figure shows an example for transforming music to pitch segment graph. In the graph, the label in the nodes represent an encoded vector value corresponding to the pitch segments (see Fig 1). Pitch segments (E, G) and (E) co-occur in music twice, during the fifth and the sixth beat, as well as during beats nine and ten (see red frame). Therefore, the edge weight between node 144 and node 16 should be 2. The pitch segment (E, F$\#$) appears three times in the 4th, 8th, and 12th beats (see blue frame). Therefore, the weight of node 80, which represents this pitch segment, should be $\frac{3}{14} = 0.21$.

at node 2064, and then at node 80 from node 2064. Thus the final pitch segment sequence is 18-2064-80.

To *better estimate* the pitch segment proximity, we carefully design the random walk procedure with the following *three* aspects.

First, each starting node should execute sufficient random walks to obtain a qualified sample for pitch segment embedding training. A node's number of walks should be in line with its role in the graph, as important nodes should enjoy more random walks than others. We assign each node with a node weight (see Eq. 2) that reflects a node's importance. The number of random walks rooted at node $v$ is computed using Eq. 3:

$$number\_walks(v) = \lfloor total\_walks \times PW_v \rfloor \tag{3}$$

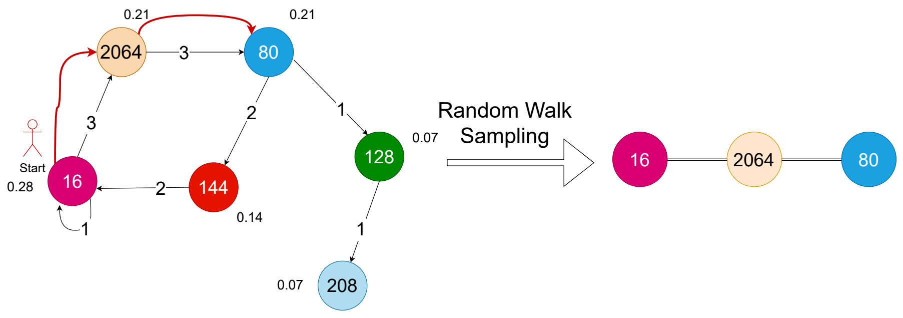

Fig. 4: The figure shows an example of random walk sampling. We choose node 16 as the starting point. In one of the random walking processes, we start from node 16 and arrive at node 2064, and then at node 80. Thus the final pitch segement sequence is 18-2064-80.

where $total\_walks$ is the total number of random walks set by the user, and $PW_v$ is the weight of a node $v$. Taking the pitch segment graph in Fig. 2 as an example, assuming we plan to sample 20 sequences from the graph, i.e., $total\_walks = 20$, then $PW_{16} = 0.28$. This means that we will sample 5 ($\lfloor 20 \times 0.28 \rfloor$) sequences starting from node 16. Once the starting points are determined, the next random walks will be conducted randomly. The transition probabilities between nodes are determined by the weights of the edges (i.e., the co-occurrence counts) between them.

Second, the transition probability is affected by directed edge weights. This forces random walks to preserve the pitch segment co-occurrence information as much as possible. The probability of traversing from node v to its neighbor node $x$ is $W_{vx}$ (see Eq. 1) divided by the sum of all neighboring directed weights. As shown in Fig. 4, probability to traverse from node 16 to node 2064 is $\frac{3}{3+1+2}$ (0.5).

Third, to capture both homophily and structural proximity between pitch segment of the graph, we employ Pecanpy [15], which offers a parallelized and accelerated random sampling methods for the graph in python, in our experiments.

## 3.3 Training pitch segment sequences with Skip-Gram

We use Skip-Gram to train the pitch segment embedding on the sampling corpus. The sampling corpus is composed of numerous random walks, which can be thought of pitch segment sequences. Skip-Gram maximizes the *co-occurrence probability* among the pitch segment that appear within a local context in a music sequence. In that way, Skip-Gram will produce very similar vector representations for two pitch segment if they are frequently within a local context of sampled pitch segment sequences.

### 3.4 Time and Space Complexity

As stated above, the Music-Graph2Vec algorithm consists of three key steps. First, constructing a graph representing pitch segments with a time complexity of $O(|N|)$. Second, the implementation of random walks to compute the pitch segments graph. Assuming the total number of sampled sequences is $n$, with each sequence having a length of $l$, the time complexity would be $O(nl)$. Third, the Skip-gram model is applied to the sampled pitch segments sequences. Since the size of the sampled sequences is $n \times l$, the time complexity is estimated as $O(nl \log |V|)$. Therefore, the overall time complexity of the algorithm is $O(|N| + nl + nl \log |V|)$. Hence, $|N|$ represents the total number of pitch segments across all music files, $|V|$ represents the number of sequences sampled during the random walk stage, $n$ denotes the number of sampled pitch segments sequences during the random walk stage, and $l$ signifies the count of distinct pitch segments encountered.

Regarding space complexity, Music-Graph2Vec requires storage for the pitch segments graph, generated pitch segments sequences, and the final pitch segment embeddings. The space complexity of the pitch segments graph is approximate $O(m|V|)$, where $m$ denotes the average degree of the graph. The space complexity of the generated pitch segments sequences is $O(nl)$. The space complexity of the pitch segments embeddings is $O(d|V|)$, where $d$ represents the dimensionality of the embeddings. Therefore, the overall space complexity of Music-Graph2Vec is $O(m|V| + nl + d|V|)$.

## 4 Experiment

Our baseline method is Word2Vec [5], which use the Word2Vec method to training the pitch segments embedding models on the music corpus. We compare Music-Graph2Vec with Word2Vec for the training performance (running time for training a model) and the accuracy of genre classification, which enables us to explore the prospect of pitch segment embeddings in various music tasks.

### 4.1 Data set Instruction

We conducted experiments using three MIDI data sets: The Largest MIDI Collection on the Internet [17], Top-MAGD [26], and Midiworld [18]. The statistic information of each Data set is shown in Table 1.
**The Largest MIDI Collection on the Internet**: The data set used in our experiments comprises MIDI files from multiple websites, providing a rich resource library that covers a wide range of music styles and genres, including pop music, classical music, electronic dance music, video game music, and film/TV show theme songs, among others. We selected MIDI files from eight distinct genres in the data set for our experiments, including 5703 songs from the Americana Folk genre, 99 songs from the Arabic and Tribal Rhythms genre, 4479 songs from the

Classical genre, 108 songs from the Dub genre, 431 songs from the Jazz genre, 1471 songs from the Metal Rock genre, 236 songs from the Ragtime genre, and 641 songs from the TV Themes genre.

**Top-MAGD**: The Lakh Data set [22] is a collection of MIDI files that are mapped to the Million Song data set (MSD) [1]. Within the MSD is a subset called the MSD Allmusic Genre data set (MAGD) that contains tracks with genre labels. Among the MAGD, there is a subset called top-MAGD [8], which consists of the 13 most frequently occurring genres. We selected all 13 genres from the top-MAGD for our experiments, including 21024 songs from the Pop/Rock genre, 3460 songs from the Electronic genre, 2410 songs from the Country genre, 2040 songs from the R&B genre, 1179 songs from the Jazz genre, 1410 songs from the Latin genre, 1008 songs from the International genre, 701 songs from the Rap genre, 698 songs from the Vocal genre, 496 songs from the New Age genre, 200 songs from the Folk genre, 141 songs from the Reggae genre, and 100 songs from the Blues genre.

**Midiworld**: Midiworld is a free MIDI file-sharing website that allows users to upload or download MIDI files. When uploading a MIDI file, users are required to specify the genre of the MIDI, with 16 genre options available. As a result, all MIDI files on the website are labelled with genre tags. Among these 16 genres, there are 65 songs in the Blues genre, 35 songs in the Christmas Carols genre, 382 songs in the Classical genre, 107 songs in the Country genre, 240 songs in the Dance genre, 41 songs in the Disney Themes genre, 10 songs in the Hip-Hop genre, 37 songs in the Jazz genre, 134 songs in the Movie Themes genre, 137 songs in the National Anthems genre, and 1105 songs in the Pop genre.

Table 1: Statistic of Music Data sets

| Data set | Total size | The number of songs | The average music length |
|---|---|---|---|
| the Largest MIDI | 28.3MB | 18557 | 356.61 beats |
| midiworld | 547MB | 5062 | 29156.39 beats |
| top-MAGD | 84.1MB | 44966 | 401.66 beats |
| top-MAGD 1KB subset | 1KB | 1 | 421.00 beats |
| top-MAGD 10KB subset | 10KB | 7 | 336.00 beats |
| top-MAGD 100KB subset | 100KB | 56 | 396.57 beats |
| top-MAGD 1MB subset | 1MB | 556 | 405.52 beats |
| top-MAGD 10MB subset | 10MB | 5605 | 402.10 beats |
| top-MAGD 20MB subset | 20MB | 11243 | 400.75 beats |
| top-MAGD 40MB subset | 40MB | 22471 | 401.00 beats |
| top-MAGD 80MB subset | 80MB | 44901 | 401.30 beats |

## 4.2 Time Consumption

In order to investigate the time with varying data set sizes, we randomly divided the top-MAGD data set with the most number of tracks into eight subsets, with sizes of 1KB, 10KB, 100KB, 1MB, 10MB, 20MB, and 40MB, respectively. Runtimes of top-MAGD and its subsets are shown in Fig. 5. Meanwhile, Table 2 shows the exact runtimes for these three data sets.

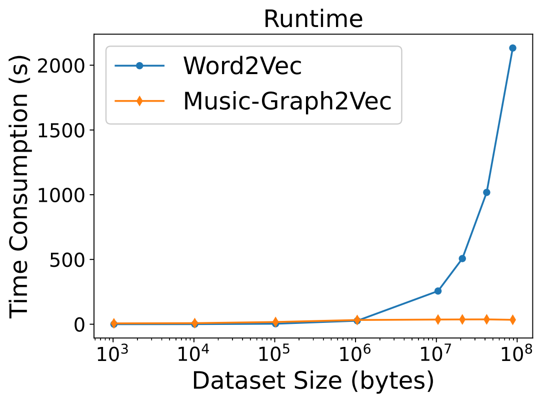

Fig. 5: The training time for Music-Graph2Vec and Word2Vec. As illustrated, the time required for training with Skip-gram increases steeply as the data set size grows, while the time required for training with Music-Graph2Vec remains consistently low. Noted that the runtimes for subsets of 1KB, 10KB, 100KB, 1MB, 10MB, 20MB, 40MB, and 80MB for Word2Vec are 0.09, 0.56, 3.61, 26.75, 256.26, 506.94, 1017.97, and 2133.75 seconds, respectively. For Music-Graph2Vec, the runtimes for subsets of 1KB, 10KB, 100KB, 1MB, 10MB, 20MB, 40MB, and 80MB are 6.57, 8.27, 17.07, 32.53, 36.01, 36.88, 37.32 and 38.91 seconds, respectively.

The empirical results show that: (i) Music-Graph2Vec is much faster than Word2Vec in training speed on **all** data sets, with the former 54.83 times faster than the latter, on top-MAGD; (ii) As training data grows, Music-Graph2Vec's runtime increases slower than Word2Vec's, indicating that larger data will see a greater runtime gap between them. For example, from 1KB data set to 80M data set, Music-Graph2Vec's runtime increases slightly from 6.57s to 38.91s, whereas Word2Vec's runtime rises from 0.09s to 2133.75s;

## 4.3 Music Genre Classification

In our experiments, we use the avg approach to calculate the music sequence embedding and apply these embedding for music genre classification tasks. Subsequently, in conjunction with the music genre labels, we employed the K-nearest neighbors (KNN) algorithm [6] for classification. The accuracy of the pitch segment embeddings trained using both methods was evaluated through ten-fold

Table 2: Exact Runtime

| data set | Method | Training Time (s) |
|---|---|---|
| The Largest MIDI | Word2Vec | 678.80 |
| | Music-Graph2Vec | 37.98 |
| Top-MAGD | Word2Vec | 2133.75 |
| | Music-Graph2Vec | 38.91 |
| midiworld | Word2Vec | 194.37 |
| | Music-Graph2Vec | 34.15 |

cross-validation, as illustrated in Fig. 6. The accuracy is calculated by the following equation [21].

$$Accuracy = \frac{The\ number\ of\ correct\ classification\ data}{The\ number\ of\ data} \tag{4}$$

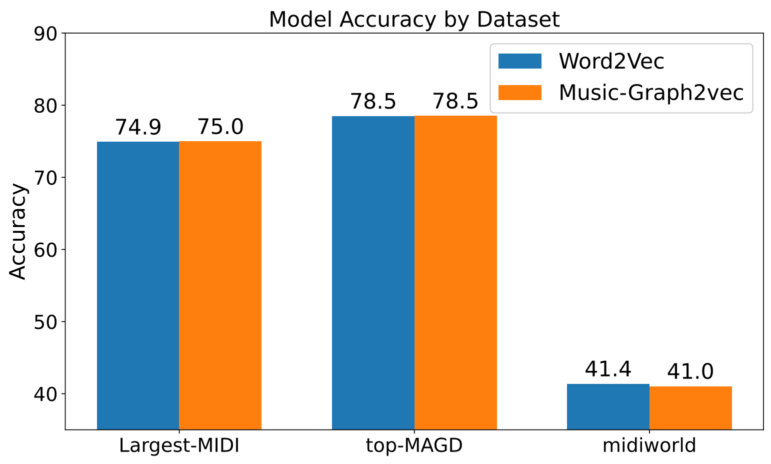

Fig. 6: The figure shows the accuracy of genre classification on each data set. The accuracy gap between the two methods in terms of training pitch segment embeddings was no more significant than **0.5%**, indicating that these methods yield similar results in training pitch segment embeddings.

As shown in Fig. 6, the accuracy gap between two methods is $|74.9\% - 75.0\%| = 0.1\%$ in Largest-MIDI data set, $|78.5\% - 78.5\%| = 0.0\%$ in top-MAGD data set, $|41.4\% - 41.0\%| = 0.4\%$ in midiworld data set. Overall, the accuracy gap between the two methods in terms of training pitch segment embeddings was no more significant than 0.1%, indicating that these methods yield similar results in training pitch segment embeddings. Notably, the accuracy performance could have been higher when using the midiworld data set for experimentation. The experimental results also revealed that the accuracy achieved using Word2Vec and Music-Graph2Vec for training pitch segment embeddings was only around 40% (41.4 for Word2Vecand 41.0 for Music-Graph2Vec). This may be attributed to the operation of the midiworld website, where both MIDI files and genre labels are uploaded by users, potentially resulting in some degree of data inaccuracy in the data set. Furthermore, the midiworld data set contains a relatively small number of songs, which could be a significant factor affecting the accuracy of the results.

## 5 Conclusion

In this paper, we proposed Music-Graph2Vec, which enhances the training of music pitch segment embeddings by incorporating pitch segment graphs and optimizing with random walk travelling. Experimental results demonstrated that Music-Graph2Vec achieved comparable accuracy to Skip-gram while exhibiting a remarkable speed advantage, up to **55** times faster than traditional methods in training pitch segment embeddings, with the same accuracy for the genre classification. Music-Graph2Vec has demonstrated a notable advantage over Skip-gram in terms of training efficiency for pitch segment embeddings, suggesting promising prospects for its application in music research. Moving forward, we will continue to delve deeper into the utilization of pitch segment embeddings in music research, as there is immense potential for further advancements in this field.

## References

1. Bertin-Mahieux, T., Ellis, D.P., Whitman, B., Lamere, P.: The million song dataset (2011)
2. Besson, M., Schön, D.: Comparison between language and music. Annals of the New York Academy of Sciences pp. 232–258 (2001)
3. Briot, J.P., Hadjeres, G., Pachet, F.D.: Deep learning techniques for music generation–a survey. arXiv preprint arXiv:1709.01620 (2017)
4. Brunner, G., Wang, Y., Wattenhofer, R., Zhao, S.: Symbolic music genre transfer with cyclegan. In: 2018 ieee 30th international conference on tools with artificial intelligence (ictai). pp. 786–793. IEEE (2018)
5. Chuan, C.H., Agres, K., Herremans, D.: From context to concept: exploring semantic relationships in music with word2vec. Neural Computing and Applications **32** (2020)
6. Cunningham, P., Delany, S.J.: k-nearest neighbour classifiers-a tutorial. ACM computing surveys (CSUR) **54**(6), 1–25 (2021)
7. Deng, J.J., Leung, C.H., Milani, A., Chen, L.: Emotional states associated with music: Classification, prediction of changes, and consideration in recommendation. ACM Transactions on Interactive Intelligent Systems (TiiS) **5**(1), 1–36 (2015)
8. Ferraro, A., Lemström, K.: On large-scale genre classification in symbolically encoded music by automatic identification of repeating patterns. In: Proceedings of the 5th International Conference on Digital Libraries for Musicology. Paris (2018)
9. Herremans, D., Chuan, C.H.: Modeling musical context with word2vec. arXiv preprint arXiv:1706.09088 (2017)

10. Hirai, T., Sawada, S.: Melody2vec: Distributed representations of melodic phrases based on melody segmentation. Journal of Information Processing **27**, 278–286 (2019)
11. Huang, C.Z.A., Duvenaud, D., Gajos, K.Z.: Chordripple: Recommending chords to help novice composers go beyond the ordinary. In: Proceedings of the 21st international conference on intelligent user interfaces. pp. 241–250 (2016)
12. Huang, C.Z.A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A.M., Hoffman, M.D., Dinculescu, M., Eck, D.: Music transformer. arXiv preprint arXiv:1809.04281 (2018)
13. Ji, S., Luo, J., Yang, X.: A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions. arXiv preprint arXiv:2011.06801 (2020)
14. Koelsch, S., Schmidt, B.H., Kansok, J.: Effects of musical expertise on the early right anterior negativity: an event-related brain potential study. Psychophysiology (2002)
15. Liu, R., Krishnan, A.: PecanPy: a fast, efficient and parallelized Python implementation of node2vec. Bioinformatics **37**(19), 3377–3379 (03 2021). https://doi.org/10.1093/bioinformatics/btab202, https://doi.org/10.1093/bioinformatics/btab202
16. Madjiheurem, S., Qu, L., Walder, C.: Chord2vec: Learning musical chord embeddings. In: Proceedings of the constructive machine learning workshop at 30th conference on neural information processing systems (NIPS2016), Barcelona, Spain (2016)
17. Man, T.M.: The largest midi collection on the internet, collected and sorted diligently by yours truly. Web (2015), https://www.reddit.com/r/datasets/comments/3akhxy/the_largest_midi_collection_on_the_internet/
18. MIDIWORLD: midiworld. Web (2022), https://midiworld.com
19. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
20. Oore, S., Simon, I., Dieleman, S., Eck, D., Simonyan, K.: This time with feeling: Learning expressive musical performance. Neural Computing and Applications **32**, 955–967 (2020)
21. Pelchat, N., Gelowitz, C.M.: Neural network music genre classification. Canadian Journal of Electrical and Computer Engineering **43**(3), 170–173 (2020)
22. Raffel, C.: Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching. 331 Ph. D. Ph.D. thesis, thesis, Columbia University (2016)
23. Schedl, M.: The lfm-1b dataset for music retrieval and recommendation. In: Proceedings of the 2016 ACM on international conference on multimedia retrieval. pp. 103–110 (2016)
24. Schedl, M.: Deep learning in music recommendation systems. Frontiers in Applied Mathematics and Statistics p. 44 (2019)
25. Schedl, M., Zamani, H., Chen, C.W., Deldjoo, Y., Elahi, M.: Current challenges and visions in music recommender systems research. International Journal of Multimedia Information Retrieval **7**, 95–116 (2018)
26. Schreiber, H.: Improving genre annotations for the million song dataset. In: ISMIR. Málaga (2015)
27. Toiviainen P, E.T.: Midi toolbox 1.1. Web (2016), https://github.com/miditoolbox/