SYMBOLIC MUSIC REPRESENTATIONS FOR CLASSIFICATION TASKS: A SYSTEMATIC EVALUATION

Huan Zhang[1] **Emmanouil Karystinaios**[2] **Simon Dixon**[1]
Gerhard Widmer[2] **Carlos Eduardo Cancino-Chacón**[2]
[1] Queen Mary University of London, United Kingdom
[2] Johannes Kepler University, Austra

ABSTRACT

Music Information Retrieval (MIR) has seen a recent surge in deep learning-based approaches, which often involve encoding symbolic music (i.e., music represented in terms of discrete note events) in an image-like or language-like fashion. However, symbolic music is neither an image nor a sentence, and research in the symbolic domain lacks a comprehensive overview of the different available representations. In this paper, we investigate matrix (piano roll), sequence, and graph representations and their corresponding neural architectures, in combination with symbolic scores and performances on three piece-level classification tasks. We also introduce a novel graph representation for symbolic performances and explore the capability of graph representations in global classification tasks. Our systematic evaluation shows advantages and limitations of each input representation. Our results suggest that the graph representation, as the newest and least explored among the three approaches, exhibits promising performance, while being more light-weight in training.

1. INTRODUCTION

The deep learning boom has profoundly impacted MIR, including research involving symbolic music representations (MIDI, scores, etc.). A large body of recent literature focuses on adapting existing architectures from computer vision and natural language processing to the field of symbolic MIR. These approaches often treat music data as an image (piano roll), as a sequence of language tokens, or, more recently, as a graph. However, a piece of music is neither an image nor a sentence or graph, therefore, a critical question still remains open concerning the choice of input representations for symbolic music.

A source of complexity in symbolic music arises from the different modalities of data such as scores and performances. A score contains information about music notation and often includes rich hierarchically structured information such as metrical structure and voicing. Symbolic music performances, on the other hand, such as those recorded on a MIDI-capable instrument, consist of a stream of controller events. Extracting a hierarchical structure from such a stream is not a trivial task [1–3]. Furthermore, such performance data omit some of the rich information that a score provides, such as pitch spelling and articulation markings, but instead, it can include information about expression, timing, local tempo, and performance dynamics.

Recent research has produced relatively large datasets containing scores and performances at the symbolic level, including efforts to align these [4–6]. Motivated by these developments, we present an attempt to shed light on questions revolving around the input representation of symbolic music for deep-learning-based MIR. We formulate an empirical framework where we test multiple input representations, models, and piece-level classification tasks.

In terms of input representations, we investigate piano rolls, tokenized sequences, and graphs. We evaluate multiple models based on these representations on three different tasks: composer classification, performer classification, and (playing) difficulty assessment. Furthermore, having datasets containing both performances and their corresponding scores such as ATEPP and ASAP [4, 5], allows us to apply each combination of representation and task to either score or performance. Our goal is to contribute an experimental overview of different symbolic music representations. The contributions of this work are threefold:

1. We investigate the performance and complexity of matrix, sequence and graph input representations, and their corresponding neural architectures (respectively Convolutional Neural Networks, Transformers, and Graph Neural Networks).
2. We compare the impact that the different information contained in symbolic scores and performances has on different piece-level classification tasks.
3. We introduce a new graph representation for symbolic performances, and explore the capability of graph representations in classification tasks.

2. RELATED WORK

The complexity of representing music data has been discussed in the literature [7–9]. Wiggins et al. [10] analyzed

the trade-offs of music representation systems with respect to expressive completeness and structural generality. In the age of deep learning, such considerations are still relevant regarding the variety of machine-readable representations such as piano rolls, MIDI-like sequences, NoteTuples, and Musical Spaces [11, 12]. In this section, we focus on three symbolic representations (matrix, sequence, and graphs) and discuss their respective strengths and limitations.

Music as a Matrix: Similar to audio spectrograms, a pitch-time representation that is typically used as input to a CNN, the piano roll representation of music naturally emerges as the symbolic equivalent. Piano rolls have been widely applied in tasks such as automatic music transcription [13, 14], classification of piece-level attributes such as difficulty and composer [15–18], as well as generation of music accompaniment or performed dynamics [19, 20].

A piano roll is a bare-bones representation of symbolic music data, and, therefore, information such as key signatures, articulation annotations, metrical structure, different instrument parts, and voicing structure are not encoded in the representation [11, 21].

Music as a Sequence: Modeling symbolic music as sequences has a longstanding tradition in MIR. The multiple viewpoint system is a sequence representation that has been widely used for music analysis, generation, and classification [22–25], as well as the basis for cognitively plausible models of expectation [26, 27]. In this system, musical elements are represented by viewpoints [28], which are abstract functions mapping musical events to abstract derived features like pitch, interval, and melodic contour.

With the advances of deep learning-based language models, sequential representation of music as *language tokens* has recently received a lot of attention in sequence-to-sequence generative tasks from automatic orchestration [29] to description-based medley generation [30]. Similar to a stream of MIDI messages, various tokenization schemes encode music features such as pitch, onset time, duration, and velocity sequentially. Besides generation, large-scale pre-training using music sequences has been applied to downstream music understanding tasks [31, 32].

However, tokenized music sequence representations create difficulty for models to learn the dependency of long contexts. Length reduction methods such as Byte Pair Encoding (BPE) [29, 33] aim to address the length overflow problem by replacing the occurrence of frequent subsequences with new tokens.

Music as a Graph: A musical score can also be seen as a graph where notes form the vertices and relations between notes define the edges. Jeong and al. [34] introduced a graph modeling of a musical score for generating expressive performances. Recently, Karystinaios and Widmer [35] presented a new modeling of the score graph based on three different note relations and a Graph Convolutional Network for cadence detection in classical music. A score graph can be homogeneous or heterogeneous, i.e. having one or several types of edges and/or vertices, respectively [36]. We will investigate both heterogeneous and homogeneous score graphs based on the representation used in [35].

Graph Neural Networks have gained popularity in recent years, however, graph learning inherently presents some limitations, such as over-smoothing in deep graph networks [37] and restrictions of Message Passing, where information in graph neural networks flows only between edge relations predetermined by the representation (in contrast to a Transformer architecture where everything is interconnected [38]).

3. METHODOLOGY

In this section, we describe the methodology followed, the corpora used, and the experiments conducted to investigate in-depth the different symbolic representations.

3.1 Representation Design

We briefly introduce a formal definition of each representation type, i.e. matrix, sequence, and graph. An example of the three representations is shown in Figure 1.

3.1.1 Matrix

We define as a matrix representation of music a 2-dimensional array $\mathbf{M} \in \mathbb{N}^{H \times W}$ that depicts musical notes on the time axis, commonly referred to as a piano roll. The vertical axis consists of 128 possible values attributed to the MIDI pitch of note events, where we add three more optional fields for the *una corda*, *sostenuto*, and sustain pedals only applied on the MIDI performances.

In this work, we experimented with multiple channels as used in Onsets and Frames [39]. The onset channel is a binarized roll with activations at onset timestamps, while the frame channel encodes the duration of the note and the velocity of the MIDI event. For scores, the velocity values are substituted by the voice index, i.e. the integer number assigned to a note to indicate the index among the number of independent voices. [1]

3.1.2 Sequence

A symbolic music sequence $\mathbf{S} \in \mathbb{N}^{1 \times N}$ is defined by a series of discrete tokens that represent attributes of notes. Vocabularies such as $V_{\texttt{pitch}}, V_{\texttt{TimeShift}}, V_{\texttt{Vel}}$ assign semantic meanings to tokens, and different tokenization schemes translate into different grammars of sequence construction. In this work, we test three popular tokenization schemes: *MIDILike* [40, 41], *REMI* [42], and *CompoundWord* [43] and use the implementation of the MidiTok library [44].

As there is no existing tokenizer for processing scores, we implemented custom MusicXML tokenizers following MidiTok's framework, in the style of *REMI* as well as *CompoundWord*. The major difference is the timing of bars and event positions, as well as the addition of score-specific tokens such as $V_{\texttt{KeySig}}, V_{\texttt{Voice}}$. [2]

Byte Pair Encoding (BPE) is a tokenizer add-on technique that has recently been applied to music sequence learning [33]. It consists of a data compression technique

[1] This voice information is commonly available in formats such as MusicXML, **Kern, and MEI.

[2] Full documentation is provided with our open-source tokenizer in the project repository.

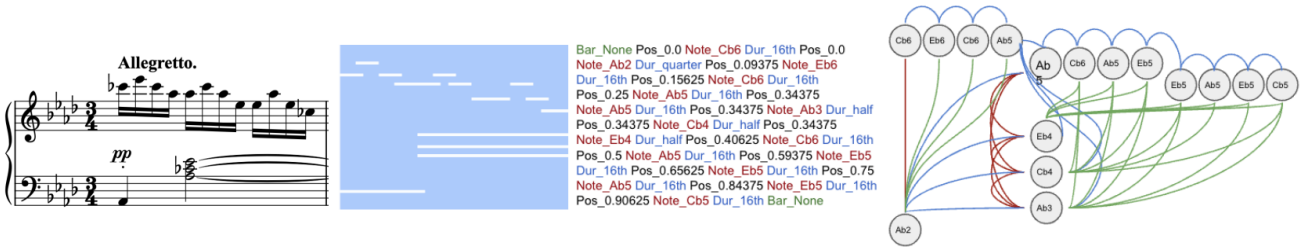

Figure 1. Excerpt of Schubert's *Impromptu Op. 90 No.4* and its input visualizations (from left to right): generic matrix, sequence (REMI-like) and graph.

that replaces the most common token subsequences in a corpus with newly created tokens. BPE increases the vocabulary size and shortens the sequence length. We follow the best results from [33] and adopt a BPE with 4 times the original vocabulary size. On average, this reduced our sequence length between $55-65\%$ in both datasets.

3.1.3 Graph

A homogeneous score graph G is defined by a tuple (V, E) of vertices and edges. V is the set of notes in a musical score and $E \subseteq V \times V$. Given a score with N notes, we extract a matrix of k-dimensional note-wise features $X \in \mathbb{R}^{N \times k}$ based on features contained in the score or performance. A heterogenous score graph $G = (V, E, \mathcal{R})$ also includes a set of relation types $\mathcal{R}$ such that for every edge $e \in E$, e is of type $r \in \mathcal{R}$ if a condition defined by r holds. In our work, we consider the following relations between two notes u, v which define the edges $e \in E$:

- u and v have the same onset, i.e. $on(v) = on(u)$, then $r =$ onset;
- The offset of u is the onset of v, i.e. $\mathit{off}(u) = on(v)$, then $r =$ consecutive;
- The onset of u lies between the onset of v and the offset of v, i.e. $on(v) < on(u) \wedge on(u) < \mathit{off}(v)$, then $r =$ overlap.

The above relations only hold in the case of score graphs. To adapt this to performance graphs, we use a window tolerance t_{tol}, such that if two notes $(u, v) \in E$ and:

- $|on(v) - on(u)| < t_{\text{tol}}$, then $r =$ onset;
- $|\mathit{off}(u) - on(v)| < t_{\text{tol}}$, then $r =$ consecutive;
- $on(v) < on(u) \wedge on(u) < \mathit{off}(v)$, then $r =$ overlap.

In our configurations, for all graphs created from performance MIDI, we set $t_{\text{tol}} = 30\,\text{ms}$, a perceptual threshold of expressive timing [45]. In addition to the above relations, we consider the possibility of adding an inversely directed edge for the overlap and the consecutive edge types, and we name the inclusion of such edges *inverse edges*. For a homogeneous graph G_{hom} and heterogeneous graph G_{het}, $e \in G_{\text{hom}} \implies e \in G_{\text{het}}$.

The node features X are divided into two categories, the basic and the advanced features. The basic features are implicitly contained in any score or performance note such as one-hot encoding of pitch class and octave of the note's pitch, and duration information. The advanced features

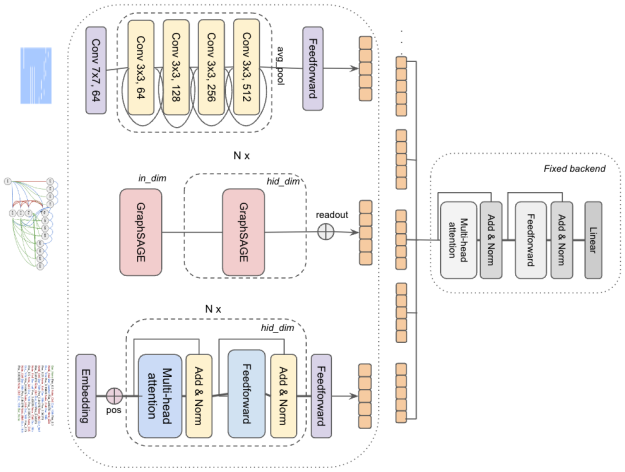

Figure 2. Left: front end for three representations, matrix, graph, and sequence, from top to bottom. Right: fixed back end with attention modules.

contains articulation, dynamics, and notation information from the *Partitura* python package [46]. The detailed computation of these features can be found in original partitura paper [47] and the basis mixer [48].

3.1.4 Information Levels

Given the differences in information captured by symbolic scores and performances (Sec. 1), we run experiments with separate levels of used information. For the base comparison experiments, we input the basic level of information that is present in both modalities: pitch, duration and onset. The advanced level of information for performance includes dynamics (MIDI velocity) and pedals, while for score includes the voice index (Sec. 3.1) as well as score markings such as articulation and dynamics. The results and comparison of each level of information, also with respect to different tasks, will be discussed in Section 4.3.

3.2 Modelling Pipelines

In this work, we evaluate the input representations under the same training pipeline of different piece-level classification tasks, as discussed in Section 3.3. We split our training architecture into two parts, a front end that projects a window of musical context into a 64-dimensional embedding, and a back end that aggregates the embedding for final prediction. The front end is representation-specific while the back end

		ASAP-performance		ASAP-score		ATEPP-performance		ATEPP-score	
		ACC	*F1*	*ACC*	*F1*	*ACC*	*F1*	*ACC*	*F1*
Matrix									
Resl	Chnl								
400	On+Fm	0.59±0.04	0.18±0.02	0.59±0.03	0.18±0.01	0.24±0.05	0.20±0.04	**0.25±0.02**	0.16±0.03
600	On+Fm	0.62±0.06	0.21±0.03	**0.61±0.07**	**0.19±0.02**	0.28±0.01	**0.22±0.03**	0.24±0.02	0.16±0.04
800	Fm	0.62±0.04	**0.21±0.02**	0.58±0.06	0.18±0.03	0.22±0.03	0.17±0.01	0.22±0.02	**0.18±0.03**
800	On+Fm	**0.63±0.04**	0.20±0.01	0.57±0.04	0.18±0.03	**0.28±0.02**	0.22±0.01	0.22±0.04	0.14±0.02
Sequence									
Tokn	BPE								
MidiLike	×	**0.53±0.05**	**0.16±0.02**	N/A	N/A	0.18±0.04	0.10±0.02	N/A	N/A
REMI	×	0.51±0.04	0.15±0.02	0.43±0.04	**0.14±0.01**	**0.23±0.04**	0.10±0.02	0.23±0.04	**0.13±0.02**
CP	×	0.48±0.02	0.09±0.05	**0.45±0.05**	0.10±0.01	0.11±0.02	0.09±0.01	0.17±0.06	0.11±0.04
MidiLike	4	0.52±0.04	0.15±0.02	N/A	N/A	0.17±0.03	0.12±0.01	N/A	N/A
REMI	4	0.51±0.02	0.15±0.01	0.43±0.03	0.13±0.01	0.21±0.01	**0.13±0.03**	**0.23±0.03**	0.13±0.01
Graph									
Bi-dir	Multi-rel								
×	×	0.56±0.01	0.17±0.02	0.51±0.05	0.16±0.02	0.22±0.02	0.10±0.03	0.23±0.03	0.21±0.05
×	✓	0.58±0.03	0.19±0.01	**0.54±0.05**	**0.17±0.02**	**0.27±0.03**	0.13±0.02	**0.29±0.10**	0.18±0.06
✓	✓	**0.62±0.02**	**0.21±0.01**	0.50±0.04	0.17±0.01	0.23±0.04	**0.16±0.03**	0.27±0.06	**0.22±0.03**

Table 1. Composer classification results for all representations, on all target subsets of our datasets on the composer classification task using only basic level features. For each subset of data, we present the accuracy score and the macro F1 score with 8-fold cross-validation. See Section 4.1 for explanation of the parameters.

rests fixed. For a fair comparison, we ensure that the same amount of musical context is given for different front ends to learn. For MIDI performances we fix a window of 60 s, and for symbolic scores, we choose a window of 120 beats given that 120 bpm is a common tempo for music.

For the front end, we employ a commonly used architecture for each respective representation domain:

Matrix: Convolutional neural network based on ResNet [49] blocks with channel numbers adapted to our input.

Sequence: Transformer-encoder [50] front end with positional encoding. Each layer includes multi-head attention with 16 heads followed by an Add & Norm layer. For the combined tokens *CPWord* we add separate embedding layers for each token category in the front end.

Graph: Our graph convolution network (GCN) is built by stacking GraphSAGE blocks [51] followed by a global mean pooling layer. We experiment with both heterogeneous and homogeneous GraphSAGE. Note that a heterogeneous network has r times more parameters, where r is the number of distinct edge relation types.

For the fixed back end, we used a multi-head attention block with linear projection heads to the desired number of classes, as shown in Figure 2. To minimize the impact of model capacity on our comparative discussion, we carried out an ablation study to understand the size of the architecture proportional to each kind of representation (Sec. 4.2).

3.3 Tasks and Datasets

In this work, we focus on three tasks: composer classification, performer classification, and difficulty assessment. Each one of these tasks is a piece-level task since a label is attributed per piece. The composer classification consists of predicting the composer of the piece. The performer classification involves the prediction of the performer among a list of predefined performers included in the data source. Finally, difficulty assessment involves the prediction of a number between 1-9, with 1 being easy and 9 being hard. The difficulty labels were assembled from Henle Music. [3]

To evaluate the aforementioned tasks, we use two large-scale collections of Western classical piano music that contain corresponding symbolic scores (MusicXML files) and performances (MIDI files), ASAP (1067 performances, 245 scores) and ATEPP (11742 performances, 415 scores). Both datasets contain individual files per movement.

For the composer classification task, we exclude the least populated composer classes for balance in experiments, resulting in 10 classes for the ASAP dataset and 9 classes for the ATEPP dataset. The performer classification task uses MIDI performances of ATEPP with 20 classes. For difficulty, given that both ASAP and ATEPP datasets focus on concert repertoire, the actual classes used range from difficulty 4-9. [4] For all experiments, we use an eight-fold cross-validation evaluation where 85% of our data is used for training and 15% for testing in each fold.

3.4 Training

We performed hyperparameter optimization sweeps to determine the optimal learning rate and model hyperparameters. Our convergence criteria include early stopping at the 60 epoch breakpoint with the patience parameter set at 0.005 on the validation accuracy. All our experiments are trained on a single A5000 GPU, and the best models, training logs,

[3] Henle Music difficulty labels, `https://www.henle.de/en/about-us/levels-of-difficulty-piano/`

[4] The full distribution of the classes for each task is shown in the supplementary material.

and the code is available in the repository. [5]

4. EXPERIMENTS AND RESULTS

To evaluate the different representations we performed three experiments. Our first experiment focuses on a detailed comparison of the predictive accuracy of the three representations/architectures applied to the composer classification task, since it is the most well-understood task among the three. The second experiment studies the impact of model capacity (number of trainable parameters) per representation. Our last experiment investigates the effect of different levels of input features (see Section 3.1.4) on the three tasks.

4.1 Representations for Composer Classification

Our first experiment is a comparative analysis of the three representations on our two datasets, in the domains of both MIDI performance and MusicXML score with basic level features. For each representation group we test different configurations, i.e. for matrix we experiment with the channel (Chnl) and timestep resolution (Resl), for sequence we change the tokenization scheme (Tokn) and apply BPE, and for graph we investigate the effect of homogeneous or heterogeneous graphs (Multi-rel) and the addition of inverse edges (Bi-dir) (see Sec. 3.1). In Table 1, we present for each data subset the accuracy score and the macro F1 score and their respective standard deviations under 8-fold cross-validation (see Sec. 3.3).

In terms of observations per representation, the matrix representation results indicate no significant differences under different experimental configurations. For sequence representations, the *MIDILike* and *REMI* tokenization schemes yield comparable performance. However, our experiments suggest that *CPWord* is a more challenging representation to learn in the same setting. Concerning the BPE technique, no significant difference is observed between results with 4 times the original vocabulary and the non-BPE version.

Our graph-based models exhibit similar performance regardless of the configuration of the graph edges. In particular, the effect of reverse edges is not significant, and homogeneous graph convolution already achieves similar results to heterogeneous graph convolutional models, which indicates that implicit structural information contained in the heterogeneous approach is not strictly necessary for piece-level classification tasks.

Overall, we observe that three representations show small performance differences in given experiments, with the matrix-CNN approach having the overall best metric across the experiment groups and sequence have the worst.

Finally, we would like to discuss the *album effect*, which concerns the tendency of classification models to learn non-intended features, such as acoustic features in pieces of the same album [52]. In our case, this effect concerns different performances of the same piece that may give away cues for classification. Training with the entire corpus of performance MIDI, which involves different interpretations of the same piece, yields an average accuracy of 90% (see supplementary material), which is 30% higher for the `ASAP-perf` group. To address this issue, we fix the splits to only contain unseen pieces in the test set, which reduced the accuracy score gap between performance and score. This issue has often been overlooked in literature [53, 54] and a commonly-used dataset split is not piece-specific [16]. Given the recent development of large score-performance datasets, we wish to establish a scientifically correct evaluation split taking into consideration the *piece effect*.

[5] `https://github.com/anusfoil/SymRep`

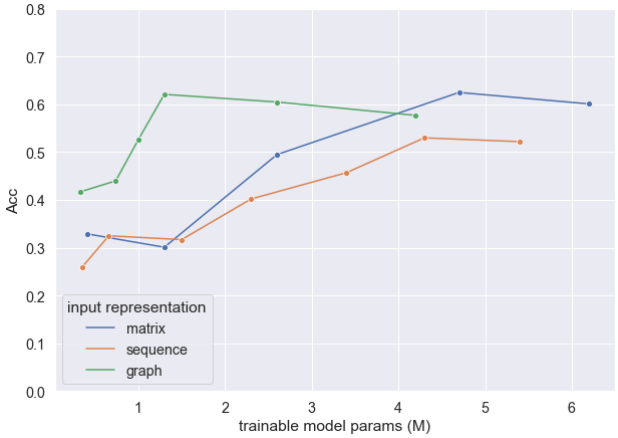

Figure 3. Model capacity vs. macro F1 score for each representation approaches on the `ASAP-composer` task.

4.2 Complexity

In our second experiment we investigate the impact of model capacity for each representation on the composer classification task using the ASAP dataset. We experiment with different hidden dimensions h and the number of layers N on each architecture corresponding to each of the three representations (Sec 3.2), and show our results in Figure 3. Overall, we observe that the GCN achieves its best performance using 1.3M parameters, while architectures for matrix and sequence achieve a similar accuracy at around three times the number of parameters.

Another observation concerns the use of large models for piece-level classification tasks on symbolic data. Large convolution models such as ResNet-18/34/50 [16] are substantially over-parametrized, as our results suggest we can achieve similar results using a reduced version of ResNet-8, using less than half the parameters of the smallest used ResNet architecture. Similar observations can be made for transformers, where scaling the model beyond 4.3M parameters does not further improve the performance. Our most efficient transformer encoder consists of 4 layers of attention modules with a hidden dimension of 256, significantly less than transformers used in previous related work [33].

Finally, we note one aspect of our results after scaling our graph network. While *oversmoothing* [37] (features of graph vertices converging to the same value) is a well-known challenge to train deep GCN, our best performing model is a relatively deep and narrow network consisting of 5 layers with a hidden dimension of 64. One possible interpretation is that convergence of node features does not

	Composer		Performer	Difficulty	
	perf	score	perf (ATEPP)	perf	score
Matrix					
basic feats	**0.625**	0.572	**0.364**	0.403	**0.420**
advanced feats	0.618	**0.577**	0.342	**0.411**	0.415
Sequence					
basic feats	**0.530**	**0.447**	0.287	**0.438**	**0.368**
advanced feats	0.513	0.393	**0.292**	0.426	0.349
Graph					
basic feats	**0.607**	0.545	0.305	**0.373**	0.361
advanced feats	0.598	**0.697**	**0.323**	0.356	**0.405**

Table 2. Accuracy of three identification tasks on the ASAP dataset, with basic or higher-level features.

complicate training in the graph-level classification context.

4.3 Comparison of Feature Levels and Tasks

As discussed in Section 3.1.4, we are also interested in understanding the impact of different levels of features on the three classification tasks. With this motivation, we performed our third set of experiments, where we adopted the best configuration of models explored in experiment 1 (see Section 4.1). We report the accuracy results in Table 2.

Our results indicate that MIDI performances and MusicXML scores have similar capabilities for distinguishing composers and difficulty. Furthermore, matrix and sequence approaches exhibit better results when learning with performances compared to scores. For the difficulty classification task, in particular, all three representations achieved approximately 40% accuracy on the 6 difficulty levels. Performer classification is more challenging since the difference lies in the timing nuances and dynamic changes instead of the pitch information, which are more prominent in our input representations. In the 20-way classification, our approaches generally achieved around 30% accuracy.

Our observations suggest that the addition of advanced features has a variable impact on the representations. Interestingly, the addition of advanced features does not improve the training from sequence representations in most experiments, which can possibly be explained by the increase in vocabulary size and relative sparsity of such information. Graph structures benefit from the addition of voice edges, especially in the representation of scores, where the performance boosts for both composer and difficulty classification. Notably, the `graph-score` with advanced features configuration achieved the best result in score-based composer classification, when jointly compared with Table 1.

4.4 Transformer vs. GNN: Are We Learning the Same Set of Musical Edges?

A transformer can be seen as a special case of Graph Neural Networks [38]. Assuming a fully connected graph where vertices are tokens in a sequence, we can draw parallels between a GCN and learned attention in a transformer block.

Therefore, we examine attention weights between `NoteOn` tokens in an effort to understand how our graph representation of the score relates to the sequence-based representation. For all pairs of `NoteOn` tokens from music sequences, we output their attention values and compute the correlation with the aggregated adjacency matrix (with all musical edges constructed in Sec. 3.1). Across the test set of ASAP composer classification on scores, there is a weak positive correlation, with Pearson's value of 0.212.

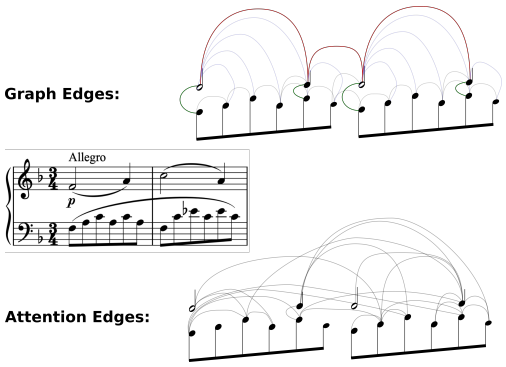

Figure 4. Visualization of graph edges (all edge types aggregated) and the attention among `NoteOn` tokens for the first measures of *Mozart Piano Sonata No.12, 1st mvt*.

In Figure 4, we visualize two measures of music with its constructed graph edges, and the attention across `NoteOn` tokens. We can observe some structural similarities, especially the overlap pattern in both measures, but overall the learned attention spans are much more global while graph edges connect nodes within a local range.

5. DISCUSSION AND FUTURE WORK

In this paper, we presented a series of systematic experiments to investigate the impact of symbolic representations for three piece-level tasks. In terms of simple *classification performance*, we found that for a given task, different representations showed small performance differences, but no clear pattern of superiority emerged. The matrix results were marginally better on average, and usually more robust to hyper-parameter changes. More advanced features were beneficial only for certain tasks and representations.

The *graph representation*, as the newest and least explored among the three approaches, exhibits promising performance, while being more light-weight (in terms of required model complexity – cf. Fig. 3). We observe that homogeneous graphs produce comparable results to heterogeneous graphs for our piece-level classification tasks, and deep GCNs perform better despite over-smoothing. As graphs are arguably a more natural representation for structured artifacts such as musical scores, we believe that they should merit more detailed studies in the future.

Our model complexity experiments demonstrated that commonly used architectures in the literature are larger than necessary for our tasks, as the same results can be achieved with smaller architectures (Section 4.2). Furthermore, we discussed the *album effect* in score-performance datasets, where multiple interpretations of the same composition may cause information leakage. Our results indicate the profound impact of the album effect, and we introduce new evaluation splits to guard against this effect.

6. ACKNOWLEDGEMENTS

This work is supported by the UKRI Centre for Doctoral Training in Artificial Intelligence and Music, funded by UK Research and Innovation [grant number EP/S022694/1], also by the European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme, grant agreement No. 101019375 (*Whither Music?*).

7. REFERENCES

[1] L. Liu, Q. Kong, V. Morfi, and E. Benetos, "Performance MIDI-to-score conversion by neural beat tracking," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2022.

[2] D. Temperley, "A unified probabilistic model for polyphonic music analysis," *Journal of New Music Research*, vol. 38, no. 1, pp. 3–18, 2009. [Online]. Available: https://doi.org/10.1080/09298210902928495

[3] D. Temperley, *The Cognition of Basic Musical Structures*. MIT Press, 2004.

[4] H. Zhang, J. Tang, S. Rafee, S. Dixon, and G. Fazekas, "ATEPP: A Dataset of Automatically Transcribed Expressive Piano Performance," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2022.

[5] S. D. Peter, C. E. Cancino-Chacón, F. Foscarin, A. P. McLeod, F. Henkel, E. Karystinaios, and G. Widmer, "Automatic note-level score-to-performance alignments in the ASAP dataset," *Transactions of International Society for Music Information Retrieval (in press)*, 2023.

[6] F. Foscarin, E. Karystinaios, S. D. Peter, C. Cancino-Chacón, M. Grachten, and G. Widmer, "The match file format: Encoding alignments between scores and performances," in *Proceedings of the Music Encoding Conference (MEC)*, 2022.

[7] I. Xenakis, *Formalized Music: Thoughts and Mathematics in Composition*, 1992.

[8] M. Harris, A. Smaill, and G. Wiggins, "Representing Music Symbolically," in *IX Colloquio di Informatica Musicale (Venice)*, 1991. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.473

[9] M. Babbit, "The use of computers in musicological research," *Perspectives of New Music*, vol. 3, no. 2, pp. 74–83, 1965.

[10] G. Wiggins, E. Miranda, A. Smaill, and M. Harris, "A Framework for the Evaluation of Music Representation Systems," *Computer Music Journal*, vol. 17, no. 3, pp. 31–42, 1993. [Online]. Available: https://about.jstor.org/terms

[11] C. Walder, "Modelling symbolic music: Beyond the piano roll," in *Journal of Machine Learning Research*, vol. 63, 2016, pp. 174–189.

[12] M. Prang, "Representation learning for symbolic music," Ph.D. dissertation, IRCAM, 2021. [Online]. Available: https://hal.archives-ouvertes.fr/tel-03329980

[13] E. Benetos, A. Klapuri, and S. Dixon, "Score-informed transcription for automatic piano tutoring," in *European Signal Processing Conference (EUSIPCO)*, 2012. [Online]. Available: http://c4dm.eecs.qmul.ac.uk/rdr/

[14] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-Resolution Piano Transcription with Pedals by Regressing Onset and Offset Times," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 29, pp. 3707–3717, 2021.

[15] Y. Ghatas, M. Fayek, and M. Hadhoud, "A hybrid deep learning approach for musical difficulty estimation of piano symbolic music," *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 10 183–10 196, 2022.

[16] S. Kim, H. Lee, S. Park, J. Lee, and K. Choi, "Deep Composer Classification Using Symbolic Representation," in *International Society for Music Information Retrieval (ISMIR) Late Breaking Demo (LBD)*, 2020. [Online]. Available: http://arxiv.org/abs/2010.00823

[17] G. Velarde, T. Weyde, C. E. Cancino-Chacón, D. Meredith, and M. Grachten, "Composer recognition based on 2D-filtered piano-rolls," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016. [Online]. Available: https://www.semanticscholar.org/paper/Composer-Recognition-Based-on-2D-Filtered-Velarde-Weyde/2ee8df37e3f5363c573b2aeed2243034ea638f71

[18] F. Foscarin, K. Hoedt, V. Praher, A. Flexer, and G. Widmer, "Concept-Based Techniques for "Musicologist-friendly" Explanations in a Deep Music Classifier," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2022. [Online]. Available: http://arxiv.org/abs/2208.12485

[19] H. W. Dong, W. Y. Hsiao, L. C. Yang, and Y. H. Yang, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018. [Online]. Available: https://salu133445.github.io/musegan/

[20] S. van Herwaarden, M. Grachten, W. de Haas, and W. Bas de Haas, "Predicting expressive dynamics in piano performances using neural networks," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[21] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, *Deep Learning Techniques for Music Generation – A Survey*, 2017. [Online]. Available: http://arxiv.org/abs/1709.01620

[22] D. Conklin and I. H. Witten, "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, vol. 24, no. 1, pp. 51–73, 1995. [Online]. Available: https://doi.org/10.1080/09298219508570672

[23] D. Conklin, "Multiple viewpoint systems for music classification," *Journal of New Music Research*, vol. 42, no. 1, pp. 19–26, 2013. [Online]. Available: https://doi.org/10.1080/09298215.2013.776611

[24] R. P. Whorley and D. Conklin, "Music generation from statistical models of harmony," *Journal of New Music Research*, vol. 45, no. 2, pp. 160–183, 2016. [Online]. Available: https://doi.org/10.1080/09298215.2016.1173708

[25] D. Conklin, "Chord sequence generation with semiotic patterns," *Journal of Mathematics and Music*, vol. 10, no. 2, pp. 92–106, 2016. [Online]. Available: https://doi.org/10.1080/17459737.2016.1188172

[26] M. T. Pearce, "Statistical learning and probabilistic prediction in music cognition: Mechanisms of stylistic enculturation," *Annals of the New York Academy of Sciences*, vol. 1423, no. 1, pp. 378–395, 2018. [Online]. Available: https://nyaspubs.onlinelibrary.wiley.com/doi/abs/10.1111/nyas.13654

[27] M. Pearce, "The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition," Ph.D. dissertation, City University of London, UK, 2005.

[28] D. Conklin and I. H. Witten, "Multiple Viewpoint Systems for Music Prediction," *Journal of New Music Research*, vol. 24, no. 1, pp. 51–73, 1995.

[29] J. Liu, Y. Dong, Z. Cheng, X. Zhang, X. Li, F. Yu, and M. Sun, "Symphony Generation with Permutation Invariant Language Model," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2022. [Online]. Available: http://arxiv.org/abs/2205.05448

[30] D. von Rütte, L. Biggio, Y. Kilcher, and T. Hofmann, "FIGARO: Generating Symbolic Music with Fine-Grained Artistic Control," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023. [Online]. Available: http://arxiv.org/abs/2201.10936

[31] M. Keller, G. Loiseau, and L. Bigo, "What Musical Knowledge Does Self-Attention Learn?" in *Proceedings of the 2nd Workshop on NLP for Music and Spoken Audio (NLP4MusA)*, 2021, pp. 6–10. [Online]. Available: https://aclanthology.org/2021.nlp4musa-1.2

[32] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T. Y. Liu, "MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP*, 2021.

[33] N. Fradet, J.-P. Briot, F. Chhel, A. E. F. Seghrouchni, and N. Gutowski, "Byte Pair Encoding for Symbolic Music," 2023. [Online]. Available: http://arxiv.org/abs/2301.11975

[34] D. Jeong, T. Kwon, Y. Kim, and J. Nam, "Graph neural network for music score data and modeling expressive piano performance," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3060–3070.

[35] E. Karystinaios and G. Widmer, "Cadence detection in symbolic classical music using graph neural networks," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2022.

[36] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2016.

[37] G. Li, M. Muller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?" in *Proceedings of the IEEE International Conference on Computer Vision*, 2019. [Online]. Available: https://sites.google.com/view/deep-gcns

[38] P. Veličković, "Everything is Connected: Graph Neural Networks," *Artificial Intelligence (AI) Methodology in Structural Biology*, 2023. [Online]. Available: http://arxiv.org/abs/2301.08210

[39] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 50–57.

[40] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: learning expressive musical performance," *Neural Computing and Applications*, vol. 32, no. 4, pp. 955–967, 2018.

[41] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer," in *Proceedings of the International Conference of Learning Representations (ICLR)*, 2019. [Online]. Available: http://arxiv.org/abs/1809.04281

[42] Y. S. Huang and Y. H. Yang, "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020.

[43] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs," in *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021.

[44] N. Fradet, J.-P. Briot, F. Chhel, A. El Fallah Seghrouchni, and N. Gutowski, "Miditok: a Python Package for Midi File Tokenization," in *International*

Society for Music Information Retrieval (ISMIR) Late Breaking Demo (LBD), 2021.

[45] W. Goebl, "Melody lead in piano performance: Expressive device or artifact?" *The Journal of the Acoustical Society of America*, vol. 110, p. 641, 2001. [Online]. Available: https://asa.scitation.org/doi/10.1121/1.1376133

[46] C. Cancino-Chacón, S. D. Peter, E. Karystinaios, F. Foscarin, M. Grachten, and G. Widmer, "Partitura: A Python package for symbolic music processing," in *Proceedings of the Music Encoding Conference (MEC)*, 2022.

[47] C. Cancino-Chacón, S. D. Peter, E. Karystinaios, F. Foscarin, M. Grachten, and G. Widmer, "Partitura: A Python Package for Symbolic Music Processing," pp. 1–9, 2022. [Online]. Available: http://arxiv.org/abs/2206.01071

[48] C. E. Cancino-Chacón, M. Grachten, W. Goebl, and G. Widmer, "Computational Models of Expressive Music Performance: A Comprehensive and Critical Review," *Frontiers in Digital Humanities*, vol. 5, no. October, pp. 1–23, 2018.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[51] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[52] A. Flexer, "A closer look on artist filters for musical genre classification," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2007.

[53] G. Micchi, "A neural network for composer classification," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR) Late-Breading Demo (LBD)*, 2018.

[54] Q. Kong, K. Choi, and Y. Wang, "Large-Scale MIDI-Based Composer Classification," in *arXiv*, 2020.

8. APPENDIX

8.1 Album effect

As mentioned in the main paper Section 4, the *Album Effect* remains a non-trivial issue in similar classification tasks. Here we present in Table 4 the same content as the original table (Table 1) from the paper which contains results from the experiment that is trained on the entire performance corpus with overlapping interpretations. Training under this non-piece-specific split, we achieved comparable accuracy (93%) with the literature [16].

8.2 Complexity

8.2.1 Memory

Given that the same amount of music context is input into the models, we are interested in understanding the memory efficiency of the representations. We used the native `numpy` and `cuda` functions to monitor the memory of data and memory changes during training.

In terms the representation of a single piece of data, sequence is the most compact one while matrix takes $70\times$ more space, given that a lot of redundant pixels are taken in the 2D representation. The size of graph varies depending on the number of nodes and edges, but overall it is in between that of the matrix and sequence.

However, during training we can observe that the sequence is the least memory-efficient representation during training, and it takes $30\times$ compares to the memory usage of matrix and graphs. Given the quadratic complexity of transformer-like architectures, the training memory needed is one of the major limitation of sequence compared to the other representations.

	KB / seg	KB / piece	Training step (MB)
Mtr	819.2	5129.6 ± 3332.7	185.9 ± 105.9
Seq	12.8	77.8 ± 56.7	5548.9 ± 1736.2
Gph	100.5 ± 57.3	610.9 ± 300.0	125.2 ± 103.4

Table 3. Size estimation of each representation with basic level features from ASAP-perf data. We include the average size per segment (60s), average size per piece (as piece have different length), as well as the average allocated memory increase during each training step with a batch size of 1.

8.2.2 Convergence epochs

During training, we also observed a difference in the time it takes the models to convergence, given the 60 epochs convergence criteria defined in Sec 3.4. We first performed learning rate search using pytorch lightning's learning rate finder. Under the suggested learning rate, among different `ASAP-perf` experiment of composer classification, the matrix have on average 143.0±24.7 epochs to converge, the sequence and the graph have 132.0±31.1 and 262.0±55.7 epochs. During training, the graph models have relatively slower learning progress.

8.3 Dataset class distributions

We present our dataset class distribution for each task in the Table 8.4.

8.4 Silence and voice edges

Besides the onset, consecutive and overlap edges in Sec 3.1, we also add optional silence edges (edges that bridge over silence) to ensure a connected graph. A silence edge $E_{silence}$ is added between a node that's not connected by any consecutive edge and the time-wise closest node before it. The silence edge doesn't carry much music semantic meaning, and its main purpose is to prevent the disjoint subgraphs formed by distinct music sections, in which stops information flow in training.

In the advanced representation of score graph, we input the voicing information as voice edges. Given that we can't guarantee the consistency of voice annotation in MusicXML scores (as they are mostly labeled for visual purposes like beaming), we limit the voice edge connection within a measure: If two notes are labelled with the same voice, then they are connected by a voice edge E_{voice}.

		ASAP-performance		ASAP-score		ATEPP-performance		ATEPP-score	
		ACC	*F1*	*ACC*	*F1*	*ACC*	*F1*	*ACC*	*F1*
Matrix									
Resl	Chnl								
400	On+Fm	0.926±0.02	0.796±0.06	0.598±0.03	0.177±0.01	0.905±0.04	0.796±0.03	**0.246±0.02**	0.156±0.03
600	On+Fm	**0.931±0.01**	0.800±0.07	**0.613±0.07**	**0.186±0.02**	**0.930±0.05**	0.818±0.03	0.238±0.02	0.156±0.04
800	Fm	0.925±0.02	0.723±0.11	0.583±0.06	0.182±0.03	0.891±0.02	0.737±0.02	0.221±0.02	**0.181±0.03**
800	On+Fm	0.926±0.02	**0.812±0.05**	0.572±0.04	0.185±0.03	0.932±0.03	**0.832±0.01**	0.225±0.04	0.138±0.02
Sequence									
Tokn	BPE								
MidiLike	×	0.860±0.03	0.674±0.11	N/A	N/A	**0.926±0.01**	**0.769±0.01**	N/A	N/A
REMI	×	0.783±0.04	0.521±0.05	0.431±0.04	**0.138±0.01**	0.910±0.01	0.729±0.02	0.229±0.04	**0.129±0.02**
CP	×	0.679±0.08	0.331±0.06	**0.447±0.05**	0.099±0.01	0.864±0.02	0.556±0.01	0.171±0.06	0.107±0.04
MidiLike	4	**0.905±0.02**	**0.727±0.06**	N/A	N/A	0.895±0.01	0.691±0.01	N/A	N/A
REMI	4	0.862±0.01	0.692±0.07	0.432±0.03	0.132±0.01	0.826±0.04	0.529±0.03	**0.234±0.03**	0.125±0.01
Graph									
Bi-dir	Multi-rel								
×	×	0.768±0.03	0.500±0.08	0.509±0.05	0.163±0.02	0.788±0.03	0.501±0.06	0.226±0.03	0.205±0.05
×	✓	**0.861±0.03**	**0.763±0.03**	**0.545±0.05**	**0.174±0.02**	**0.928±0.01**	**0.781±0.03**	**0.289±0.10**	0.176±0.06
✓	✓	0.833±0.03	0.703±0.11	0.500±0.04	0.173±0.01	0.897±0.01	0.767±0.02	0.271±0.06	**0.217±0.03**

Table 4. Base experiment composer classification results with the entire performance MIDI corpus and no piece-specific split.

ASAP composer		**ATEPP composer**		**ATEPP performer**		**ASAP difficulty**	
Beethoven	195	Beethoven	3033	Richter	1581	9	164
Bach	163	Chopin	1739	Ashkenazy	1188	8	176
Chopin	162	Mozart	653	Arrau	833	7	132
Liszt	67	Schubert	264	Brendel	743	6	150
Schubert	55	Debussy	254	Kempff	609	5	56
Schumann	26	Schumann	243	Barenboim	603	4	23
Haydn	23	Bach	231	Schiff	595		
Mozart	10	Ravel	169	Horowitz	576		
Scriabin	9	Liszt	122	Gulda	459		
Ravel	9			Gieseking	362		
				Gould	326		
				Gilels	322		
				Perahia	288		
				Pollini	256		
				Argerich	240		
				Schnabel	240		
				François	234		
				Uchida	210		
				Casadesus	164		
				Lugansky	125		

Table 5. Dataset class distribution for the tasks. The performer task is in regards to the distribution of the performed MIDI, and the other three columns are in regards to the MusicXML score.