Byte Pair Encoding for Symbolic Music

Nathan Fradet [1 2] **Jean-Pierre Briot** [1] **Fabien Chhel** [3] **Amal El Fallah Seghrouchni** [1] **Nicolas Gutowski** [4]

Abstract

The symbolic music modality is nowadays mostly represented as discrete and used with sequential models such as Transformers, for deep learning tasks. Recent research put efforts on the tokenization, i.e. the conversion of data into sequences of integers intelligible to such models. This can be achieved by many ways as music can be composed of simultaneous tracks, of simultaneous notes with several attributes. Until now, the proposed tokenizations are based on small vocabularies describing the note attributes and time events, resulting in fairly long token sequences. In this paper, we show how Byte Pair Encoding (BPE) can improve the results of deep learning models while improving its performances. We experiment on music generation and composer classification, and study the impact of BPE on how models learn the embeddings, and show that it can help to increase their isotropy, i.e., the uniformity of the variance of their positions in the space.

1. Introduction

Deep learning tasks on symbolic music are nowadays mostly tackled by sequential models[1], such as the Transformers (Vaswani et al., 2017). These models receive sequences of tokens as input, and convert them to learned embedding vectors. A token is an integer associated to a high level element, such as a word or sub-word in natural language, and both are linked in a vocabulary that acts as a look-up table. An embedding represents the semantic information of a token as a vector of fixed-size, and is learning contextually by the model. To use such models for symbolic music, one needs to *tokenize* the data, i.e., convert it to sequences of tokens that can be decoded back. This can be achieved by several ways, as music can be composed of simultaneous tracks, of simultaneous notes with several attributes such as their pitch and duration.

Recently, the token representation of symbolic music has been extensively studied, with the goal to improve 1) the results, e.g. the quality of generated results or the accuracy of a certain Music Information Retrieval (MIR) task, and; 2) the efficiency of the models. The former is tackled with more expressive representations (Huang & Yang, 2020; Kermarec et al., 2022), and the latter by representations based on either token combinations (Payne, 2019; Donahue et al., 2019), or embedding pooling (Hsiao et al., 2021; Zeng et al., 2021; Ren et al., 2020), which reduce the overall sequence length. Still, current tokenizations only use tokens representing the values of time and note attributes, such as *Pitch* or *Duration*. This comes with a big limitation: these tokens do not carry much information by themselves, and neither their associated embeddings. By analogy to natural language, these tokens are closer to the characters than words. Yet, the expressive information carried by music is deduced by the combinations of its notes and their attributes. Considering the infinite possible arrangements, deep learning models may struggle to implicitly learn their common features.

In this paper, we study the application of Byte Pair Encoding (BPE, described in Section 3) for symbolic music generation, aiming to improve the two objectives mentioned above, while making the models learn more isotropic embedding representations in some cases. To the best of our knowledge, BPE has yet not been studied for the symbolic music modality, although it can be applied on top of any music tokenization that do not perform embedding pooling. This work aims at closing this gap by shedding light on the results and performance gains of using BPE:

- We experiment on two public datasets (Wang et al., 2020b; Kong et al., 2021), with two base tokenizations, on which BPE is learned with several vocabulary sizes, on the generation and composer classification tasks, and show that it improves the results;
- We compare BPE with other sequence reduction techniques introduced in recent research;
- We study the geometry of the learned embeddings, and show that BPE can improve their isotropy;
- We show some limits of BPE, such as on the proportion

[1]LIP6, Sorbonne University - CNRS, Paris, France [2]Aubay, Boulogne-Billancourt, France [3] ESEO-TECH / ERIS, Angers, France [4]University of Angers, Angers, France. Correspondence to: Nathan Fradet <nathan.fradet@lip6.fr>.

[1]Commonly referred as Language Models (LM)

of sampled tokens, and that the vocabulary size has to be carefully chosen.

The source code is provided for reproducibility: `https://github.com/Natooz/BPE-Symbolic-Music`

The paper is organised as follows: Section 2 reviews the related work while Section 3 sheds light on the BPE technique. Section 4 describes our experimental settings and Section 5 describes the evaluation metrics that we use for the experimental evaluation. Section 6 presents the results and analysis. Furthermore, Section 7 provides an additional study on the impact of BPE on how the models learn the embeddings. Finally, Section 8 presents our conclusion and perspectives.

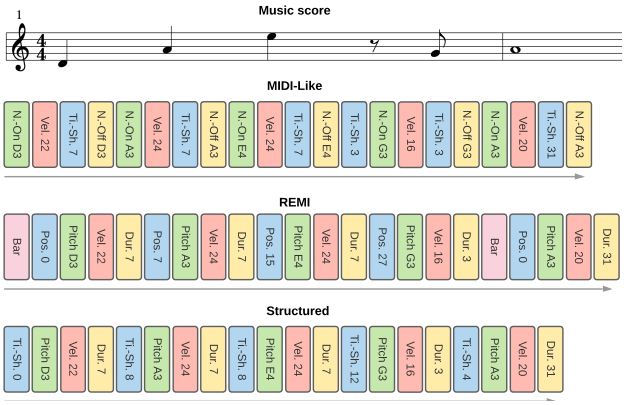


Figure 1. A sheet music and several token representations.

2. Related work

In this section we start by reminding research of specific music representation of symbolic music generation. Then, we present how recent works put efforts on different strategies to reduce the sequence length. Finally, we explain their limitations which conduce us to propose our novel approach that is to apply Byte Pair Encoding in the field of symbolic music for reducing sequence length.

2.1. Representation of symbolic music

Most works on symbolic music generation from deep learning use a specific music representation. Early research introduced representations specifically tied to the training data being used, such as DeepBach (Hadjeres et al., 2017), FolkRNN (Sturm et al., 2015) or BachBot (Liang et al., 2017). Non-sequential models such as MuseGAN (Dong et al., 2018) often represent music as pianoroll matrices.

Since, more universal representations have been studied, allowing to convert any sequence of (simultaneous) notes into tokens (Oore et al., 2018; Huang & Yang, 2020; Hadjeres & Crestel, 2021; Fradet et al., 2021). Some of them are depicted in Figure 1.

2.2. Sequence reduction strategies

In more recent works, efforts have been put towards the efficiency. Indeed, most recent models are based on the Transformer architecture (Vaswani et al., 2017). The attention mechanism, at the heart of Transformers, has however a time and space complexity that grows quadratically with the input sequence length. This is a well known bottleneck, that led researchers to work on more efficient attention estimations (Tay et al., 2021), down to linear complexity. In the field of symbolic music specifically, researchers worked on strategies to reduce the sequence length in order to increase 1) the efficiency of the models; 2) the scope of the attention mechanism; 3) the quality of the generated results. These strategies can be split in two categories: 1) embedding pooling strategies such as *Compound Word* (Hsiao et al., 2021) (*CPWord*), *Octuple* (Zeng et al., 2021) or PopMag (Ren et al., 2020); 2) token combination strategies such as in MuseNet (Payne, 2019) or LakhNES (Donahue et al., 2019). Embedding pooling consists in merging the embeddings of several distinct tokens with a pooling operation. This is often done by concatenating the embeddings and projecting the sequence, resulting in an aggregated embedding of fixed size. Token combinations is simply the use of a vocabulary containing tokens that represent several values, e.g., `Pitch-x_Duration-y` that represent both the pitch and velocity information.

2.3. Limitations

However, these strategies show the following limitations. Embedding pooling: 1) requires a more complex training procedure; 2) for generation, inferring from such model can be seen as sampling from a multivariate distribution, which can be a delicate operation; 3) the results can easily degenerate if the pooling does not yield semantically rich embeddings that represent the underlying tokens. On the other hand, token combinations of entire types of tokens can lead to large vocabularies with unused tokens and potentially non-optimized or unbalanced token distributions.

To the best of our knowledge, no work has been conducted on applying BPE, introduced in Section 3, to symbolic music generation. A similar technique is used with SymphonyNet (Liu et al., 2022), which does not rely on token adjacency but rather on the concurrence of multiple notes, and they only experimented with a vocabulary size of 1k tokens.

The following section describes the Byte Pair Encoding technique, its algorithm and depicts how it can be relevant to use in the field of symbolic music.

3. Byte Pair Encoding

Byte Pair Encoding (BPE) (Gage, 1994) is a data compression technique. It converts the most recurrent successive bytes (or in our case tokens) in a corpus into newly created ones. For instance, in the character sequence `aabaabaacaa`, the sub-sequence `aa` occurs three times and is the most recurrent. Learning and applying BPE on this sequence would replace `aa` with a new symbol, e.g., `d`, resulting in a reduced sequence `dbdbdcd`. The latter can be reduced again by replacing the `db` subsequence, giving `eedcd`. In the context of deep learning, BPE naturally increases the size of the vocabulary, while reducing the overall sequence lengths. In practice BPE is learned on a corpus until the vocabulary reaches a target size. BPE learning is described by the pseudo-code of Algorithm 1.

Algorithm 1 Learning of BPE pseudo-code

Require: Base vocabulary $\mathcal{V}$, target vocabulary size N, dataset $\mathcal{X}$
while $|\mathcal{V}| < N$ **do**
  Find $s = \{t_1, t_2\} \in \mathcal{V}^2$, from $\mathcal{X}$, the most recurrent token succession
  Add a new token t in $\mathcal{V}$, mapping to s
  Substitute every occurrence of s in $\mathcal{X}$ with t
end while
return $\mathcal{V}$

BPE is nowadays largely used in the NLP field as it allows to encode rare words and segmenting unknown or composed words as sequences of sub-word units (Sennrich et al., 2016).

In symbolic music, notes are represented by successions of tokens that represent the values of their attributes. In this context, BPE can allow to represent a note, or even a succession of notes, that is very recurrent in the dataset, as a single token. For instance, a note that would be coded as the succession of tokens `Pitch_D3`, `Velocity_60`, `Duration_2.0` could be replaced by a single new one. Rare note (and attributes) would still be encoded as non-BPE tokens. The same logic applies to time tokens, that can also be associated to note tokens.

4. Experimental settings

This section details the experimental protocol by describing the models, the training and the datasets used along with the specific tokenization processes.

4.1. Model and training

As we specifically focus on sequential models, we experiment with the state of the art deep learning architecture for most NLP tasks at the time of writing, the Transformer (Vaswani et al., 2017) architecture. The generator uses a causal attention mask and is trained with teacher forcing, while the classifier does not use attention mask and is first pre-trained to retrieve randomized tokens then finetuned to classify the input sequences. They are respectively similar to GPT2 (Radford et al., 2019) and BERT (Devlin et al., 2019). The details of implementation, such as their sizes and training, can be found in Appendix A

All models receive sequences between 384 and 460 tokens, beginning with special `BOS` (Beginning of Sequence) and ending `EOS` (End of Sequence) tokens. We split datasets in two subsets: one only used for training and updating the models, one for validation to monitor trainings, that is also used to test the models after training. These subsets represent respectively 65% and 35% of the original datasets.

4.2. Datasets

We experiment with two datasets: POP909 (Wang et al., 2020b) and GiantMIDI (Kong et al., 2021).

The POP909 dataset (Wang et al., 2020b) is composed of 909 piano tracks of Pop musics, with aligned MIDI and audio versions. Each MIDI file contains three tracks: the first is the lead melody, the second is secondary melodies and bridges, the third is the arrangements with chords and arpeggios. For our experiments we merge all three tracks into a single one.

The GiantMIDI dataset (Kong et al., 2021) is composed of 10k piano MIDI files, transcribed from audio to MIDI without downbeat and tempo estimation. Each file contains a single track of non-interrupted piano music, often with complex melodies and harmonies. Considering the complexity of its content, we make the assumption that it is a difficult dataset for a model to learn from.

We perform data augmentation on the pitch dimension on both datasets. Each MIDI file is augmented up and down to two octaves.

4.3. Music tokenization

We experiment with *Remi* (Huang & Yang, 2020) and *TSD* (for Time Shift Duration) as base tokenizations, on which BPE will be applied on top. Both tokenizations describe notes as a succession of the `Pitch`, `Velocity` and `Duration` tokens. *Remi* represents time with `Bar` and `Position` tokens, which respectively indicates when a new bar is beginning and at which position within the time is. *TSD* represents time with `TimeShift` tokens, indicating explicitly time movements.

When tokenizing symbolic music, it is common to downsample continuous features to discrete sets of values. For instance, velocities can be downsampled from 128 to 32

values. These sets should be sufficiently precise so that the global information remains coherent (Huang & Yang, 2020; Oore et al., 2018; Hadjeres & Crestel, 2021). Downsampling features helps models to learn more easily, as it allows to reduce the perplexity of the predictions, especially for values which are less commons in the training set. The details of our downsamplings can be found in Appendix B.

BPE is learned from tokenized corpuses, up to a maximum of 1500 randomly picked files, to reduce the learning time. We choose to experiment with six vocabulary sizes. One without BPE, and five where the original vocabulary size is multiplied by 4, 10, 20, 50 and 100.

To extend our analysis, we also experiment with a version of *TSD* and *Remi* where `Pitch` and `Velocity` tokens are merged (*PVm*), and one where `Pitch`, `Velocity` and `Duration` are merged (*PVDm*). *PVm* is similar to the strategy used with MuseNet (Payne, 2019). We finally experiment with the *CPWord* (Hsiao et al., 2021) and *Octuple* (Zeng et al., 2021) embedding pooling strategies, that we group with *Remi* in our experiments as they represent time similarly. We use the same pooling strategy, and sample independently from the logits of each output modules. For implementation simplicity reasons, all embeddings have the same size than the model dimension.

5. Evaluation metrics

Generative models are often evaluated with automatic metrics on the generated results. Image and audio models are assessed with the Fréchet Inception Distance (FID) (Heusel et al., 2017) and Fréchet Audio Distance (FAD) (Kilgour et al., 2019), both comparing the distribution of original data and generated results. Language models are often assessed with BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) or other metrics that compare generated results with reference sentences.

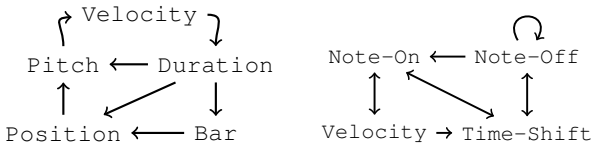
Automatic evaluation of symbolic music remains however an open issue. It exists no reference-free metric measuring its quality or fidelity. Metrics with reference such as BLEU may be suited for machine translation tasks, but remains irrelevant for open-ended generation, such as in our case.

We then perform both human and automatic evaluations, as commonly done for symbolic music (Huang & Yang, 2020; Huang et al., 2018; Hsiao et al., 2021). Our automatic metrics aim to measure the errors of prediction of the models, and the similarity of some features.

5.1. Tokenization syntax error

Every tokenization has an underlying syntax of token type and value successions, that can normally be made. For instance, if the last token of an input sequence is of type `Pitch`, a tokenization could require that the next token to predict must be of type `Velocity`. We could also expect a model to not predict more than once the same note at a same moment, or to not go back in time.

Successions of incorrect token types can be interpreted as errors of prediction. These errors can help us to measure if a model has efficiently learned the music representation and if it can yield coherent results. With this motivation, we introduce a new metric we called Tokenization Syntax Errors (TSE).



(a) REMI. (b) MIDI-Like

Figure 2. Directed graphs of the token types succession (without additional tokens) for a) REMI (Huang & Yang, 2020) and b) MIDI-Like (Oore et al., 2018).

We distinguish five categories of errors:

- $\mathbf{TSE_{type}}$: the predicted token does not have a type that should follow the previous one. For any tokenization, we can draw a directed graph representing the possible token types successions, such as in Figure 2.
- $\mathbf{TSE_{time}}$: when using `Position` tokens, the predicted `Position` value is inferior or equal to the current one, making the time goes backward.
- $\mathbf{TSE_{dupn}}$ (duplicated note): when the model predicts a note that has already been played at the current moment (by the same instrument).
- $\mathbf{TSE_{nnof}}$ (no NoteOff): when using `NoteOn` and `NoteOff`, and that a `NoteOn` token has been predicted with no `NoteOff` later to end it, or too distant in time.
- $\mathbf{TSE_{nnon}}$ (no NoteOn): when a `NoteOff` token is predicted but the corresponding note has not been played.

For a given sequence of tokens, TSE measures the ratio, scaled between 0 and 1, of errors for these five categories. A TSE of 0 means that there is no error in the sequence, while a ratio of 1 means only errors were predicted. Our experiments are not concerned by the last two categories as we do not use `NoteOff` tokens.

Finally, we should mention that most of these errors can be avoided by a ruled-based sampling. When predicting a token, one can easily keep track of the time, notes played and token types to automatically exclude invalid predictions.

In practice, this can be achieved by setting the invalid indices of the predicted logits to $-\infty$ before applying softmax.

5.2. Feature similarity

We expect models to generate continuations that keep the features of the input prompt consistent. For instance, it should predict first notes within the same scale and with the same velocity range. We measure this similarity by calculating the overlapping area of distributions of features, for the prompt and the first 16 generated beats.

Previous works (Yang & Lerch, 2020; Choi et al., 2020; Mittal et al., 2021; von Rütte et al., 2022) use the probability density function of the distributions, estimated with kernel density estimations, and emphasizes that it smooths and transforms the distributions into more general representations. While this method can be suited for continuous modalities, it can lead to inaccuracies with categorical ones. Here, pitch and duration features can be considered as discrete. Their distributions are both sparse, containing for instance many white keys and fewer black keys, yet adjacent and corresponding to close integer values in the MIDI format. In order to be more accurate, we measure this similarity with the histogram intersection of these features, as described in Equation (1).

$$\begin{aligned} \text{Similarity}\left(\mathcal{D}_1, \mathcal{D}_2\right) &= \text{HI}\left(\text{Hist}\left(\mathcal{D}_1\right), \text{Hist}\left(\mathcal{D}_2\right)\right) \\ \text{HI}(\mathbf{x}, \mathbf{y}) &= \textstyle\sum_i \min(x_i, y_i),\ x_i \geq 0,\ y_i \geq 0 \end{aligned} \tag{1}$$

$\text{Hist} : \mathbb{R}^{|\mathcal{D}|} \mapsto \mathbb{N}^e$ returns the normalized histogram of a distribution of a feature with e elements, HI stands for Histogram Intersection.

5.3. Human evaluations

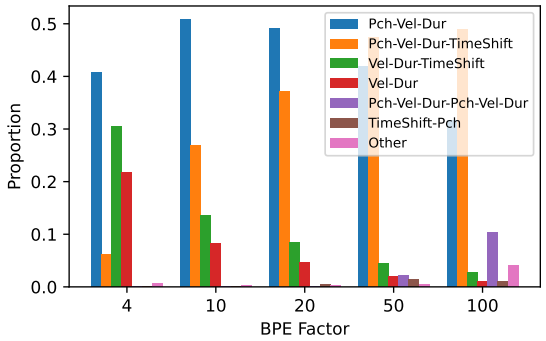
For each experiment, we select 40 prompts of 8 beats. For each prompts, we generate continuations of 1k tokens with the benchmarked models. Three musicians open the continuations as a MIDI file, allowing them to listen the tracks and also visualize them as piano rolls. Among the tracks, they are asked to select the one: 1) with the highest fidelity on pitch scale, velocity, note density and rhythm, with the prompt; 2) they subjectively prefer overall, considering its correctness, structure and richness.

6. Results and analysis

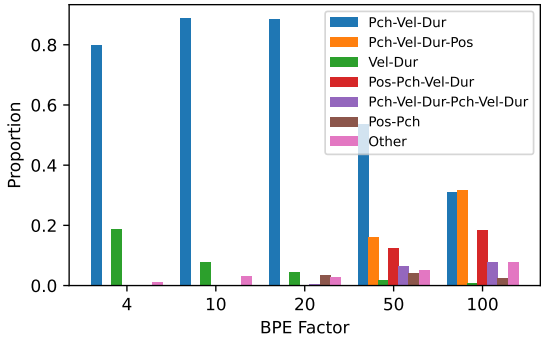
We focus on how BPE is learned on the corpuses, then on its benefits for music generation and composer classification.

6.1. BPE learning

Figure 3 shows the distribution of token types combinations of the learned BPE tokens. We observe that the majority of the combinations learned on the *Remi* tokenization represent notes, by their `Pitch`, `Velocity` and



(a) TSD



(b) Remi

Figure 3. Normalized distributions of the token types of the BPE tokens, per BPE factor for the POP909 dataset.

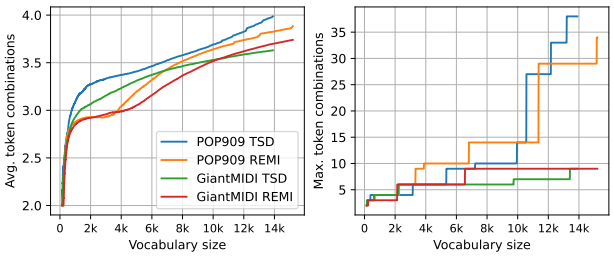


Figure 4. Average (left) and maximum (right) number of token combinations represented by BPE tokens in function of the vocabulary size.

`Duration` attributes. For *TSD*, the combinations also include `TimeShift` tokens early in the learning. This difference mostly comes from common `TimeShift` tokens following notes, whereas for *Remi* the notes are distributed at different `Position`(s). As the vocabulary grows, the combinations tend to be more diverse. The distribution for the GiantMIDI dataset are showned in Appendix C.

Figure 4 shows the evolution of the average number of non-BPE token combinations represented by the BPE tokens. At the beginning of the learning, the mean number of combinations grows more quickly as the most recurrent token successions are often made of more than two tokens. The POP909 dataset being smaller than GiantMIDI, it naturally leads to a higher maximum number of combinations as the latter is more diverse. When the vocabulary begins to con-

Table 1. Metrics of generated results. TSE numbers are all scaled at e-3 for better readability. Sim stands for similarity, the best results are the closest to the datasets. Hum. Fidelity and Overall are the human evaluations.

Data / Strategy	TSE_{type} (↓)	TSE_{dupn} (↓)	TSE_{time} (↓)	Sim. pit.	Sim. vel.	Sim. dur.	Hum. Fidelity (↑)	Hum. Overall (↑)
POP909 TSD				0.66 ± 0.13	0.84 ± 0.12	0.69 ± 0.14		
No BPE	1.0 ± 1.8	13.6 ± 8.0	-	0.59 ± 0.08	**0.82 ± 0.10**	0.64 ± 0.09	0.00	0.00
BPE×4	**0.2 ± 0.9**	21.9 ± 19.9	-	**0.65 ± 0.07**	**0.82 ± 0.10**	0.74 ± 0.08	0.24	0.19
BPE×10	0.5 ± 2.2	13.4 ± 14.6	-	**0.64 ± 0.07**	0.78 ± 0.12	0.74 ± 0.07	**0.53**	**0.42**
BPE×20	0.8 ± 2.1	12.8 ± 11.0	-	0.62 ± 0.07	0.79 ± 0.11	**0.70 ± 0.09**	0.20	0.31
BPE×50	22.4 ± 24.0	4.4 ± 5.3	-	0.56 ± 0.07	0.70 ± 0.12	0.62 ± 0.11	0.02	0.02
BPE×100	21.5 ± 40.2	35.6 ± 56.0	-	0.54 ± 0.08	0.66 ± 0.14	0.63 ± 0.10	0.00	0.00
PVm	6.1 ± 6.6	6.9 ± 9.3	-	0.59 ± 0.08	0.78 ± 0.12	0.73 ± 0.08	0.01	0.06
PVDm	23.6 ± 19.3	**0.2 ± 0.7**	-	0.43 ± 0.09	0.57 ± 0.19	0.54 ± 0.12	0.00	0.00
POP909 REMI				0.66 ± 0.13	0.84 ± 0.12	0.69 ± 0.14		
No BPE	**0.0 ± 0.1**	115.4 ± 33.8	**74.7 ± 26.7**	**0.61 ± 0.08**	**0.85 ± 0.09**	0.72 ± 0.07	0.02	0.03
BPE×4	0.1 ± 0.4	65.7 ± 21.7	154.9 ± 27.6	0.55 ± 0.09	0.77 ± 0.12	0.70 ± 0.09	0.27	0.34
BPE×10	0.3 ± 1.1	**52.3 ± 18.3**	167.1 ± 30.5	0.49 ± 0.08	0.77 ± 0.10	0.63 ± 0.09	**0.52**	**0.44**
BPE×20	0.8 ± 2.2	81.8 ± 37.3	242.6 ± 46.5	0.46 ± 0.08	0.71 ± 0.13	0.61 ± 0.10	0.12	0.12
BPE×50	37.8 ± 35.5	128.2 ± 22.2	324.1 ± 21.5	0.30 ± 0.12	0.56 ± 0.20	0.55 ± 0.12	0.00	0.00
BPE×100	83.9 ± 78.0	136.3 ± 32.4	324.6 ± 28.8	0.28 ± 0.11	0.54 ± 0.22	0.55 ± 0.12	0.00	0.00
PVm	2.3 ± 7.1	160.0 ± 75.3	102.7 ± 48.2	0.60 ± 0.08	0.77 ± 0.12	**0.69 ± 0.09**	0.05	0.04
PVDm	49.3 ± 46.2	99.8 ± 25.1	301.9 ± 26.5	0.32 ± 0.13	0.50 ± 0.24	0.45 ± 0.12	0.02	0.02
CPWord	331.9 ± 33.8	144.5 ± 46.8	99.3 ± 16.6	0.57 ± 0.08	**0.85 ± 0.07**	0.73 ± 0.09	0.00	0.00
Octuple	-	789.3 ± 111.1	891.9 ± 76.1	0.05 ± 0.15	0.07 ± 0.21	0.06 ± 0.17	0.00	0.00
GiantMIDI TSD				0.49 ± 0.17	0.74 ± 0.18	0.52 ± 0.23		
No BPE	0.2 ± 1.1	3.9 ± 4.6	-	**0.50 ± 0.10**	0.77 ± 0.12	0.63 ± 0.13	0.24	0.19
BPE×4	0.5 ± 1.4	15.2 ± 18.1	-	0.51 ± 0.10	**0.75 ± 0.13**	0.62 ± 0.14	**0.33**	0.27
BPE×10	1.5 ± 3.3	35.2 ± 45.6	-	0.51 ± 0.11	0.68 ± 0.17	0.65 ± 0.13	0.29	**0.37**
BPE×20	**0.0 ± 0.0**	17.5 ± 29.3	-	0.52 ± 0.09	0.73 ± 0.15	0.65 ± 0.12	0.11	0.08
BPE×50	**0.0 ± 0.3**	6.8 ± 8.5	-	**0.50 ± 0.09**	0.70 ± 0.13	0.64 ± 0.11	0.00	0.00
BPE×100	1.5 ± 3.7	1.1 ± 1.5	-	0.46 ± 0.09	0.63 ± 0.17	0.53 ± 0.13	0.01	0.01
PVm	3.0 ± 3.7	0.7 ± 1.3	-	0.46 ± 0.11	0.69 ± 0.15	0.67 ± 0.11	0.02	0.09
PVDm	35.6 ± 56.1	**0.5 ± 1.2**	-	0.39 ± 0.13	0.61 ± 0.18	0.25 ± 0.18	0.00	0.00
GiantMIDI REMI				0.49 ± 0.17	0.74 ± 0.18	0.52 ± 0.23		
No BPE	**0.2 ± 0.9**	57.8 ± 40.2	95.1 ± 42.8	0.53 ± 0.10	**0.75 ± 0.14**	0.63 ± 0.13	0.00	0.01
BPE×4	**0.2 ± 0.8**	44.3 ± 23.5	**82.3 ± 36.4**	0.46 ± 0.11	0.71 ± 0.15	0.62 ± 0.12	0.41	0.43
BPE×10	2.5 ± 3.5	**31.7 ± 20.2**	175.6 ± 60.3	0.43 ± 0.10	0.63 ± 0.21	**0.54 ± 0.15**	**0.53**	**0.52**
BPE×20	0.7 ± 2.4	36.6 ± 29.3	221.9 ± 66.4	0.33 ± 0.12	0.65 ± 0.16	0.46 ± 0.15	0.02	0.01
BPE×50	34.8 ± 11.1	80.5 ± 53.1	316.4 ± 54.1	0.36 ± 0.11	0.58 ± 0.18	0.30 ± 0.23	0.00	0.00
BPE×100	476.1 ± 148.3	159.8 ± 60.1	285.3 ± 31.5	0.19 ± 0.10	0.59 ± 0.20	0.20 ± 0.19	0.00	0.00
PVm	0.7 ± 2.4	53.8 ± 47.4	181.5 ± 56.9	0.46 ± 0.11	0.70 ± 0.15	0.60 ± 0.14	0.00	0.01
PVDm	31.9 ± 63.9	65.6 ± 28.8	285.6 ± 32.6	0.33 ± 0.14	0.58 ± 0.19	0.29 ± 0.17	0.02	0.02
CPWord	408.9 ± 28.3	160.1 ± 54.4	69.3 ± 16.7	**0.51 ± 0.11**	0.81 ± 0.09	0.69 ± 0.12	0.00	0.00
Octuple	-	763.8 ± 134.4	894.3 ± 62.1	0.03 ± 0.11	0.06 ± 0.19	0.04 ± 0.15	0.00	0.00

tain BPE tokens with a large number of combinations, it starts to specialize on very specific note successions that may appear in few data samples. In particular, big jumps of maximum number of combinations, e.g. from 14 to 27 for POP909 *Remi*, indicate that two already big BPE tokens represent the most recurrent succession. These numbers, correlated with the model, dataset sizes and overall token distribution of the dataset, might help to choose an optimal vocabulary size.

Further analysis in Appendix C shows that BPE considerably reduces the sequence length, and so the training and generation time, at the cost of an increased tokenization time. Tokenization of data is however often performed once, and the training time gain is very likely to be larger than the tokenization time loss.

6.2. Generated results

For the generation task, we generate continuations of input prompt from the validation subset. The continuations are autoregressively generated with 1024 steps, with nucleus sampling (Holtzman et al., 2020), with $p = 0.9$.

The results of all metrics are reported in Table 1. For *TSD*, BPE allows to reduce both the token type and note duplication errors in most cases, while the time errors slightly increase for *Remi* baselines. These results show that models can easily scale to bigger vocabularies, up to a certain limit. Here, starting from a BPE factor of 50, the TSE seems to increase, as do the other results. BPE tends to however produce results with features slightly less similar, especially with big vocabulary sizes.

We gathered a total of 400 human evaluations. They show that BPE with factors of 4 and 10 significantly outperform other baselines, in all experiments. BPE helps models to

Table 2. Number of tokens sampled and not sampled by generative models, respectively right and left separated by |.

Strategy	POP909 TSD	POP909 Remi	GiantMIDI TSD	GiantMIDI Remi
No BPE	116 | 23 (16%)	141 | 11 (7%)	136 | 3 (2%)	151 | 1 (0%)
BPE×4	454 | 102 (18%)	487 | 121 (19%)	456 | 100 (17%)	386 | 222 (36%)
BPE×10	479 | 911 (65%)	514 | 1006 (66%)	456 | 934 (67%)	618 | 902 (59%)
BPE×20	592 | 2188 (78%)	552 | 2488 (81%)	478 | 2302 (82%)	504 | 2536 (83%)
BPE×50	521 | 6429 (92%)	249 | 7351 (96%)	401 | 6549 (94%)	155 | 7445 (97%)
BPE×100	521 | 13379 (96%)	244 | 14956 (98%)	281 | 13619 (97%)	89 | 15111 (99%)
PVm	321 | 426 (57%)	338 | 422 (55%)	342 | 405 (54%)	369 | 391 (51%)
PVDm	391 | 13712 (97%)	144 | 13972 (98%)	252 | 13851 (98%)	166 | 13950 (98%)

Table 3. Average accuracy of classification models.

Strategy	TSD (↑)	Remi (↑)	TSD Large (↑)	Remi Large (↑)
No BPE	0.196 ± 0.031	0.169 ± 0.021	0.208 ± 0.033	0.175 ± 0.022
BPE×4	0.218 ± 0.033	0.168 ± 0.021	0.226 ± 0.034	0.171 ± 0.022
BPE×10	0.226 ± 0.038	0.190 ± 0.030	0.228 ± 0.037	0.201 ± 0.034
BPE×20	**0.236 ± 0.038**	0.195 ± 0.026	0.240 ± 0.039	0.210 ± 0.029
BPE×50	0.199 ± 0.027	0.207 ± 0.032	**0.247 ± 0.041**	0.216 ± 0.035
BPE×100	0.122 ± 0.009	0.119 ± 0.008	0.243 ± 0.037	0.126 ± 0.010
PVm	0.199 ± 0.027	0.150 ± 0.016	0.213 ± 0.029	0.188 ± 0.025
PVDm	0.226 ± 0.035	0.192 ± 0.028	0.228 ± 0.036	0.194 ± 0.029
CPWord	-	0.204 ± 0.28	-	0.214 ± 0.024
Octuple	-	**0.274 ± 0.041**	-	**0.283 ± 0.043**

generate more correct and pleasant music. We make the assumption that having a larger set of learned embeddings help the model to capture more easily the global melody, harmony and music structure, and in turn improve the generated results. These embedding, when well learned contextually, may represent richer and more explicit information.

Table 2 shows that while models give high probabilities to more unique tokens with BPE in absolute number, the proportion of sampled tokens decreases. Models tend to focus on the sets of more recurrent tokens and omitting more rare ones. Beyond a BPE factor of 20 (or vocabulary size between 2k and 2.5k tokens), the models are even focusing on a more restricting sets of tokens. These numbers highlight the limitations of using a too large vocabulary size, as the extra effort is unlikely to result in better results.

6.3. Composer classification

Composer classification is performed with the top-10 most present composers of the GiantMIDI dataset. The results, reported in Table 3, show that BPE outperforms other baselines. Here, the model seems to benefit from larger vocabulary sizes. We also remark that the model size plays in its capacity to handle large vocabularies. While the results of BPE100 for the small model indicate it was unable to learn anything, the larger one performed almost as good as the top baseline. A second observation is the good performances of embedding pooling strategies (CPWord and OCtuple). While they performed poorly for generative tasks, they are among the best for this classification task. They seem to be better for MIR tasks than generation. As stated in Section 1, generation implies sampling, and sampling from several distributions is delicate, as for training a model with an autoregressive objective on several output distributions.

Table 4. IsoScore results.

Generator	POP909 TSD	POP909 Remi	GiantMIDI TSD	GiantMIDI Remi
No BPE	0.09	**0.14**	**0.08**	**0.09**
BPE×4	0.02	0.04	0.02	0.02
BPE×10	0.12	0.11	0.02	0.07
BPE×20	**0.13**	0.05	0.02	0.02
BPE×50	0.02	0.01	0.01	0.01
BPE×100	0.01	0.01	0.01	0.00
PVm	0.02	0.02	0.01	0.02
PVDm	0.00	0.00	0.00	0.00
CPWord	-	0.04	-	0.08
Octuple	-	0.04	-	0.02
Classifier	TSD (↑)	Remi (↑)	TSD Large (↑)	Remi Large (↑)
No BPE	0.74	0.71	0.80	0.77
BPE×4	0.35	0.33	0.54	0.37
BPE×10	0.36	0.31	0.48	0.50
BPE×20	0.54	0.57	0.64	0.53
BPE×50	0.77	0.80	0.75	0.82
BPE×100	**0.82**	**0.90**	0.87	**0.89**
PVm	0.27	0.27	0.32	0.32
PVDm	0.69	0.88	**0.88**	0.88
CPWord	-	0.08	-	0.05
Octuple	-	0.08	-	0.06

7. Learned embedding spaces

Results presented in this section rely on Table 4, and Figures 5 and 6. Isotropy is a measure of the uniformity of the space occupied by a distribution, across all dimensions. In our case, the distribution is a manifold $X \in \mathbb{R}^{N \times d}$ where $N = |V|$ and d is the model/embedding dimension. It has been associated with improved performances with language models (Biś et al., 2021; Liang et al., 2021), mostly because embeddings are more discriminative and enable models to capture and distinguish more easily subtle semantic information. It has been observed that representations from Transformers often exhibit anisotropy, i.e., they tend to occupy only a small subspace of the embedding space, and often not uniformly (Gao et al., 2019; Ethayarajh, 2019; Wang et al., 2020a; Gong et al., 2018; Reif et al., 2019), especially causal generative models (Ethayarajh, 2019).

Isotropy is often estimated by different ways: singular value decomposition (Biś et al., 2021; Gao et al., 2019; Liang et al., 2021; Wang et al., 2020a), intrinsic dimension (Cai et al., 2021), partition function (Arora et al., 2016; Mu & Viswanath, 2018), average cosine similarity (Ethayarajh, 2019). Although these methods are correlated with isotropy, recent research shed light on some of their limits (Rudman et al., 2022). We choose to estimate it with intrinsic value, IsoScore (Rudman et al., 2022), singular value and cosine similarity, to have results that corroborate and complement themselves. The results of the two latter can be found in Appendix D. For tokenizations with embedding pooling, we used 50k randomly sampled embeddings of combinations of tokens representing notes, as using all the embedding combinations would be intractable and would not reflect the ones actually learned by the models. Results for tokenizations where $N \lesssim d$ (no BPE) have to be interpreted loosely. Isotropy cannot be reliably measured with less samples than the number of dimensions they occupy. The estimations are more accurate when $N \gg d$, as more samples populate all

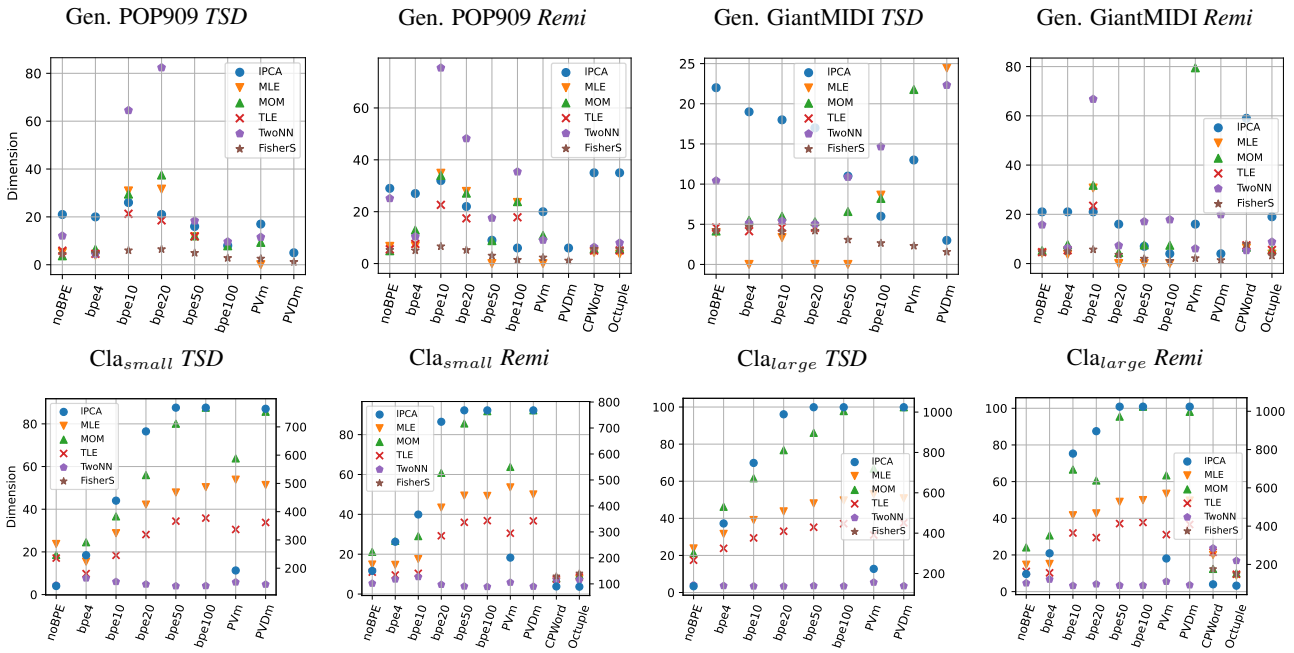


Figure 5. Intrinsic dimension estimations. A second x axis has been added on the right for *lPCA* on classifier plots for better readability.

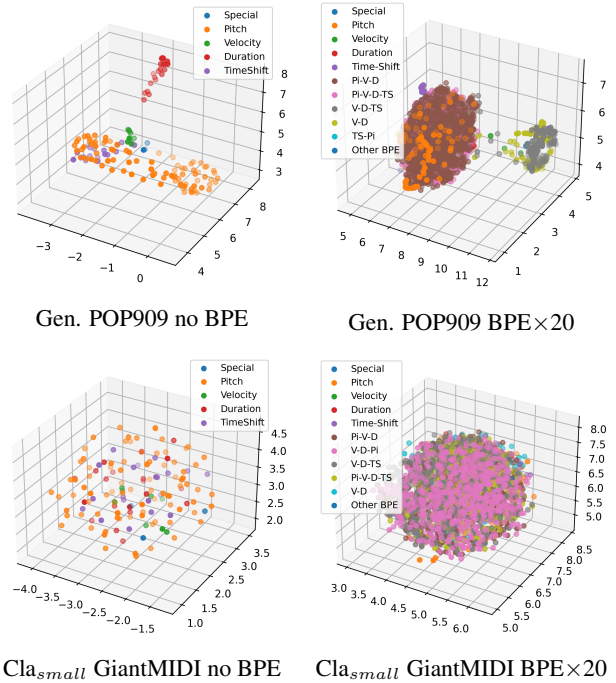


Figure 6. 3d UMAP representations of learning embedding spaces, with *TSD* tokenization. Abbreviations in legend stand for: Pi: Pitch; V: Velocity; D: Duration; Po: Position: TS: TimeShift.

dimensions of $\mathbb{R}^d$.

The IsoScore results (See Table 4), show that BPE does not increase the score for generative models. It seems that big vocabularies with BPE yield lower IsoScore results, that corroborate with the intrinsic dimension results (Figure 5). Causal generative models have been shown to learn anisotropic embedding representations (Cai et al., 2021; Ethayarajh, 2019). Embeddings form cones and clusters, that can be observed in Figure 6. As we estimated isotropy on all embeddings altogether, the presence of clusters naturally correlate with anisotropy, as the variance is mostly pronounced on their distances. The cluster themselves might be more isotropic (Cai et al., 2021).

On the other hand, BPE can help bi-directional models to learn more isotropic embedding representations. The IsoScore grows with the vocabulary size, as do the intrinsic dimension. In Figure 6 we observe that the embeddings have no preferred direction in space, forming a sphere (See more figures in Appendix D).

8. Conclusion

We showed that BPE can increase the quality of results for symbolic music generation, and composer classification, while improving the performances, with a well chosen vocabulary size. BPE can be applied on top of any tokenization, and we advice the reader to do so for projects involving symbolic music. The drawbacks are a time-consuming vocabulary learning, and a slower tokenization of data. BPE can also helps models to learn more isotropic embedding representations. Future work will explore more in depth the isotropy of clusters of embeddings of generative models. We also plan to experiment with larger model, dataset and vocabulary sizes, hoping to find guidelines for choosing an optimum vocabulary size.

References

Arora, S., Li, Y., Liang, Y., Ma, T., and Risteski, A. A Latent Variable Model Approach to PMI-based Word Embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399, 07 2016. ISSN 2307-387X. doi: 10.1162/tacl_a_00106. URL https://doi.org/10.1162/tacl_a_00106.

Biś, D., Podkorytov, M., and Liu, X. 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*, pp. 5117–5130, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.403. URL https://aclanthology.org/2021.naacl-main.403.

Cai, X., Huang, J., Bian, Y., and Church, K. Isotropy in the contextual embedding space: Clusters and manifolds. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=xYGNO86OWDH.

Choi, K., Hawthorne, C., Simon, I., Dinculescu, M., and Engel, J. Encoding musical style with transformer autoencoders. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1899–1908. PMLR, Jul 2020. URL https://proceedings.mlr.press/v119/choi20b.html.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 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)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Donahue, C., Mao, H. H., Li, Y. E., Cottrell, G. W., and McAuley, J. J. 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*, pp. 685–692, 2019. URL http://archives.ismir.net/ismir2019/paper/000083.pdf.

Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., and Yang, Y.-H. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. URL https://ojs.aaai.org/index.php/AAAI/article/view/11312.

Ethayarajh, K. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 55–65, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1006. URL https://aclanthology.org/D19-1006.

Fradet, N., Briot, J.-P., Chhel, F., El Fallah Seghrouchni, A., and Gutowski, N. 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. URL https://archives.ismir.net/ismir2021/latebreaking/000005.pdf.

Gage, P. A new algorithm for data compression. *C Users J.*, 12(2):23–38, feb 1994. ISSN 0898-9788. URL https://dl.acm.org/doi/10.5555/177910.177914.

Gao, J., He, D., Tan, X., Qin, T., Wang, L., and Liu, T. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkEYojRqtm.

Gong, C., He, D., Tan, X., Qin, T., Wang, L., and Liu, T.-Y. Frage: Frequency-agnostic word representation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/e555ebe0ce426f7f9b2bef0706315e0c-Paper.pdf.

Hadjeres, G. and Crestel, L. The piano inpainting application, 2021. URL https://arxiv.org/abs/2107.05944.

Hadjeres, G., Pachet, F., and Nielsen, F. DeepBach: a steerable model for Bach chorales generation. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1362–1371. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/hadjeres17a.html.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

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

Hsiao, W.-Y., Liu, J.-Y., Yeh, Y.-C., and Yang, Y.-H. Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1):178–186, May 2021. doi: 10.1609/aaai.v35i1.16091. URL https://ojs.aaai.org/index.php/AAAI/article/view/16091.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. Music transformer, 2018. URL https://arxiv.org/abs/1809.04281.

Huang, Y.-S. and Yang, Y.-H. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, pp. 1180–1188, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379885. doi: 10.1145/3394171.3413671. URL https://doi.org/10.1145/3394171.3413671.

Kermarec, M., Bigo, L., and Keller, M. 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. URL https://ismir2022program.ismir.net/lbd_369.html.

Kilgour, K., Zuluaga, M., Roblek, D., and Sharifi, M. Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *Interspeech*, pp. 2350–2354, 09 2019. doi: 10.21437/Interspeech.2019-2219. URL https://www.isca-speech.org/archive_v0/Interspeech_2019/pdfs/2219.pdf.

Kingma, D. P. and Ba, J. 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. URL http://arxiv.org/abs/1412.6980.

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

Liang, F. T., Gotham, M., Johnson, M., and Shotton, J. Automatic stylistic composition of bach chorales with deep LSTM. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pp. 449–456, 2017. URL https://ismir2017.smcnus.org/wp-content/uploads/2017/10/156_Paper.pdf.

Liang, Y., Cao, R., Zheng, J., Ren, J., and Gao, L. 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*, pp. 448–459, Berlin, Heidelberg, 2021. Springer-Verlag. ISBN 978-3-030-86382-1. doi: 10.1007/978-3-030-86383-8_36. URL https://doi.org/10.1007/978-3-030-86383-8_36.

Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Liu, J., Dong, Y., Cheng, Z., Zhang, X., Li, X., Yu, F., and Sun, M. Symphony generation with permutation invariant language model. In *Proceedings of the 21rd International Society for Music Information Retrieval Conference*, Bengalore, India, December 2022. ISMIR. URL https://arxiv.org/abs/2205.05448.

McInnes, L., Healy, J., Saul, N., and Grossberger, L. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018. URL https://joss.theoj.org/papers/10.21105/joss.00861.

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

Mittal, G., Engel, J. H., Hawthorne, C., and Simon, I. Symbolic music generation with diffusion models. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, pp. 468–475, 2021. URL https://archives.ismir.net/ismir2021/paper/000058.pdf.

Mu, J. and Viswanath, P. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkuGJ3kCb.

Oore, S., Simon, I., Dieleman, S., Eck, D., and Simonyan, K. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, 32:955–967, 2018. URL https://link.springer.com/article/10.1007/s00521-018-3758-9.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

Payne, C. Musenet, 2019. URL https://openai.com/blog/musenet.

Press, O. and Wolf, L. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 157–163, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2025.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners, 2019. URL https://openai.com/blog/better-language-models/.

Reif, E., Yuan, A., Wattenberg, M., Viegas, F. B., Coenen, A., Pearce, A., and Kim, B. Visualizing and measuring the geometry of bert. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/159c1ffe5b61b41b3c4d8f4c2150f6c4-Paper.pdf.

Ren, Y., He, J., Tan, X., Qin, T., Zhao, Z., and Liu, T.-Y. Popmag: Pop music accompaniment generation. In *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 1198–1206. Association for Computing Machinery, 2020. ISBN 9781450379885. doi: 10.1145/3394171.3413721. URL https://doi.org/10.1145/3394171.3413721.

Rudman, W., Gillman, N., Rayne, T., and Eickhoff, C. IsoScore: Measuring the uniformity of embedding space utilization. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 3325–3339, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.262. URL https://aclanthology.org/2022.findings-acl.262.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

Sturm, B. L., Santos, J. F., and Korshunova, I. 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. URL https://ismir2015.ismir.net/LBD/LBD13.pdf.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qVyeW-grC2k.

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

von Rütte, D., Biggio, L., Kilcher, Y., and Hoffman, T. Figaro: Generating symbolic music with fine-grained artistic control, 2022. URL https://arxiv.org/abs/2201.10936.

Wang, L., Huang, J., Huang, K., Hu, Z., Wang, G., and Gu, Q. Improving neural language generation with spectrum control. In *International Conference on Learning Representations*, 2020a. URL https://openreview.net/forum?id=ByxY8CNtvr.

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

Yang, L.-C. and Lerch, A. On the evaluation of generative models in music. *Neural Comput. Appl.*, 32(9): 4773–4784, may 2020. ISSN 0941-0643. doi: 10.1007/s00521-018-3849-7. URL https://doi.org/10.1007/s00521-018-3849-7.

Zeng, M., Tan, X., Wang, R., Ju, Z., Qin, T., and Liu, T.-Y. MusicBERT: Symbolic music understanding with large-scale pre-training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 791–800, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.70. URL https://aclanthology.org/2021.findings-acl.70.

A. Model and training

Table 5. Model configurations. The number of parameters is based on the baseline with no BPE, and may vary depending on the baseline with the size of the first and last layers. Gen stands for generator and Cla for classifier.

	Gen	Cla_{small}	Cla_{large}
Dimension	512	768	1024
Nb attention heads	8	12	16
Nb layers	10	10	18
Feedforward size	2048	2048	3078
Parameters	32.6M	58.0M	193.3M

The sizes of the models are reported in Table 5. The generator is trained with a teacher forcing objective on 100k steps. The classifier pre-trained on 60k steps to retrieve the value of randomized positions. Between 1 to 15% of each input sequences is randomized during pre-training. It is then fine-tuned on 100k steps to predict the composer of the input sequence, from the first output hidden state, i.e., the `BOS` position, which is projected through an output classification layer. The input embedding and output pre-training module weights are tied to improve the performances (Press & Wolf, 2017).

The batch size is set to 16 for the generator, and 24 for the classifier. All trainings are done with automatic mixed-precision (Micikevicius et al., 2018), the Adam optimizer (Kingma & Ba, 2015) 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. We use a one cycle learning rate scheduler: the initial learning rate is close to 0 and gradually grows for the 30% first steps to $5e-6$, $1e-6$ and $5e-7$ for the generators, classifier pre-training and classifier fine-tuning respectively, then slowly decreases down to 0. We perform 5 validations steps every 30 training steps, and compute their average accuracy and loss. The model parameters are saved when the validation loss is the lowest ever observed, and after training the last version saved is used for testing. The training is stopped early if the validation losses did not decrease for 15k steps and 25k steps for respectively the generator and classifier.

B. Data downsampling

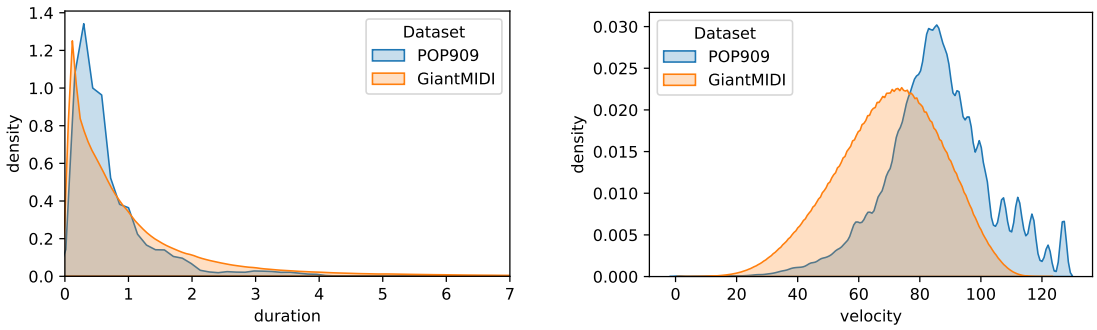


Figure 7. Distributions of the note durations and velocities of the POP909 and GiantMIDI datasets. The duration axis is limited to 7 beats.

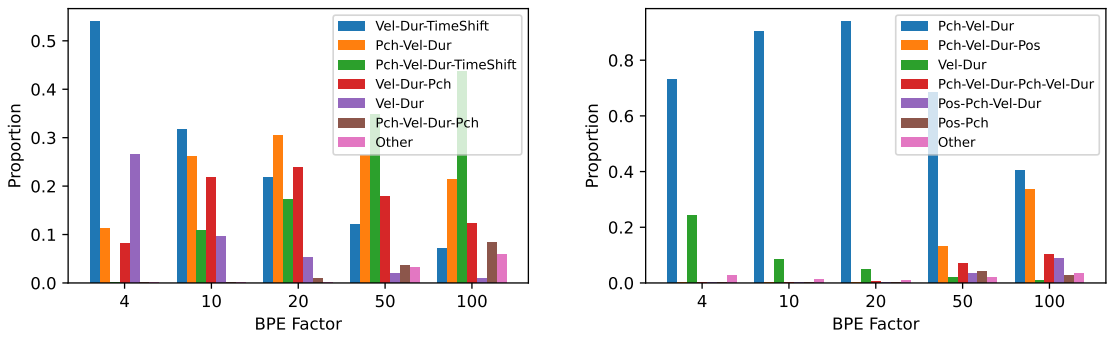
Figure 7 shows the distributions of velocity and duration values of the notes from the two datasets we use. As there is a larger proportion of low note durations (below two beats), we decided to downsample the `Duration` and `TimeShift` tokens with different resolutions: those up to one beat are downsampled 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. This way, short notes are represented more precisely than longer ones, reducing the vocabulary size. For *Remi*, `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[2]: 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.

[2]Available on the MIDI Manufacturers Association website

C. BPE Learning

Table 6. Vocabulary sizes, mean tokens per beat (tpb), and variation of tpb from without BPE, average tokenizing time and detokenizing time. A maximum of 1000k randomly sampled MIDI files were used for each row. Vocabulary sizes for CPWord and Octuple are the product of the sizes of their "sub-vocabularies", or in other words the number of possible token combinations, and are rounded for better readability. Tokenizing and detokenizing times were run on an Intel Xeon Gold 5128 CPU.

Data	Vocab. size	tpb	tpb variation (%)	Tok. time (sec)	Detok. time (sec)
POP909 TSD					
No BPE	139	17.81 ± 4.12	-	0.04 ± 0.02	0.01 ± 0.02
BPE×4	556	9.71 ± 2.12	-45.50	0.20 ± 0.05	0.02 ± 0.02
BPE×10	1390	8.05 ± 1.75	-54.80	0.44 ± 0.10	0.02 ± 0.02
BPE×20	2780	6.95 ± 1.53	-60.99	0.77 ± 0.18	0.02 ± 0.02
BPE×50	6950	5.84 ± 1.28	-67.20	1.59 ± 0.37	0.02 ± 0.02
BPE×100	13.9k	5.33 ± 1.16	-70.10	2.72 ± 0.63	0.02 ± 0.02
PVm	747	12.72 ± 2.92	-28.59	0.03 ± 0.01	0.01 ± 0.01
PVDm	14.1k	7.63 ± 1.73	-57.17	0.02 ± 0.01	0.01 ± 0.01
POP909 Remi					
No BPE	152	18.06 ± 4.12	-	0.03 ± 0.02	0.01 ± 0.01
BPE×4	608	10.55 ± 2.26	-41.61	0.21 ± 0.05	0.02 ± 0.02
BPE×10	1520	8.85 ± 1.90	-51.00	0.47 ± 0.11	0.02 ± 0.02
BPE×20	3040	8.01 ± 1.74	-55.64	0.86 ± 0.19	0.02 ± 0.02
BPE×50	7600	7.32 ± 1.58	-59.46	1.97 ± 0.43	0.02 ± 0.02
BPE×100	15.2k	6.70 ± 1.43	-62.92	3.64 ± 0.79	0.02 ± 0.02
PVm	760	12.97 ± 2.92	-28.19	0.02 ± 0.01	0.01 ± 0.01
PVDm	14k	7.88 ± 1.73	-56.38	0.02 ± 0.01	0.01 ± 0.01
CPWord	49k	7.88 ± 1.73	-56.38	0.03 ± 0.01	0.03 ± 0.02
Octuple	161k	5.09 ± 1.21	-71.81	0.02 ± 0.01	0.02 ± 0.02
GiantMIDI TSD					
No BPE	139	15.64 ± 6.29	-	0.08 ± 0.10	0.03 ± 0.05
BPE×4	556	8.87 ± 3.30	-43.26	0.45 ± 0.57	0.08 ± 0.16
BPE×10	1390	7.88 ± 2.86	-49.64	1.04 ± 1.29	0.07 ± 0.16
BPE×20	2780	7.04 ± 2.40	-54.98	1.90 ± 2.34	0.07 ± 0.15
BPE×50	6950	5.94 ± 2.20	-62.03	4.11 ± 5.03	0.07 ± 0.14
BPE×100	13.9k	5.45 ± 2.04	-65.15	7.49 ± 9.16	0.07 ± 0.14
PVm	747	11.26 ± 4.46	-28.03	0.06 ± 0.08	0.03 ± 0.04
PVDm	14.1k	6.57 ± 2.59	-57.98	0.06 ± 0.07	0.02 ± 0.03
GiantMIDI Remi					
no BPE	152	15.89 ± 6.42	-	0.08 ± 0.10	0.04 ± 0.05
BPE×4	608	9.58 ± 3.39	-39.70	0.53 ± 0.67	0.08 ± 0.18
BPE×10	1520	8.18 ± 2.96	-48.51	1.22 ± 1.51	0.08 ± 0.17
BPE×20	3040	7.22 ± 2.78	-54.56	2.18 ± 2.70	0.08 ± 0.17
BPE×50	7600	6.41 ± 2.42	-59.67	4.87 ± 5.98	0.08 ± 0.16
BPE×100	15.2k	5.96 ± 2.20	-62.51	9.08 ± 11.12	0.07 ± 0.15
PVm	760	11.30 ± 4.66	-28.90	0.06 ± 0.08	0.03 ± 0.04
PVDm	14k	6.94 ± 2.63	-56.29	0.05 ± 0.06	0.02 ± 0.03
CPWord	49k	6.90 ± 2.55	-56.60	0.07 ± 0.09	0.07 ± 0.10
Octuple	161k	4.37 ± 1.88	-72.51	0.05 ± 0.06	0.05 ± 0.07



(a) *TSD* (b) *Remi*

Figure 8. Normalized distributions of token types per BPE factor for the GiantMIDI dataset.

Table 6 shows the vocabulary sizes, sequence length variation and tokenization times of all baselines. When learning BPE, the average number of tokens per beat (tpb) quickly decreases, so the sequence length. As the vocabulary grows, the tpb decreases more slowly, as the most recurrent token successions have already be learned and replaced. A lower tpb allows to generate faster.

The tradeoff of BPE, besides the vocabulary learning time, is the tokenization time, as a MIDI file is first tokenized without BPE, then BPE is applied by finding the token subsequences to be replaced by the BPE tokens. The decoding step time, i.e., the time of the conversion of tokens to a MIDI file, is almost not impacted by BPE. The tokenization and detokenization times have been gotten with MidiTok (Fradet et al., 2021) which is implemented in Python. The tokenization time could be decreased if performed by a faster compiled language such as Rust or C. The Figure 8 complements the Figure 3, with the GiantMIDI dataset.

D. Learned embedding space

Figure 9 shows the singular values for the generative and classification models. As the different tokenizations features vocabularies with very different sizes, the values are normalized for better readability. Note that the *NoBPE* tokenizations feature vocabularies with a size inferior to the embedding dimension. *NoBPE adj.* corresponds to the *NoBPE* results adjusted to cover the x-axis on the whole embedding size.

Figure 10 shows the pairwise cosine similarity of the learned embedding vectors, for the *TSD* and *Remi* representation on the POP909 dataset. The first tokens up to 90 are `Pitches`, followed by `Velocities` up to 125, `Durations` up to 160 and then `Time-Shift` or `Position`. Without BPE, we can clearly distinguish patterns in the cosine similarity matrices. These high similarities shows that embeddings are close to each other. With BPE and larger vocabulary sizes, the average cosine similarity tend to decrease, especially between BPE tokens. Embeddings are less similar and more discriminative.

UMAP (McInnes et al., 2018) representations shown in Figure 6, Figure 12 and Figure 11 have been calculated with the default parameters of the official Python package. We clearly see that generative models learn clusters of embeddings of the same type, distant from each other. The embeddings do not occupy the space uniformly. On the other hand, pre-trained bi-directional models learn more isotropic embedding representations. The embeddings are spread uniformly across all directions, for all token types.

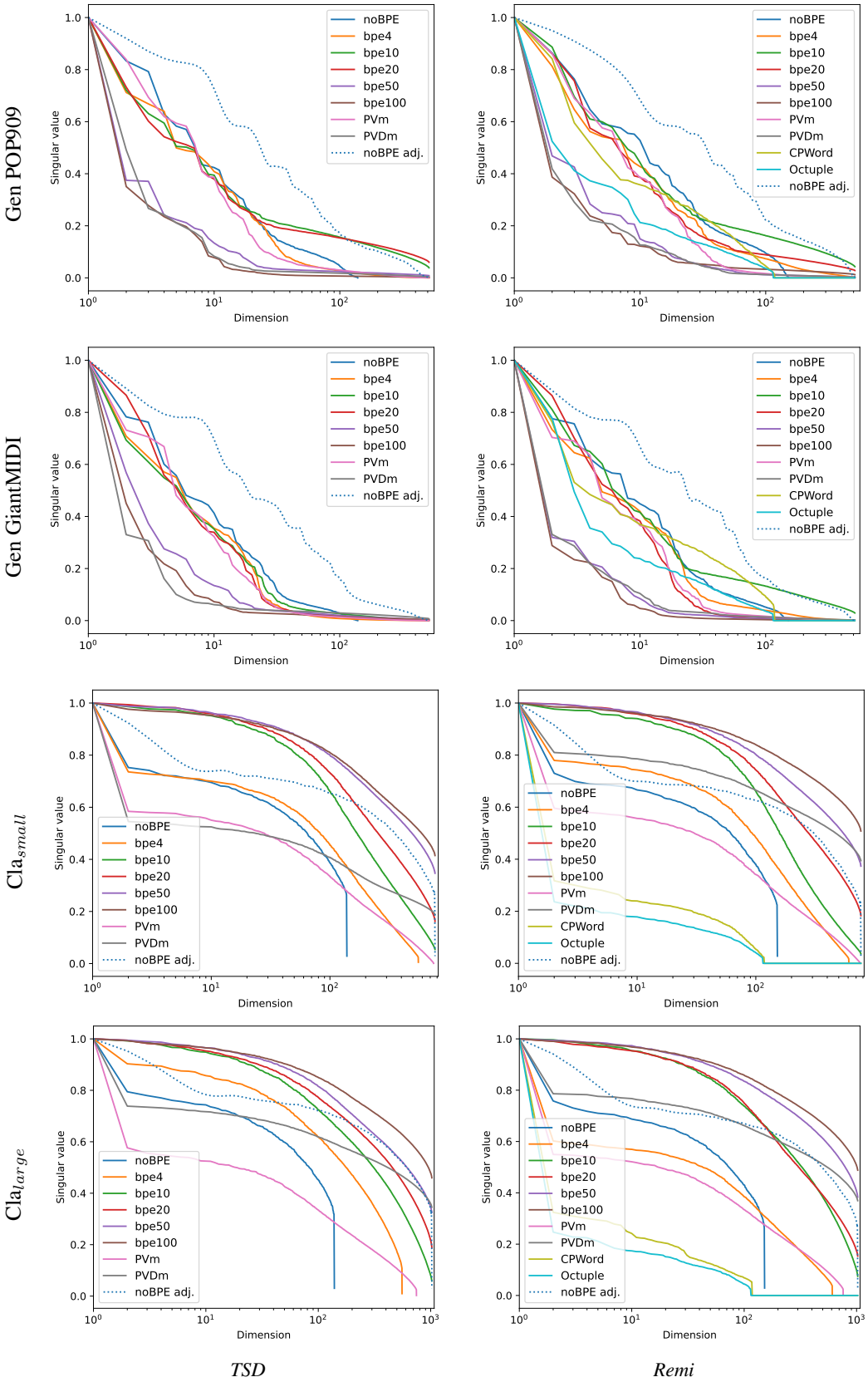


Figure 9. Normalized singular values of embedding matrices of classifier models.

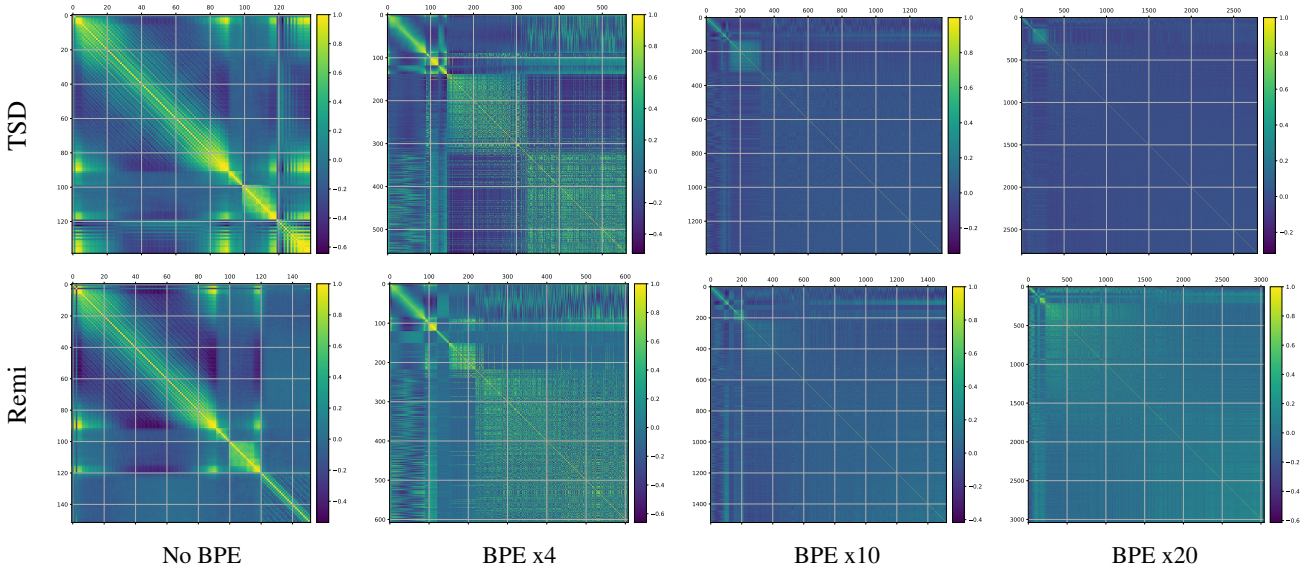


Figure 10. Pairwise cosine similarity matrix of learned embedding of the generative models, on the POP909 dataset.

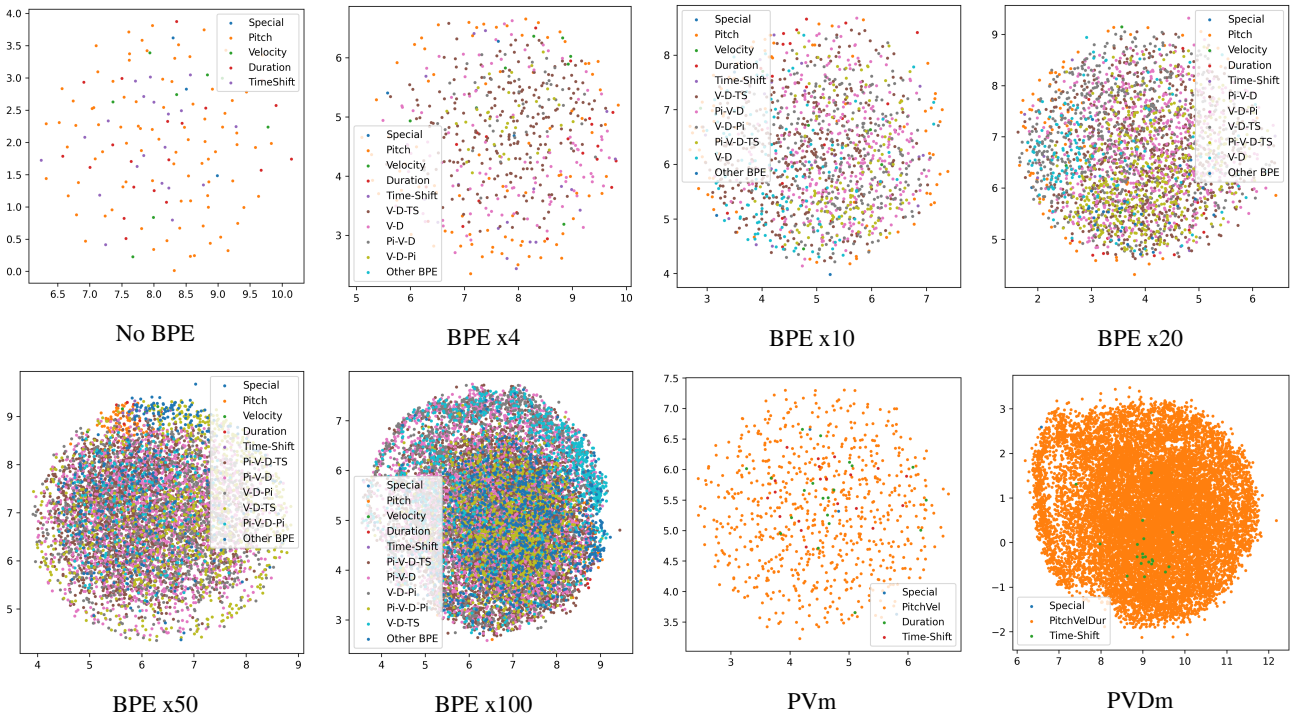


Figure 11. UMAP 2d representations of the embeddings of classifier models pre-trained with the GiantMIDI dataset and *TSD* tokenization. Abbreviations in legend stand for: Pi: Pitch; V: Velocity; D: Duration; Po: Position; TS: TimeShift.

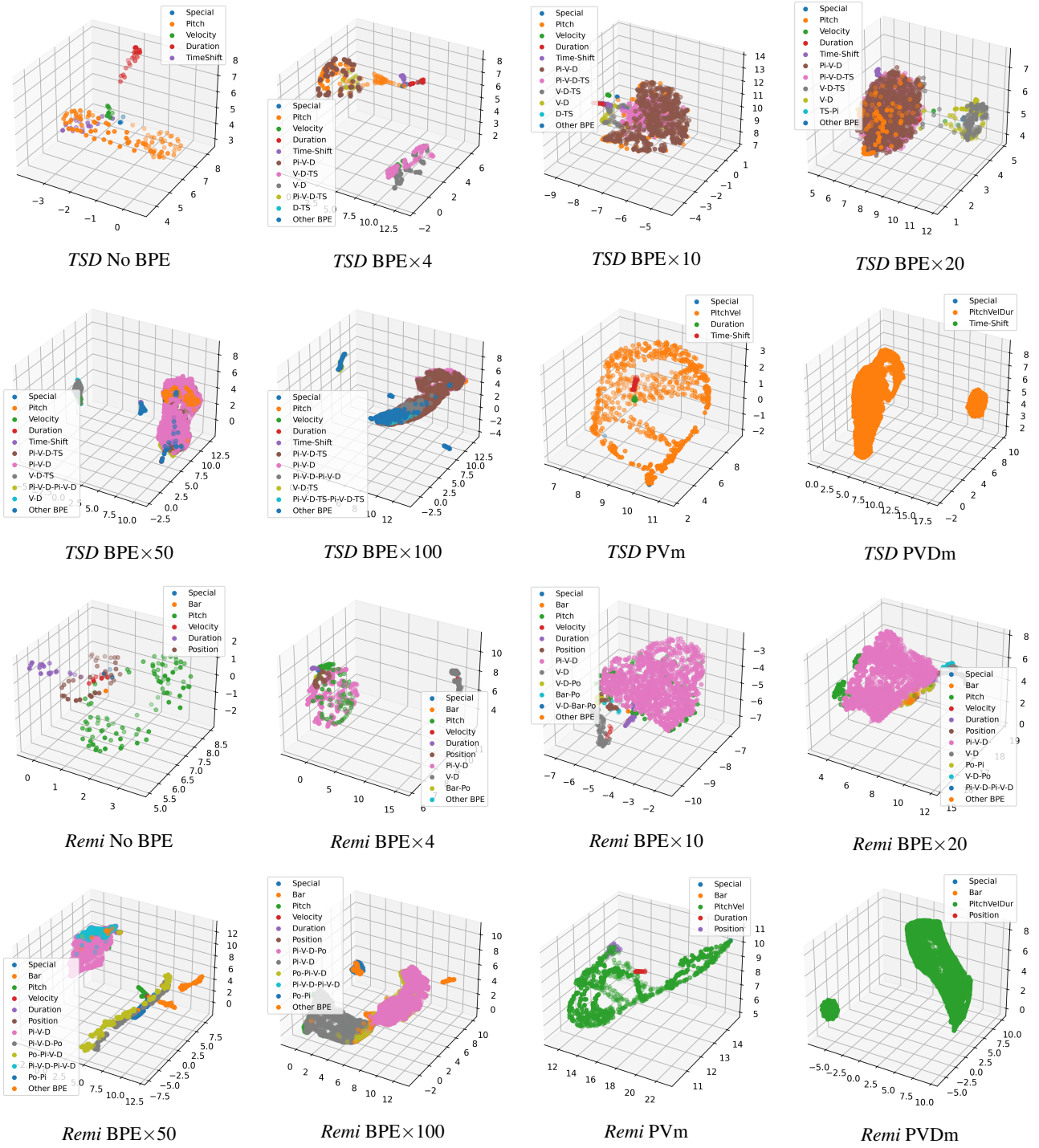


Figure 12. UMAP 3d representations of the embeddings of generative models with the POP909 dataset. Abbreviations in legend stand for: Pi: Pitch; V: Velocity; D: Duration; Po: Position: TS: TimeShift.