IMPACT OF TIME AND NOTE DURATION TOKENIZATIONS ON DEEP LEARNING SYMBOLIC MUSIC MODELING

Nathan Fradet[1,2] **Nicolas Gutowski**[3] **Fabien Chhel**[3,4] **Jean-Pierre Briot**[1]
[1] Sorbonne University, CNRS, LIP6, F-75005 Paris
[2] Aubay, Boulogne-Billancourt, France
[3] University of Angers, LERIA, 49000 Angers, France
[4] ESEO-TECH / ERIS, 49100 Angers, France
nathan.fradet@lip6.fr

ABSTRACT

Symbolic music is widely used in various deep learning tasks, including generation, transcription, synthesis, and Music Information Retrieval (MIR). It is mostly employed with discrete models like Transformers, which require music to be tokenized, i.e., formatted into sequences of distinct elements called tokens. Tokenization can be performed in different ways, and recent research has focused on developing more efficient methods. However, the key differences between these methods are often unclear, and few studies have compared them. In this work, we analyze the current common tokenization methods and experiment with time and note duration representations. We compare the performance of these two impactful criteria on several tasks, including composer classification, emotion classification, music generation, and sequence representation. We demonstrate that explicit information leads to better results depending on the task.

1. INTRODUCTION

Most tasks involving using deep learning with symbolic music [1] are performed with discrete models, such as Transformers [2]. To use these models, the music must first be formatted into sequences of distinct elements, commonly called tokens. For instance, a token can represent a note attribute or a time event. The set of all known tokens is commonly called the vocabulary, and each token is associated to a unique integer id. These ids are used as input and output of models.

Compared to text, tokenizing music provides greater flexibility, as a musical piece can be played by different instruments and composed of multiple simultaneous notes, each having several properties such as pitch, duration and velocity. As a result, it is necessary to represent these elements in conjunction with the time dimension. To achieve this, researchers have developed various methods of tokenizing music, which are introduced in the next section.

While these works offer model performance comparisons between tokenization strategies, their main differences or similarities are not always clearly stated. Few experiments have been conducted to compare model performances using different tokenization strategies. Additionally, these studies mostly focus on music generation, for which evaluations are performed on results obtained autoregressively, which accumulates biases [3] and is arguably difficult to evaluate [4].

This paper's primary contribution is a thorough and well-designed comparison of common tokenization techniques. Our focus is on two critical aspects: the representation of time and note duration. We believe that they are significant and impactful design choices for any music tokenization approach. Through experiments on composer classification, emotion classification, music generation, and sequence representation, we demonstrate that these design choices produce varying results depending on the task, model type, and inference process. Autoregressive generation benefits from explicit note duration and time shift tokens, while explicit note offset is more discriminating better suited for contrastive learning approaches.

We present next the related works, followed by an analysis of music tokenization, experimental results, and finally a conclusion. The source code is available for reproducibility. [1]

2. DECOMPOSING MUSIC TOKENIZATION

2.1 Related works

Early works using discrete models for symbolic music, such as DeepBach [5] or FolkRNN [6], rely on specific tokenizations often tied to their training data. Since then, researchers introduced more general representations applicable to any kind of music. The most commonly used are *Midi-Like* [7] and *REMI* [8]. The former tokenizes music by representing tokens as the same types of events from the MIDI protocol, while the latter represents time with *Bar* and *Position* tokens and note durations with explicit

[1] https://github.com/Natooz/time-duration-music-modeling

	Time		Note duration	
Tokenization	`TimeShift`	`Bar + Pos.`	`Duration`	`NoteOff`
MIDI-Like [7]	√	-	-	√
REMI [8]	-	√	√	-
Structured [17]	√	-	√	-
TSD [15]	√	-	√	-
Octuple [10]	-	√	√	-

Table 1: Time and note duration representations of common tokenizations. `Pos.` stands for Position.

Duration tokens. Additionally, *REMI* includes tokens with additional information such as chords and tempo.

More recently, researchers have focused on improving the efficiency of models with new tokenizations techniques: *Compound Word* [9], *Octuple* [10] and *PopMAG* [11] merge embedding vectors before passing them to the model; 2) LakhNES [12] and [13], SymphonyNet [14] and [15] use tokens combining several values, such as pitch and vocabulary.

2.2 Music tokenization design

When analyzing the possible designs of music tokenization, we can distinguish seven key dimensions:

- **Time**: Type of token representing time, either *TimeShift* indicating time movements, or *Bar* and *Position* indicating new bars and the positions of the notes within them. We can also consider the unit of *Time-Shift* tokens, either in beats or in seconds. [2]
- **Notes duration**: How notes durations are represented, with either *Duration* or *NoteOff* tokens.
- **Pitch**: Most works use tokens representing absolute pitch values, although recent work shed light on the expressiveness gain of representing as intervals instead [16];
- **Multitrack representation**: The representation of several music tracks in a sequence, i.e., how are the notes linked to their associated track.
- **Additional information**: Any additional information such as chords, tempo, rests, note density. Velocity can also falls in this category;
- **Downsampling**: How "continuous-like" features are downsampled into discrete sets, e.g. the 128 velocity values reduced to 16 values;
- **Sequence compression**: Methods to reduce the sequence lengths, such as merging tokens and embedding vectors.

As time and note duration can both be represented in two different ways, existing tokenizations can be easily classified based on these dimensions, as shown in table 1. However, other dimensions offer a broader spectrum of potential designs.

For instance multitrack can be represented by `Program` tokens [3] preceding notes as in FIGARO [18], distinct tracks sequences separated by `Program` tokens as in MMM [19], combined note and instrument tokens as LakhNes [12] and MuseNet [13], or merging `Program` embeddings with the associated note tokens (MMT [20], MusicBert [10]). One could even infer each sequence separately and lately model their relationships with operations aggregating their hidden states an in ColBERT [21].

The MIDI protocol supports a set of effects and metadata that can also be represented when tokenizing symbolic music, such as tempo, time signature, sustain pedal or control changes. Some works also include explicit `Chord` tokens, detected with rule-based methods. Nevertheless, only a few works experimented with such additional tokens so far ([8, 22]).

Previous works have mainly compared tokenization strategies by evaluating models with automatic and sometimes subjective (human) metrics, but often do not proceed to comparisons between the ways to represent one of the dimensions we introduced previously. [8] compared results for the generation task, for the use of `Bar` and `Position` tokens versus `TimeShift` in seconds and beats.

To the best of our knowledge, no comprehensive work and empirical analysis have fairly compared these possible tokenization choices. Conducting such an assessment would require an extensive survey. In this paper, we specifically focus on the time and note duration representations, as they are the two main characteristics present in every tokenization.

We want to highlight the importance of the explicit information carried by the token types, as they directly impact the performances of models. `TimeShift` tokens represent explicit time movements, and especially the time distances between successive notes. On the other hand, `Bar` and `Position` tokens bring explicit information on the absolute positions (within bars) of the notes, but not the onset distances between notes. One could assume that the former might help to model melodies, and the latter rhythm and structure. For note duration, `Duration` tokens intuitively express the absolute durations of the notes, while `NoteOff` tokens explicitly indicates the offset times. With `NoteOff`, a model would have to model note durations from the combinations of previous time tokens.

Our experiments aim to demonstrate the impact of different combinations of time and note duration tokens on model performance and which combinations are suitable for different tasks. Next, we introduce our methodology.

3. METHODOLOGY

3.1 Models and trainings

For all experiments, we use the Transformer architecture [2], with the same model dimensions: 12 layers, with di-

[2] In this paper we only treat of the beat unit. The MIDI protocol represents time in *tick* unit, which value is proportional to the time division (in ticks per beat) and tempo. Hence, working with seconds would require a conversion from ticks.

[3] Following the conventional programs from the MIDI protocol.

mension of 768 units, 12 attention heads and inner feed-forward layers of 3072.

For classification and sequence representation, it is first pretrained on 100k steps and a learning rate of 10^{-4}, then finetuned on 50k steps and a learning rate of 3×10^{-5}, with a batch size of 48 examples. An exception is made for the EMOPIA dataset, for which we set 30k pretraining steps and 15k finetuning steps, as it is fairly small. These models are based on the BERT [23] implementation of the Transformers library [24]. We use the same pretraining than the original BERT: 1) from 15% of the input tokens, 80% is masked with a special `MASK` token, and 20% is randomized; 2) half of the inputs have 50% of their tokens (starting from the end) shuffled and separated with a special `SEP` token, and the model is trained to detect if the second part is the next of the first.

For generation, the model is based on the GPT2 implementation of the Transformers library [24]: it uses a causal attention mask, so that for each element in the sequence, the model can only attend to the current and previous elements. The training is performed with teacher forcing, the cross-entropy loss is defined as: $\ell = -\sum_{t=1}^{n} \log p_\theta(x_t|\mathbf{x}_{\leq n})$.

All trainings are performed on V100 GPUs, using automatic mixed precision [25], the Adam optimizer [26] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, and dropout, weight decay and a gradient clip norm of respectively 10^{-1}, 10^{-2} and 3. Learning rates follow a warm-up schedule: they are initially set to 0, and increase to their default value during the first 30% of training, then slowly decrease back to 0.

10% of the data is used for validation during training, and 15% to test models. Inputs contains 384 to 512 tokens, and begin with a `BOS` (Beginning of Sequence) token and end with a `EOS` (End of Sequence) one.

3.2 Tokenizations

We investigate here the four combinations of possible time and note duration representation. In the results, we refer to them as *TS* (`TimeShift`), *Pos* (`Position`), *Dur* (`Duration`) and *NOff* (`NoteOff`). It is worth noting that *TS* + *Dur* is equivalent to *TSD* [15] and *Structured* [17], *TS* + *NOff* is equivalent to *MIDI-Like* [7], and *Pos* + *Dur* is equivalent to *REMI* (without additional tokens for chords and tempo).

We apply different resolutions for `Duration` and `TimeShift` token values: those up to one beat are downsampled to 8 samples per beat (spb), those from one to two beats to 4 spb, those from two to four beats to 2 spb, and those from four to eight beats to 1 spb. Thus, short notes are represented more precisely than longer ones. `Position` tokens are downsampled to 8 spb, resulting in 32 different tokens as we only consider the 4/* time signature. This allows to represent the 16^{th} note. We only consider pitches within the recommended range for piano (program 0) specified in the General MIDI 2 specifications [4]: 21 to 108. We then deduplicate all duplicated notes. Velocities are downsampled to 8 distinct values. No additional token (e.g., `Chord`, `Tempo`) is used.

We perform data augmentation by creating variations of the original data with pitches increased and decreased by two octaves, and velocity by one value. Finally, following [15], we use Byte Pair Encoding to build the vocabularies up to 2k tokens for generation and 5k for other tasks. All these preprocessing and tokenization steps were performed with MidiTok [27].

[4] Available on the MIDI Manufacturers Association website.

4. GENERATION

For the generative task, we use the POP909 dataset [28]. The models start with prompt made of between 384 to 512 tokens, then autoregressively generate 512 additional tokens. Evaluation of generated results remains an open issue [4]. Previous work often perform measures of similarity of certain features such as pitch range or class, between prompts and generated results, alongside human evaluations. Feature similarity is however arguably not very insightful: a generated result could have very similar features to its prompts while being of poor quality. Human evaluations, while being more reliable on the quality can also induce biases. Besides, [8] already shows results on an experiment similar to ours.

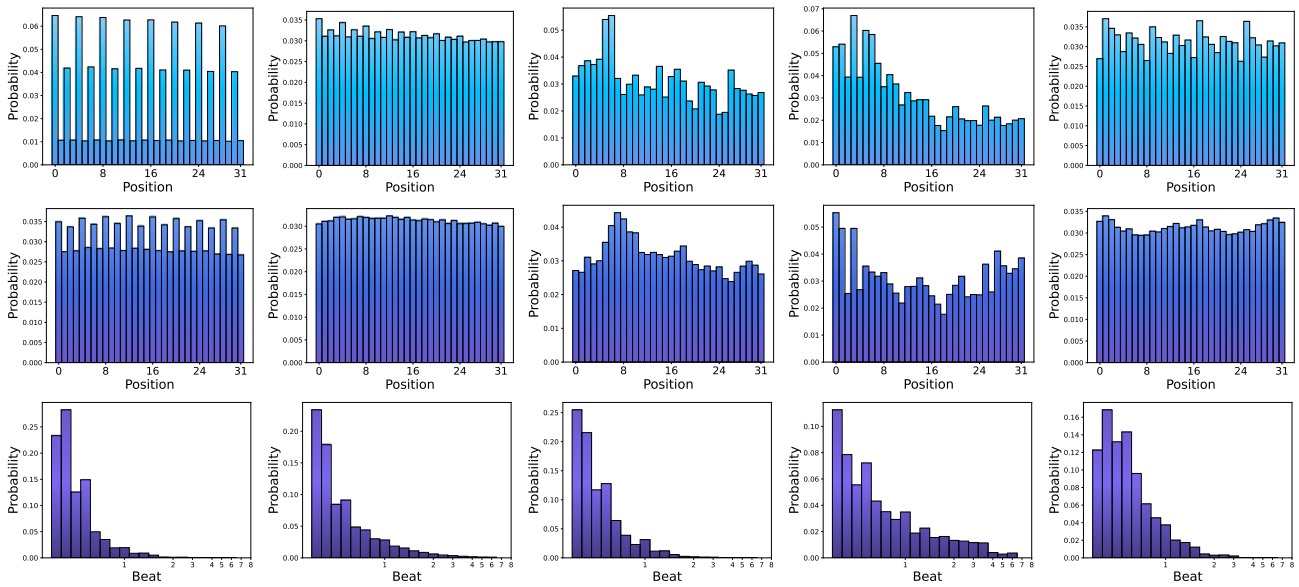
Hence we choose to evaluate results on the ratios of prediction errors: Token Syntax Error (TSE) [15]. This metric is bias-free and directly linked to the design choices of the tokenizations. It allows us to measure how a model achieves to make reliable predictions based on the input context and the knowledge it learned.

We use the categories from [15]:

- $\mathbf{TSE_{type}}$: an error of type, e.g., when the model predicts a token of an incompatible type with the previous one.
- $\mathbf{TSE_{time}}$: a wrong predicted `Position` value, that goes back or stay in time.
- $\mathbf{TSE_{dupn}}$ (duplicated note): a note predicted whereas it was already being played at the current time being.
- $\mathbf{TSE_{nnof}}$ (no NoteOff): a `NoteOn` token been predicted with no following `NoteOff` token to end it.
- $\mathbf{TSE_{nnon}}$ (no NoteOn): `NoteOff` token predicted whereas this note was not being played.

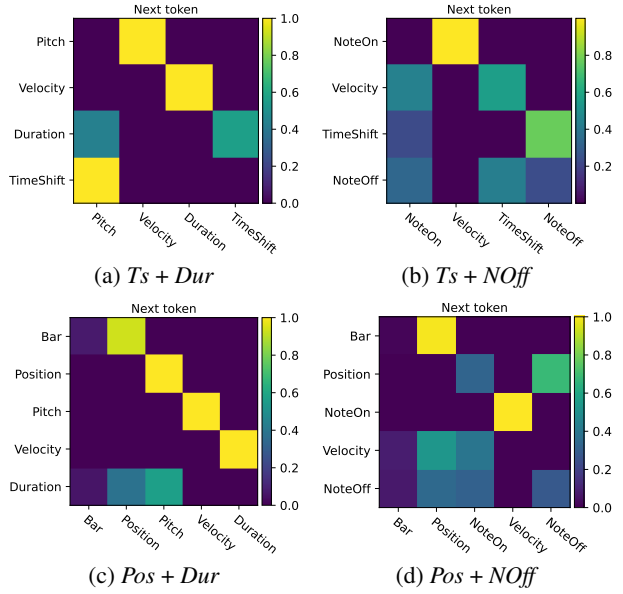
For each generated token, a rule-based function analyzes its type and value to determine if both are valid, or which type of error was made otherwise. The overall number of errors is normalized by the number of predicted tokens.

The results are reported in table 2. We first observe that the type error ratios are lower than in other categories. This is excepted since it is less computationally demanding to model the possible next types depending solely on the last one, rather than on the value of the predicted token, for



(a) *Ts + Dur* (b) *Ts + NOff* (c) *Pos + Dur* (d) *Pos + NOff* (e) POP909 dataset

Figure 1: Histograms of the note onset positions within bars (top-row), note offset positions within bars (middle-row) and note durations (bottom-row) of the generated notes. There are 32 possible positions within a bar, numerated from 0 (beginning of bar) to 31 (last 32^{th} note). The durations are expressed in beats, ranging from a 32^{th} note to 8 beats.



(a) *Ts + Dur* (b) *Ts + NOff*

(c) *Pos + Dur* (d) *Pos + NOff*

Figure 2: Token type succession heatmaps of the generated results. The horizontal axis denotes the next token type per from the ones on the vertical axis. Each row is normalized to a sum of 1.

which the validity depends on a the whole previous context.

`Position` tokens bring almost no type errors, but a noticeable proportion of time errors. When decoding tokens to notes, this means that the time may go backward, and resulting in sections of overlapping notes.

Although `Duration` tokens seem to bring slightly more note duplication errors, the use of `NoteOn` and `NoteOff` tokens results in a considerable proportion of

Tokenization	$\mathbf{TSE_{type}}\downarrow$	$\mathbf{TSE_{time}}\downarrow$	$\mathbf{TSE_{dupn}}\downarrow$	$\mathbf{TSE_{nnon}}\downarrow$	$\mathbf{TSE_{nnof}}\downarrow$
TS + Dur	$< 10^{-3}$	-	0.014	-	-
TS + NOff	$< 10^{-3}$	-	0.001	0.109	0.040
Pos + Dur	0.002	0.113	0.032	-	-
Pos + NOff	0.002	0.127	0.005	0.095	0.066

Table 2: Prediction error ratios when performing autoregressive generation. - symbol stands for not concerned, and can be interpreted as 0.

note prediction errors. `NoteOff` tokens predicted while the associated note was not being played ($\mathbf{TSE_{nnon}}$) does not have undesirable consequences when decoding tokens to notes, but it pointlessly extends the sequence, reducing the efficiency of the model, and may mislead the next token predictions. Additionaly, `NoteOn` tokens predicted without associated `NoteOff` ($\mathbf{TSE_{nnof}}$) result in notes not properly ended. This error can only be handled by applying a maximum note duration after decoding. Explicit `Duration` tokens allows to specify in advance this information, for both short and long notes. Conversely, with `NoteOff` tokens, the note duration information is implicit and inferred by the combinations of `NoteOn`, `NoteOff` and time tokens. This can be interpreted as an extra effort for the model. Consequently, some uncertainty on the duration accumulates over autoregressive steps during generation. Based on these results, the best tradeoff ensuring good predictions seems to represent time with `TimeShift` tokens and note duration with `Duration` tokens.

In fig. 1 we observe the positions within bars and durations of the generated notes. In all cases, onset positions are more distributed at the beginning of the bars. This is especially the case with `Bar` and `Position` tokens, for which we may find unexpected rests at the end of bars,

when Bar tokens are predicted during the generation before that the current bar is completed. The *TS + Dur* combination places note onsets much more on even positions. The probability mass of *TimeShift* tokens (especially for short values) seems to be much higher. However, this is not the case for the *TS + NOff* combination, as TimeShift tokens have to be predicted to move the time on odd positions of note offsets. As shown in fig. 2, right after the model is likely to predict a next note, resulting in evenly distributed onset distribution.

Finally, the use of NoteOff tokens tends to produce longer note durations, especially when combined with Position tokens. In this last case, we can assume that the model might "forget" the notes currently being played, and that it struggles more to model their durations that have to be implicitly deduced from the past Bar and Position tokens.

Tokenization	Top-20 composers ↑	Top-100 composers ↑	Emotion ↑
TS + Dur	**0.973**	**0.941**	**0.983**
TS + NOff	0.962	0.930	0.962
Pos + Dur	0.969	0.927	0.963
Pos + NOff	0.963	0.925	0.956

Table 3: Accuracy on classification tasks.

5. CLASSIFICATION

For some classification tasks, symbolic music is arguably better suited than audio or piano roll. This is particularly true for classical music feature classification, such as composer [29]. Mono-instrument music with complex melodies and harmonies and no particular audio effect benefit from being represented as discrete for classification and modeling tasks. Given this, it felt important to us to conduct experiments on such task.

We choose to experiment with the GiantMIDI [30] dataset for composer classification and the EMOPIA [31] dataset for emotion classification. The results, as shown in table 3, indicate that there is very little difference between the various tokenization methods. However, the combination of TimeShift and Duration consistently outperforms the others by one point

The classification task involves modeling the patterns from data that are characteristic to composers or emotions. Here, it seems that the time distance between notes, and their explicit duration play a role in these task, more than note offsets or onset positions. This comes with no surprise for the composer classification task, considering that the data is largely composed of complex music with dense melodies and harmonies, featuring mostly short successive notes. Intuitively, patterns of note successions and chords are more easily distinguishable with explicit durations. With implicit note durations, the overall patterns must be deduced by the combinations of NoteOn and NoteOff tokens while keeping track of the time.

6. SEQUENCE REPRESENTATION

The last task that we wished to explore is sequence representation. It consists in obtaining a fixed size embedding representation of an input sequence of tokens $p_\theta : \mathbb{V}^L \mapsto \mathbb{R}^d$. Here $\mathbb{V} \subset \mathbb{N}$ denotes the token ids of the vocabulary $\mathcal{V}$, L is the variable input sequence length, and d the size of embeddings. In other words, the model learns to project an input token sequence into a embedding space, thus providing a universal representation. We find this task interesting and well-suited to assess model performances as it directly trains it to model the relationships between tokens within the input sequence and between different representations themselves. While the real-world applications of this task for symbolic music are currently limited, it serves as a useful benchmarking technique for measuring how tokenization impacts the learning of models.

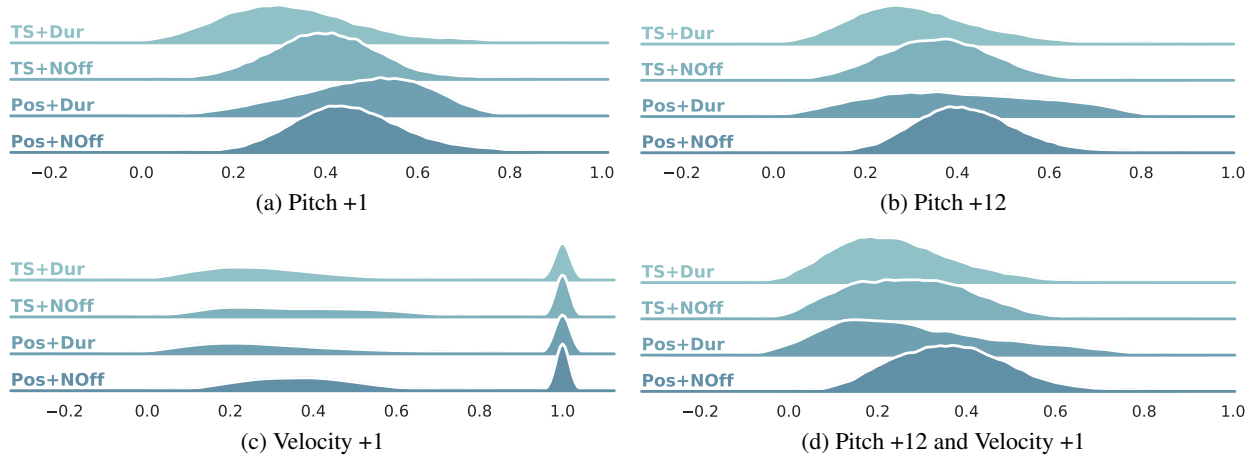
This task has previously been addressed in natural language processing by SentenceBERT [32] or SimCSE [33]. We adopted the approach of the latter, which uses contrastive learning to train the model to learn sequence representations, for which similar inputs have higher cosine similarities. The sequence embedding is obtained by performing a pooling operation on the output hidden states of the model. We decided to use the last hidden state of the BOS token position, as it yielded good results with SimCSE [33] [5]. We trained the models with the dropout method: during training, a batch of n sequences $\mathcal{X} = \{\mathbf{x}_i\}_{i=0}^n$ is passed twice to the model, but with different dropout masks, resulting in different output sequence embeddings $\mathcal{Z} = \{\mathbf{z}_i\}_{i=0}^N$ and $\bar{\mathcal{Z}} = \{\bar{\mathbf{z}}_i\}_{i=0}^N$. Although the dropout altered the outputs, most of the input information is still accessible to the model. Hence, we expect pairs of sequence embeddings $(\mathbf{z}_i, \bar{\mathbf{z}}_i)$ to be similar, so having a high cosine similarity. To achieve this objective, we train the model with a loss function defined by the cross-entropy for in-batch pairwise cosine similarities (sim):

$$\ell_i = -\log \frac{e^{\mathrm{sim}(\mathbf{z}_i, \bar{\mathbf{z}}_i)/\tau}}{\sum_{j=1}^N e^{\mathrm{sim}(\mathbf{z}_i, \bar{\mathbf{z}}_j)/\tau}} \quad (1)$$

As a result, the model will effectively learn to create similar sequence embeddings for similar inputs, while pushing apart those with dissimilarities. We kept a 0.1 dropout value to train the models, and used the GiantMIDI dataset [30].

Evaluation of sequence representation is intuitively performed by measuring the distances and similarities of pairs of similar sequences. We resort to data augmentation by shifting the pitch and velocity of the sequences in order to get pairs of similar music sequences. The augmented data keeps most of the information of the original data. As such, the models are expected to produce similar embeddings for pairs of original-augmented sequence. Ideally, the cosine similarity should be high, yet not to be equal to 1, as this would indicate that the model fails to capture the differences between the two sequences. The results, presented in fig. 3, indicate that Position-based tokenizations per-

[5] SimCSE uses a CLS token which is equivalent to BOS in our case.



(a) Pitch +1 (b) Pitch +12

(c) Velocity +1 (d) Pitch +12 and Velocity +1

Figure 3: Density plots of cosine similarities between pairs of original and augmented token sequences.

form slightly better. Therefore, it appears that explicit note onset and offset positions information facilitates models to obtains a universal musical representation.

Unlike classification, the contrastive learning objective models the similarities and dissimilarities between examples in the same batch. In this context, note onset and offset positions appear to be helpful for the models to distinguish music.

We also note the contrasting results when augmenting the velocity. Increasing it by one unit, which would be equivalent to playing just a little bit louder, have arguably a very small impact. As a result, the models mostly produces embeddings that are almost identical for the original and the augmented sequences, but also exhibits uncertainty for a notable proportion of samples.

To complement these results, we estimated the isotropy of sets of sequence embeddings. Isotropy measures the uniformity of the variance of a set points in a space. More intuitively, in an isotropic space, the embeddings are evenly distributed. It has been associated with improved performances in natural language tasks [34–36], because embeddings are more equally distant proportionally to the density of their area, and are in turn more distinct and distinguishable. We choose to estimate it with the intrinsic dimension of the sets of embeddings. Intrinsic dimension is the number of dimensions required to represent a set of points. It can be estimated by several manners [37]. We choose Principal Component Analysis (PCA) [38], method of moments (MOM) [39], Two Nearest Neighbors (TwoNN) [40] and FisherS [41]. The results, reported in table 4, show that the embeddings created from the *Pos + Dur* combination tend to occupy more space across the dimension of the model, and are potentially better distributed.

Tokenization	lPCA ↑	MOM ↑	TwoNN ↑	FisherS ↑
TS + Dur	**213**	42.6	34.3	17.5
TS + NOff	161	43.7	32.7	17.5
Pos + Dur	146	39.1	33.1	17.1
Pos + NOff	177	**45.2**	**35.6**	**17.8**

Table 4: Intrinsic dimension of sequence embeddings, as an estimation of isotropy.

7. CONCLUSION

We have discussed the importance of different aspects of symbolic music tokenization, and focused on two major ones: the time and note duration representations. We showed that different tokenization strategies can lead to different model performances due to the explicit information carried by tokens, depending on the task at hand.

Explicitly representing note duration leads to better classification accuracy as it helps the models to capture the melodies and harmonies of a music. Modeling durations, when represented implicitly, adds an extra effort to the model. However, the note offset position information it brings have been found to be more discriminative and effective in our contrastive learning experiment.

For music generation, the time representation plays a significant role, for which the note onset and offsets distributions vary due to the successions of token types. Implicit note durations are less suited for the autoregressive nature of this task, from a prediction error perspective, and sometimes "forgetting" notes being played resulting in higher durations.

We did not explore music transcription, for which we can assume that implicit note durations (note onset and offset) might be better suited. When training with chunks of log-scaled mel-spectrograms as done by [42, 43], these may contain frequencies of unended or not begun notes. Specifying their original durations might approximate onsets might alter model performances.

Future research will further explore the other dimensions of music tokenization, such as multitrack or metadata, on transcription and other tasks analogous to natural language understanding.

8. REFERENCES

[1] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, *Deep Learning Techniques for Music Generation*, ser. Computational Synthesis and Creative Systems. Springer International Publishing, 2020. [Online]. Available: https://www.springer.com/gp/book/9783319701622

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[3] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rygGQyrFvH

[4] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Comput. Appl.*, vol. 32, no. 9, p. 4773–4784, 5 2020. [Online]. Available: https://doi.org/10.1007/s00521-018-3849-7

[5] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: a steerable model for Bach chorales generation," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 8 2017, pp. 1362–1371. [Online]. Available: https://proceedings.mlr.press/v70/hadjeres17a.html

[6] B. L. Sturm, J. F. Santos, and I. Korshunova, "Folk music style modelling by recurrent neural networks with long short-term memory units," in *Extended abstracts for the Late-Breaking Demo Session of the 16th International Society for Music Information Retrieval Conference*, 2015. [Online]. Available: https://ismir2015.ismir.net/LBD/LBD13.pdf

[7] 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, p. 955–967, 2018. [Online]. Available: https://link.springer.com/article/10.1007/s00521-018-3758-9

[8] 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*, ser. MM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1180–1188. [Online]. Available: https://doi.org/10.1145/3394171.3413671

[9] 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," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, pp. 178–186, 5 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16091

[10] 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*. Online: Association for Computational Linguistics, 8 2021, pp. 791–800. [Online]. Available: https://aclanthology.org/2021.findings-acl.70

[11] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "Popmag: Pop music accompaniment generation," in *Proceedings of the 28th ACM International Conference on Multimedia*. Association for Computing Machinery, 2020, p. 1198–1206. [Online]. Available: https://doi.org/10.1145/3394171.3413721

[12] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. J. McAuley, "Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*, 2019, pp. 685–692. [Online]. Available: http://archives.ismir.net/ismir2019/paper/000083.pdf

[13] C. Payne, "Musenet," 2019. [Online]. Available: https://openai.com/blog/musenet

[14] 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 23rd International Society for Music Information Retrieval Conference*. Bengalore, India: ISMIR, Dec. 2022. [Online]. Available: https://arxiv.org/abs/2205.05448

[15] N. Fradet, J.-P. Briot, F. Chhel, A. E. F. Seghrouchni, and N. Gutowski, "Byte Pair Encoding for symbolic music," 2023. [Online]. Available: https://arxiv.org/abs/2301.11975

[16] M. Kermarec, L. Bigo, and M. Keller, "Improving tokenization expressiveness with pitch intervals," in *Extended Abstracts for the Late-Breaking Demo Session of the 23nd International Society for Music Information Retrieval Conference*, 2022. [Online]. Available: https://ismir2022program.ismir.net/lbd_369.html

[17] G. Hadjeres and L. Crestel, "The piano inpainting application," 2021. [Online]. Available: https://arxiv.org/abs/2107.05944

[18] D. von Rütte, L. Biggio, Y. Kilcher, and T. Hofmann, "FIGARO: Controllable music generation using learned and expert features," in

The Eleventh International Conference on Learning Representations, 2023. [Online]. Available: https://openreview.net/forum?id=NyR8OZFHw6i

[19] J. Ens and P. Pasquier, "Mmm : Exploring conditional multi-track music generation with the transformer," 2020. [Online]. Available: https://arxiv.org/abs/2008.06048

[20] H.-W. Dong, K. Chen, S. Dubnov, J. McAuley, and T. Berg-Kirkpatrick, "Multitrack music transformer," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[21] O. Khattab and M. Zaharia, "Colbert: Efficient and effective passage search via contextualized late interaction over bert," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 39–48. [Online]. Available: https://doi.org/10.1145/3397271.3401075

[22] J. Ching and y.-h. Yang, "Learning to generate piano music with sustain pedals," in *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*, 2021. [Online]. Available: https://archives.ismir.net/ismir2021/latebreaking/0000017.pdf

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[24] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6

[25] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=r1gs9JgRZ

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[27] N. Fradet, J.-P. Briot, F. Chhel, A. El Fallah Seghrouchni, and N. Gutowski, "MidiTok: A python package for MIDI file tokenization," in *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*, 2021. [Online]. Available: https://github.com/Natooz/MidiTok

[28] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proceedings of 21st International Conference on Music Information Retrieval, ISMIR*, 2020. [Online]. Available: https://arxiv.org/abs/2008.07142

[29] Q. Kong, K. Choi, and Y. Wang, "Large-scale midi-based composer classification," 2020. [Online]. Available: https://arxiv.org/abs/2010.14805

[30] Q. Kong, B. Li, J. Chen, and Y. Wang, "Giantmidi-piano: A large-scale midi dataset for classical piano music," in *Transactions of the International Society for Music Information Retrieval*, vol. 5, 2021, pp. 87–98. [Online]. Available: https://transactions.ismir.net/articles/10.5334/tismir.80/#

[31] H. Hung, J. Ching, S. Doh, N. Kim, J. Nam, and Y. Yang, "EMOPIA: A multi-modal pop piano dataset for emotion recognition and emotion-based music generation," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, J. H. Lee, A. Lerch, Z. Duan, J. Nam, P. Rao, P. van Kranenburg, and A. Srinivasamurthy, Eds., 2021, pp. 318–325. [Online]. Available: https://archives.ismir.net/ismir2021/paper/000039.pdf

[32] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[33] T. Gao, X. Yao, and D. Chen, "SimCSE: Simple contrastive learning of sentence embeddings," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6894–6910. [Online]. Available: https://aclanthology.org/2021.emnlp-main.552

[34] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 9929–9939. [Online]. Available: https://proceedings.mlr.press/v119/wang20k.html

[35] D. Biś, M. Podkorytov, and X. Liu, "Too much in common: Shifting of embeddings in transformer language models and its implications," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, Jun. 2021, pp. 5117–5130. [Online]. Available: https://aclanthology.org/2021.naacl-main.403

[36] Y. Liang, R. Cao, J. Zheng, J. Ren, and L. Gao, "Learning to remove: Towards isotropic pre-trained bert embedding," in *Artificial Neural Networks and Machine Learning – ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part V*. Berlin, Heidelberg: Springer-Verlag, 2021, p. 448–459. [Online]. Available: https://doi.org/10.1007/978-3-030-86383-8_36

[37] J. Bac, E. M. Mirkes, A. N. Gorban, I. Tyukin, and A. Zinovyev, "Scikit-dimension: A python package for intrinsic dimension estimation," *Entropy*, vol. 23, no. 10, 2021. [Online]. Available: https://www.mdpi.com/1099-4300/23/10/1368

[38] K. Fukunaga and D. Olsen, "An algorithm for finding intrinsic dimensionality of data," *IEEE Transactions on Computers*, vol. C-20, no. 2, pp. 176–183, 1971. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1671801

[39] L. Amsaleg, O. Chelly, T. Furon, S. Girard, M. E. Houle, K.-I. Kawarabayashi, and M. Nett, "Extreme-value-theoretic estimation of local intrinsic dimensionality," *Data Mining and Knowledge Discovery*, vol. 32, no. 6, pp. 1768–1805, 11 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01864580

[40] E. Facco, M. d'Errico, A. Rodriguez, and A. Laio, "Estimating the intrinsic dimension of datasets by a minimal neighborhood information," *Scientific Reports*, vol. 7, no. 1, p. 12140, 9 2017. [Online]. Available: https://doi.org/10.1038/s41598-017-11873-y

[41] L. Albergante, J. Bac, and A. Zinovyev, "Estimating the effective dimension of large biological datasets using fisher separability analysis," in *International Joint Conference on Neural Networks (IJCNN)*, 7 2019, pp. 1–8. [Online]. Available: https://arxiv.org/abs/1901.06328

[42] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. H. Engel, "Sequence-to-sequence piano transcription with transformers," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, 2021, pp. 246–253. [Online]. Available: https://archives.ismir.net/ismir2021/paper/000030.pdf

[43] J. P. Gardner, I. Simon, E. Manilow, C. Hawthorne, and J. Engel, "MT3: Multi-task multitrack music transcription," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=iMSjopcOn0p