BMDATASET: A MUSICOLOGICALLY CURATED LILYPOND DATASET

Matteo SPANIO (spanio@dei.unipd.it) [1], **Ilay GULER** (ilay2004@bu.edu) [2], and **Antonio RODÁ** (roda@dei.unipd.it) [1]

[1] **University of Padua**, Padua, Italy
[2] **Boston University**, Boston, USA

ABSTRACT

Symbolic music research has relied predominantly on MIDI-based datasets; text-based engraving formats such as LilyPond remain unexplored for music understanding. We present **BMdataset**, a musicologically curated dataset of 391 LilyPond scores (2,645 movements) transcribed by experts directly from original Baroque manuscripts, with metadata covering composer, musical form, instrumentation, and sectional attributes. Building on this resource, we introduce **LilyBERT** [1], a CodeBERT-based encoder adapted to symbolic music through vocabulary extension with 115 LilyPond-specific tokens and masked language model pre-training. Linear probing on the out-of-domain Mutopia corpus shows that, despite its modest size (∼90M tokens), fine-tuning on BMdataset alone outperforms in terms of accuracy continuous pre-training on the full PDMX corpus (∼15B tokens) for both composer and style classification, demonstrating that small, expertly curated datasets can be more effective than large, noisy corpora for music understanding. Combining broad pre-training with domain-specific fine-tuning yields the best results overall (84.3% composer accuracy), confirming that the two data regimes are complementary. We release the dataset, tokenizer, and model to establish a baseline for representation learning on LilyPond.

1. INTRODUCTION

Recent work in music information retrieval (MIR) and generative AI has renewed attention to symbolic music representations [1–3]. While audio-domain models have seen rapid commercial development, symbolic formats remain essential for musicological analysis, score generation, and artist-in-the-loop workflows where fine-grained control over musical structure is required.

Among symbolic formats, MIDI dominates: large-scale datasets such as Lakh MIDI [4] and PDMX [5] support classification, generation, and retrieval tasks. However, MIDI discards much of the notational information present in written scores, including engraving directives, section boundaries, and the hierarchical structure of multi-movement works. MusicXML partially addresses this gap, but no comparable resources exist for *LilyPond* [2], a widely used open-source music engraving system whose text-based markup language encodes both musical content and typographic structure in a human-readable, version-controllable format.

LilyPond's syntax shares structural properties with programming languages: it uses nested blocks, backslash commands, variable assignments, and macro definitions. This observation motivates the use of *code pre-trained* language models, which have been shown to capture hierarchical and syntactic patterns in structured text [6], as a foundation for learning LilyPond representations.

In this paper, we make four contributions:

1. **BMdataset**, a musicologically curated dataset of 391 compilable LilyPond scores (2,645 movements) sourced from BaroqueMusic.it [3], transcribed directly from original manuscripts. The dataset is accompanied by structured metadata (composer, musical form, instrumentation, key, tempo, time signature, and historical period) and are released with `.ly`, `.midi`, `.pdf`, and `.json` metadata files.
2. A **LilyPond tokenizer** that extends the CodeBERT vocabulary with 115 domain-specific tokens, ensuring that musically meaningful commands (e.g., `\relative`, `\clef`, `\key`) are represented as atomic units rather than fragmented subwords.
3. **LilyBERT**, a CodeBERT-based encoder adapted to symbolic music through masked language model (MLM) pre-training on LilyPond corpora.
4. A systematic **probing evaluation** on the Mutopia corpus comparing four model variants that isolate the effects of pre-training data source, corpus size, and domain-specific fine-tuning on downstream composer and style classification.

2. RELATED WORK

2.1 Symbolic Music Datasets

Table 1 summarises existing symbolic music datasets. Most large-scale datasets use MIDI [4, 7, 8, 10] or MusicXML [5, 9] formats. MIDI is compact and widely

[1] https://huggingface.co/csc-unipd/lilybert

[2] https://lilypond.org

[3] https://www.baroquemusic.it, last visited: november 2025

Copyright: © 2026. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Table 1. Comparison of symbolic music datasets. BMdataset is the first musicologically curated LilyPond-native dataset.

Dataset	Format	Size	License
LMD [4]	MIDI	174K	CC-BY†
SymphonyNet [7]	MIDI	46K	–
MAESTRO [8]	MIDI	1.3K	CC-BY-NC-SA
Wikifonia [9]	MusicXML	6.4K	–
POP909 [10]	MIDI	909	–
Nottingham [11]	ABC	1.2K	PD
ABC-Tune [12]	ABC	165K	CC-0
PDMX [5]	MusicXML	254K	CC-0
Mutopia	LilyPond	2.1K	PD
BMdataset	**LilyPond**	**391**	**CC-BY-NC-ND**

† Contains documented copyrighted works [3].

supported but discards notational details such as engraving directives, articulation placement, and the hierarchical structure of multi-movement works. MusicXML preserves richer score information but remains tied to XML-based tooling. LilyPond, by contrast, represents music as structured plaintext, making it version-controllable, human-readable, and directly processable by language models without format conversion. While the Mutopia Project provides a community-curated collection of public domain LilyPond scores, it lacks structured metadata beyond basic composer and style tags and has not been used as a basis for representation learning. To our knowledge, no musicologically curated LilyPond dataset with systematic metadata annotation currently exists.

2.2 Music Representation Learning

Pre-trained Transformer models for symbolic music have improved both classification and generation benchmarks. MusicBERT [2] and MidiBERT-Piano [13] adapt BERT-style pre-training to MIDI representations and report strong downstream classification performance. Tokenisation-focused work [14, 15] demonstrates that compression-aware vocabularies can improve both quality and efficiency in MIDI-based modelling. On the generative side, MIDI-GPT [16] shows controllable multitrack composition in MIDI, while NotaGen [17] scales symbolic generation with ABC notation pre-training.

Most of these pipelines operate on MIDI. The principal text-based alternative, ABC notation [4], is more limited in structural and engraving expressivity, and parts of the ABC 2.1 specification are still marked as beta [18]. LilyPond exposes a richer score representation than MIDI, including engraving directives, reusable variables, and hierarchical score structure. Whether current language models effectively exploit this additional information remains an open question that this work begins to investigate.

2.3 Code Pre-trained Models

CodeBERT [6] is a bimodal pre-trained model for programming languages and natural language, based on the RoBERTa architecture [19]. Pre-trained on six programming languages using both MLM and replaced-token-detection (RTD) objectives, CodeBERT achieves strong results on code search, documentation generation, and clone detection. Subsequent work has incorporated richer structural signals: GraphCodeBERT [20] augments the input with data-flow graphs, while AST-aware approaches encode parse-tree structure directly. Because LilyPond files are compiled by a strict grammar, they could expose parse-tree or block-hierarchy information to such structure-aware encoders in future work.

Our motivation for selecting CodeBERT as the backbone is that LilyPond behaves more like a compiled formal language than a simple markup format: files are parsed according to a strict grammar and compiled into engraved score outputs. Its syntax employs code-like constructs such as nested blocks (`{ }`), backslash commands (`\relative`), variable bindings, includes, and macro definitions. We therefore hypothesise that a code-pretrained encoder is a better inductive fit than a model trained only on plain text token statistics. This parallels recent work showing that code-trained models can be effective on non-code structured languages such as chemical notation and mathematical expressions [21].

3. BMDATASET DATASET

3.1 Source and Compilation

The BMdataset dataset originates from BaroqueMusic.it, a collection of LilyPond projects transcribed by musicologists who worked directly from original manuscript sources. Each transcription is annotated with a reference to the original manuscript and its catalogue number, so that every score can be traced to its primary source. The majority of the collection consists of works by lesser-known Baroque composers; to our knowledge, this is the first musicologically curated LilyPond dataset focused on underrepresented figures of the period.

Each project is organised as a multi-file LilyPond workspace comprising: (i) a macro-definition file with shorthand musical markups; (ii) a header file specifying movement order via `\include` directives; (iii) individual movement files containing instrument-specific variables within `\relative` blocks; and (iv) a score file orchestrating all instruments across movements.

To produce self-contained files suitable for modelling, we parsed each project's header to extract the dependency order, concatenated the movement files sequentially, and appended the score file. The resulting files were validated by compiling each one with the LilyPond executable and scanning `stderr` for error messages. As a result, we obtained **391 compilable LilyPond files** comprising **2,645 individual movements** (i.e., self-contained musical units such as *Allegro*, *Adagio*), which in turn contain **1,679 tempo-marked sections** (passages delimited by tempo or character indications within a movement). The corpus totals approximately 90M tokens as measured by our extended tokenizer.

3.2 Metadata and Labels

Each file is annotated with the following metadata, stored in a structured JSON manifest:

[4] `https://abcnotation.com`

- **Composer** (71 unique): extracted from filename conventions and standardised.
- **Musical form** (16 unique): matched against a predefined list of Baroque forms (concerto, sonata, opera, suite, aria, cantata, etc.).
- **MIDI instruments** (25 unique): extracted via regex from `\set Staff.midiInstrument` directives.
- **Section metadata**: for each section (movement, dance, etc.), we record key, scale (major/minor), tempo, and time signature, extracted from `forma` variable blocks.
- **Historical period**: Early Baroque (<1650), High Baroque (1650–1700), Late Baroque (1700–1750), and Transitional Classical (>1750).

Section names were classified into semantically meaningful categories using a curated taxonomy: *speed* labels (e.g., allegro, largo), *intention* labels (e.g., affettuoso, maestoso), *suite* dances (e.g., bourrée, sarabande), *no-tempo* labels (e.g., recitativo, ouverture), and *non-descriptive* labels (e.g., aria, trio). For sections with uninformative names, quarter-note BPM was computed and mapped to standard tempo categories. This taxonomy is formalised as a directed acyclic graph (DAG) to preserve overlapping semantic relationships: for instance, suite dances such as *giga* belong to both the *suite* and *fast* speed categories.

3.3 Dataset Statistics

Figure 1 summarises the dataset distributions. The dataset exhibits characteristic imbalances: the composer distribution is heavily skewed towards Vivaldi (106 pieces, 30.5%), string instruments dominate the instrumentation (violin appears in 91.6% of files), and Late Baroque works (1700–1750) constitute 76.4% of the corpus. Concertos are the most common form (34.6%), followed by sinfonias (16.7%). The 1,679 individual sections span 65 unique section names, with speed-related labels being most frequent (1,212 occurrences across slow, mid, fast, and very fast categories), followed by suite dances (232) and non-tempo labels (137). The dataset nonetheless spans 71 composers, 16 musical forms, 25 instruments, 10 keys, and 12 time signatures.

What sets BMdataset apart from crowd-sourced alternatives is provenance. Each transcription was produced from the original manuscript, and the metadata records the manuscript reference and catalogue number for every score. This chain of evidence ties the musical content to the historical source, rather than to the editorial interpretations or modern reharmonisations found in community-contributed corpora.

3.4 Parser Validation

To verify the accuracy of the parsed musical content, we developed a batch-testing pipeline that compares the parser's note count against the number of rendered notehead glyphs in compiled PostScript outputs. Each `.ly` file was compiled with the `-ps` flag, and notehead glyphs were extracted via a regex search for `/noteheads.` patterns. After accounting for unused variable blocks (e.g., cadenza variables defined but never called in the score) and editorial incipits, 332 out of 391 files achieved a perfect 1:1 match. The remaining files exhibited a mean error of 0.26%, attributable to rare LilyPond formatting edge cases.

4. LILYBERT MODEL AND TOKENIZER

4.1 Tokenizer Design

LilyPond's syntax relies heavily on backslash-prefixed commands (e.g., `\relative`, `\clef`, `\key`, `\time`), which carry strong musical semantics. A standard BPE tokenizer, as used in CodeBERT, fragments these commands into meaningless subwords. For example, `\relative` might be split into `\`, `rel`, `ative`, destroying the semantic unit.

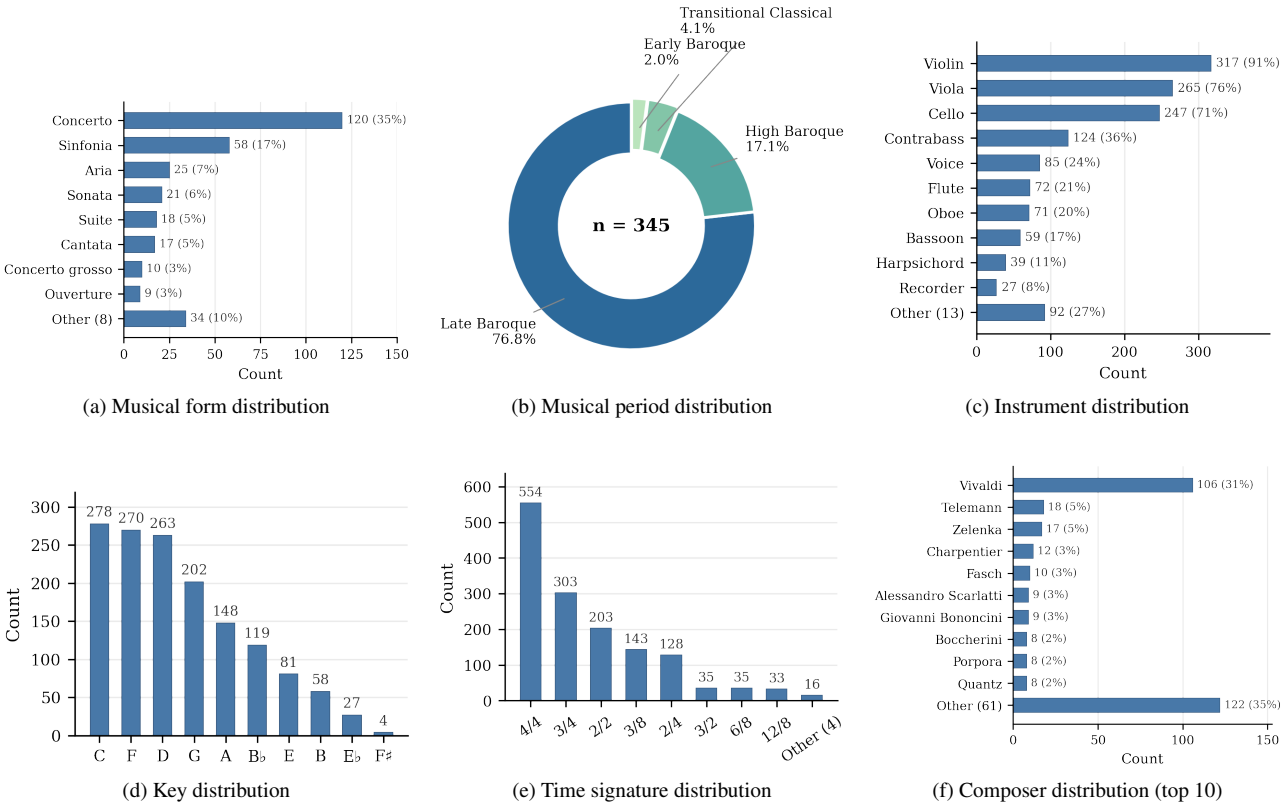
To address this, we extend the CodeBERT tokenizer (RoBERTa BPE, 50,265 vocabulary) with 115 LilyPond-specific tokens using the HuggingFace `add_tokens()` mechanism. The added tokens fall into six categories:

- **Musical commands** (15): `\time`, `\key`, `\clef`, `\tempo`, `\repeat`, `\tuplet`, `\grace`, `\partial`, etc.
- **Dynamics** (19): `\p`, `\pp`, `\f`, `\ff`, `\mp`, `\mf`, `\sfz`, `\cresc`, `\decresc`, etc.
- **Structural blocks** (20): `\score`, `\relative`, `\absolute`, `\transpose`, `\header`, `\layout`, `\midi`, `\markup`, etc.
- **Articulations/ornaments** (14): `\trill`, `\fermata`, `\mordent`, `\staccato`, `\accent`, `\arpeggio`, etc.
- **Key modes** (9): `\major`, `\minor`, `\dorian`, `\phrygian`, `\lydian`, `\mixolydian`, etc.
- **Property overrides and other** (38): `\override`, `\set`, `\once`, `\segno`, `\coda`, pedal marks, voice directives, etc.

The resulting tokenizer has a vocabulary of 50,380 tokens. Each added token is treated as a single, unsplittable unit during tokenisation, so musically meaningful commands remain atomic. Although the underlying RoBERTa architecture natively uses `<s>` and `</s>` as special tokens, our extended tokenizer remaps these to `[CLS]` and `[SEP]` for consistency with the BERT convention used in the probing literature.

4.2 Model Architecture

LilyBERT is based on CodeBERT (`microsoft/codebert-base`) [6], a RoBERTa-based [19] encoder with 12 transformer layers, 768 hidden dimensions, 12 attention heads, and approximately 125M parameters. We resize the token embedding matrix from 50,265 to 50,380 to accommodate the added LilyPond



(a) Musical form distribution (b) Musical period distribution (c) Instrument distribution

(d) Key distribution (e) Time signature distribution (f) Composer distribution (top 10)

Figure 1. BMdataset dataset statistics. The dataset is dominated by Late Baroque works, string-centric instrumentation, and concerto forms. Vivaldi accounts for 30.5% of compositions, while the remaining 70 composers contribute 1–18 pieces each.

vocabulary. The new embedding rows are randomly initialised and learned during pre-training.

As discussed in Section 2, LilyPond shares structural properties with programming languages (nested blocks, command–argument patterns, variable definitions, macro expansion), which makes CodeBERT's pre-training on six programming languages a natural starting point.

4.3 Corpus Preparation

Two LilyPond corpora are used for pre-training. The first is BMdataset (∼90M tokens, Section 3). The second, which we refer to as *PDMX-LilyPond*, is obtained by converting a subset of the PDMX dataset [5] from MusicXML to LilyPond with `musicxml2ly`, yielding approximately 15 billion tokens as measured by our extended tokenizer.

Two normalisation issues arise when combining these sources. First, pitch names must be consistent: the BMdataset scores, originally written in Italian pitch notation, are converted to LilyPond's `nederlands` pitch language prior to training. This matches the converted PDMX scores, for which `musicxml2ly` defaults to `nederlands`, as well as the majority of files in the Mutopia evaluation corpus. Second, automatic MusicXML-to-LilyPond conversion tends to produce syntactically valid but stylistically non-idiomatic code, including redundant overrides, verbose voicing constructions, and explicit directives that would rarely appear in hand-authored scores. We deliberately apply no post-conversion normalisation, so that PDMX-LilyPond represents the realistic outcome of a fully automatic pipeline; we return to the implications of this choice in Section 6.

4.4 Pre-training Procedure

LilyBERT is trained with the standard masked language modelling (MLM) objective: 15% of the input tokens are randomly masked and the model is trained to predict the original token from context. CodeBERT's original pre-training additionally employs a replaced-token-detection (RTD) objective, which requires a separate generator network and was shown to yield only marginal gains over MLM alone in CodeBERT's own ablations [6]. We therefore use MLM only, which simplifies the training pipeline and avoids training an auxiliary generator over our domain-specific vocabulary.

We consider two training stages, which the model variants of Section 5 combine in different ways:

1. **Continued pre-training on PDMX-LilyPond**: one epoch over the ∼15B-token corpus, establishing general LilyPond literacy.

2. **Domain-adaptive pre-training on BMdataset**: ten epochs over the ∼90M-token corpus. Note that this stage is still MLM-based and unsupervised; no label information is used at any point before the probing experiments.

The full two-stage model (Section 5, variant 4) applies stage 2 on top of stage 1; the remaining variants apply a single stage in isolation.

Input files are split into non-overlapping chunks of 512 tokens, each wrapped with `[CLS]` and `[SEP]`. Both stages use identical hyperparameters: an effective batch size of 288 across four NVIDIA A40 GPUs (48 GB) with Distributed Data Parallel (DDP), a learning rate of 2×10^{-4} with a cosine schedule and 10% warmup, weight decay 0.01, gradient clipping at 1.0, `bfloat16` mixed precision, the AdamW optimiser, and a maximum sequence length of 512 tokens. Each corpus is split into training and validation partitions, and early stopping with a patience of 5 is applied on the validation loss.

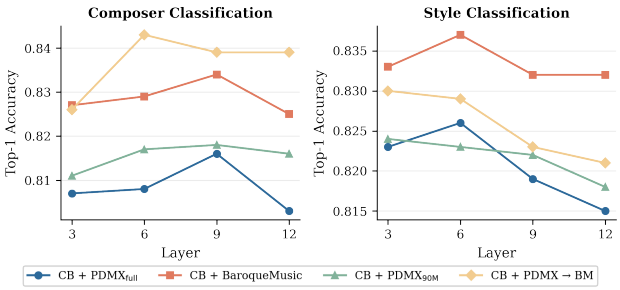


Figure 2. Layer-wise probing accuracy for composer and style classification. Composer information is best captured at middle layers (6–9), while style features are strongest in earlier layers and degrade towards the output.

5. EXPERIMENTS

5.1 Probing Setup

To evaluate whether LilyBERT's representations encode musically meaningful properties, we conduct linear probing experiments on the Mutopia corpus — a community-maintained collection of 2,123 public-domain LilyPond files spanning 320 composers and 13 musical styles. Mutopia is independent from BMdataset and covers a broader range of periods and styles, providing an out-of-domain evaluation setting.

Tasks. We evaluate two classification tasks: (1) **composer classification** and (2) **style/period classification**. To ensure sufficient training examples, we filter out classes with fewer than 10 pieces, yielding 30 composer classes (1,438 samples) and 10 style classes (1,972 samples).

Probe architecture. To prevent the model from exploiting authorship metadata, LilyPond file headers (containing composer name and revision history) are stripped prior to embedding extraction. Each file is split into non-overlapping 512-token chunks and processed by a frozen model; token-level hidden states are mean-pooled across all chunks to produce a single embedding per file at layers 3, 6, 9, and 12. Embeddings are standardised with `StandardScaler` and fed to a single linear layer classifier (MLP with no hidden layers). We report results using 5-fold stratified cross-validation.

Metrics. We report accuracy and macro-averaged precision and recall.

5.2 Models Compared

To isolate the effects of pre-training data source, corpus size, and domain-specific fine-tuning, we compare four model variants:

1. **CodeBERT + PDMX$_{\text{full}}$**: CodeBERT with continuous MLM pre-training on the full PDMX-LilyPond corpus (∼15B tokens, 1 epoch).

2. **CodeBERT + BMdataset**: CodeBERT fine-tuned on BMdataset (∼90M tokens, 10 epochs).

3. **CodeBERT + PDMX$_{\text{90M}}$**: CodeBERT fine-tuned on a size-matched PDMX-LilyPond subset (∼90M tokens, 10 epochs), enabling a fair comparison with variant 2 that controls for corpus size.

4. **CodeBERT + PDMX$_{\text{full}}$ → BMdataset**: two-stage training — continuous pre-training on full PDMX followed by fine-tuning on BMdataset (10 epochs).

Comparing variants 2 and 3 directly tests whether the domain specificity of BMdataset (musicologically curated, Baroque-focused) provides an advantage over a generic LilyPond corpus of equal size. Variant 4 tests whether large-scale pre-training followed by domain-specific fine-tuning is complementary.

5.3 Results

Table 2 presents the probing results for composer and style classification at layer 6, while Figure 2 shows performance across all probed layers. Figure 3 visualises the learned embedding space, and Figure 4 shows the composer confusion matrix for the best model.

Domain specificity vs. scale. Comparing CB + BMdataset against CB + PDMX$_{90M}$ (two models trained on corpora of equal size) reveals a consistent advantage for the musicologically curated data: BMdataset yields higher accuracy on both composer (+1.2pp) and style (+1.4pp) classification. This suggests that expert curation and stylistic coherence in the training corpus are more valuable than raw diversity when the downstream task requires fine-grained musical discrimination.

Complementarity of pre-training stages. The two-stage pipeline CB + PDMX → BM achieves the highest composer accuracy (0.843) and the best recall on both tasks, outperforming either pre-training stage alone. This supports a curriculum learning interpretation: broad-coverage pre-training on PDMX establishes general LilyPond literacy, while domain-specific fine-tuning on BMdataset sharpens the model's sensitivity to stylistic features. For style classification, however, the gains are more modest, with CB + BMdataset remaining the best model by top-1 accuracy (0.837). This asymmetry may reflect the fact that composer identity depends on fine-grained idiosyncratic patterns that benefit from large-scale pre-training, whereas broader stylistic categories (e.g.,

Table 2. Linear probing results (layer 6, mean ± std over 5 folds). Best in **bold**, second best underlined.

	Composer (30 classes)			Style (10 classes)		
Model	**Acc.**	**Prec.**	**Recall**	**Acc.**	**Prec.**	**Recall**
CB + $\text{PDMX}_{\text{full}}$	.808±.021	.751±.025	.751±.032	.826±.015	**.838**±.044	.781±.031
CB + BMdataset	.829±.013	.750±.039	.752±.036	**.837**±.004	.813±.047	.794±.016
CB + PDMX_{90M}	.817±.017	**.767**±.035	.747±.036	.823±.009	.805±.053	.775±.039
CB + PDMX → BM	**.843**±.010	.765±.031	**.775**±.028	.829±.011	.825±.013	**.797**±.035

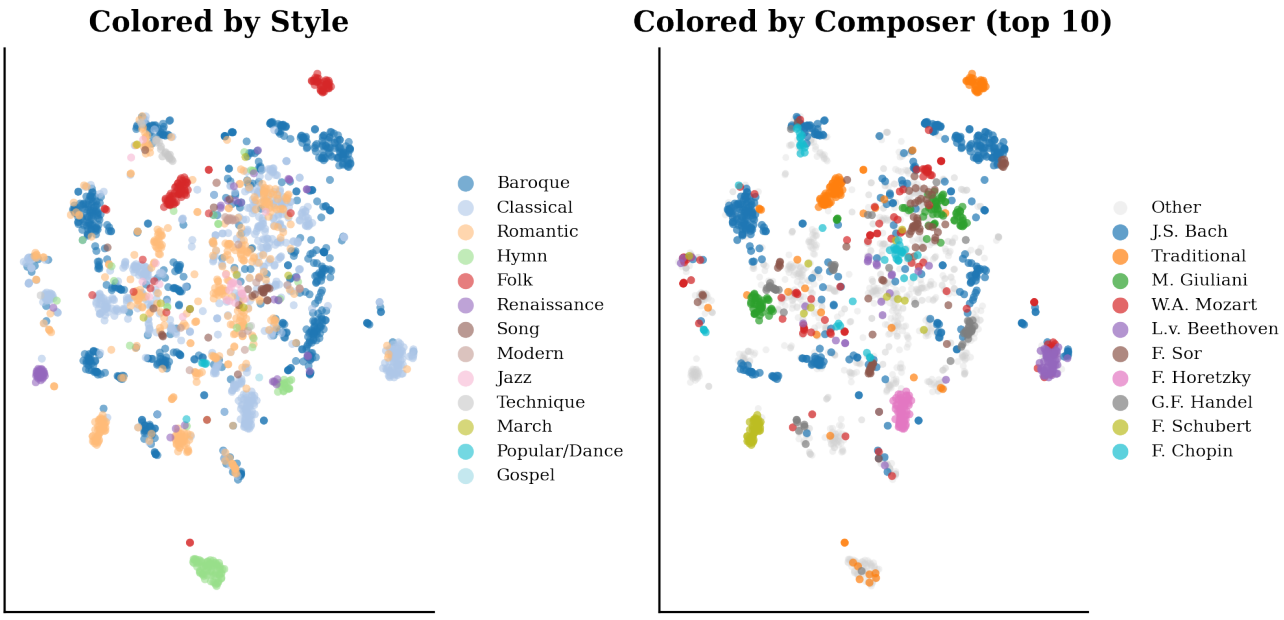


Figure 3. t-SNE projection of layer-6 embeddings (CB + PDMX → BM) on Mutopia, colored by style (left) and composer (right). Stylistic clusters (Baroque, Classical, Romantic) are clearly separated; within-style composer clusters indicate that the model captures both global period conventions and individual compositional idioms.

Baroque, Classical) are already well-captured by a smaller, domain-focused corpus.

Embedding structure and error analysis. The t-SNE projection in Figure 3 suggests that LilyBERT's representations organise along musically meaningful dimensions: style clusters (Baroque, Classical, Romantic) form distinct regions, while within each cluster, individual composers occupy localised sub-regions. The confusion matrix in Figure 4 reveals that most misclassifications occur between stylistically proximate composers — for example, Haydn is occasionally confused with Mozart, and Grieg with Chopin — suggesting that the model's errors reflect genuine musical similarity rather than random noise.

Layer-wise analysis. Following prior work on BERT probing [22], Figure 2 reveals distinct layer dynamics for the two tasks. Composer classification peaks at the middle layers (6–9), consistent with the view that composer identity requires integrating both syntactic patterns (note sequences, voicing choices) and higher-level structural preferences. Style classification, by contrast, is best captured at early-to-middle layers (3–6) and degrades towards the output layer across all models. This suggests that stylistic features (tempo conventions, instrumentation patterns, key preferences) are encoded as relatively surface-level distributional regularities, while the deeper layers progressively specioalise for the MLM objective at the expense of such global properties.

6. CONCLUSION

We introduced BMdataset, a musicologically curated LilyPond dataset transcribed directly from Baroque manuscript sources, together with an extended LilyPond tokenizer and LilyBERT, a CodeBERT-based encoder adapted to symbolic music through vocabulary extension and MLM pre-training.

Findings. Under a size-matched comparison (∼90M tokens, 10 epochs in both cases), pre-training on BMdataset outperforms pre-training on an automatically converted PDMX-LilyPond subset on both composer (+1.2pp) and style (+1.4pp) accuracy, indicating that expert curation and stylistic coherence can compensate for a lack of raw diversity. Notably, BMdataset also surpasses the accuracy of continued pre-training on the full ∼15B-token PDMX corpus, although that comparison confounds corpus size with the number of epochs and should be read as indicative rather than controlled. Finally, combining the two regimes yields the best composer accuracy overall (84.3%) and the best recall on both tasks, suggesting that broad-coverage pre-training and domain-specific adaptation are complementary rather than redundant.

Limitations. Three limitations qualify these results.

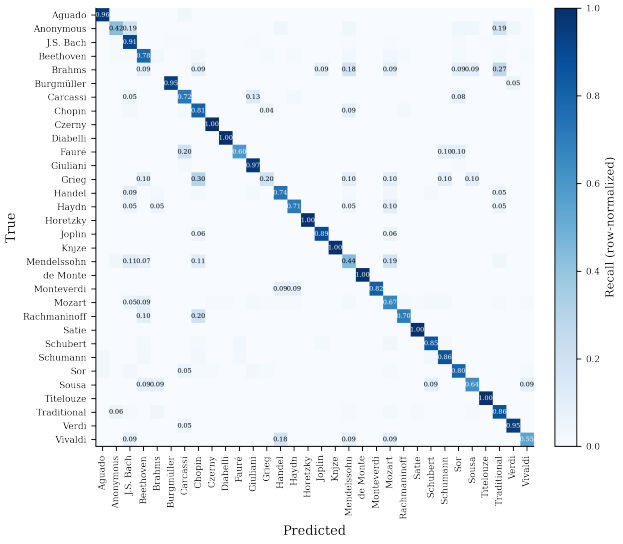


Figure 4. Row-normalised confusion matrix for composer classification (CB + PDMX → BM, layer 6, aggregated over 5 folds). Most misclassifications occur between stylistically related composers (e.g., Grieg ↔ Chopin, Haydn ↔ Mozart), suggesting that errors reflect genuine stylistic proximity rather than random noise.

First, the dataset is distributionally skewed towards Vivaldi, string instruments, and Late Baroque works. Evaluation on the broader Mutopia corpus indicates that the learned representations transfer beyond the training repertoire, but our experiments do not establish uniform performance across historical periods or instrumentations; Romantic piano literature, twentieth-century notation practices, and contemporary orchestral writing remain underrepresented. Second, the advantage of BMdataset over the size-matched PDMX subset cannot be attributed to curation alone: BMdataset is also stylistically closer to part of the Mutopia evaluation repertoire, and the two factors are not disentangled by our design. We hypothesise that the conversion artifacts described in Section 4.3 increase token diversity without adding musical information, but quantifying their impact remains an open question. Third, long scores are processed as independent, non-overlapping 512-token chunks whose representations are mean-pooled. This captures local syntactic and musical regularities but does not explicitly model long-range dependencies, making the current architecture better suited to movement-level representation learning than to modelling global musical form.

Future work. Each limitation suggests a direction. The corpus can be expanded to further periods and sources (e.g. Mutopia, IMSLP transcriptions) and rebalanced across composers and instrumentations. The effect of conversion artifacts can be isolated by applying post-conversion normalisation to PDMX-LilyPond and re-running the size-matched comparison. Long-range structure can be addressed with sparse attention or long-context transformers. Beyond these, LilyPond's parsable grammar invites structure-aware encoders (AST-based or graph-augmented models), and the hierarchical metadata released with BMdataset could serve as a conditioning signal for controllable generation, for instance through LoRA-based LilyPond generation on top of LilyBERT embeddings.

Because LilyPond is at once a music engraving system and a structured formal language, we expect representations learned on it to be useful for practical tasks such as score linting, style-aware auto-completion, and mixed-initiative editing — an expectation that our probing results motivate but do not yet test. We release BMdataset, the extended tokenizer, and LilyBERT as a baseline for representation learning on LilyPond.

Data availability.

All resources are publicly available to support reproducibility:

- **Code**: `https://github.com/CSCPadova/lilybert`
- **Model weights**: `https://huggingface.co/csc-unipd/lilybert`
- **Dataset**: `https://doi.org/10.5281/zenodo.18723290`

Acknowledgments

We thank Mario Bolognani and the BaroqueMusic.it community for granting access to their LilyPond transcriptions. This work was partially funded by the European Union - NextGenerationEU, under the National Recovery and Resilience Plan (PNRR).

7. REFERENCES

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

[2] 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)*, 2021.

[3] J. Thickstun, D. Hall, C. Donahue, and P. Liang, "Anticipatory music transformer," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[4] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching," Ph.D. dissertation, Columbia University, 2016.

[5] P. Long, Z. Novack, T. Berg-Kirkpatrick, and J. McAuley, "PDMX: A large-scale public domain MusicXML dataset for symbolic music processing," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025.

[6] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou, "CodeBERT: A pre-trained model for programming and natural languages," in *Findings of the Association for Computational Linguistics (EMNLP)*, 2020.

[7] J. Liu, Y. Dong, Z. Cheng, X. Zhang, X. Li, F. Yu, and M. Sun, "SymphonyNet: A benchmarking framework for large-scale symbolic music generation," *arXiv preprint arXiv:2207.05694*, 2022.

[8] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factored piano music modeling and generation with the MAESTRO dataset," in *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[9] "Wikifonia lead sheet dataset," [Online]. Available: https://www.wikifonia.org, accessed: 2025.

[10] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "POP909: A pop-song dataset for music arrangement generation," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[11] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proc. International Conference on Machine Learning (ICML)*, 2012.

[12] S. Wu, X. Li, F. Yu, and M. Sun, "TunesFormer: Forming Irish tunes with control codes by bar patching," in *Proc. AAAI Conference on Artificial Intelligence*, 2023.

[13] C.-H. Chou, I.-T. Lee, and Y.-H. Yang, "MIDI-BERT-Piano: Large-scale pre-training for symbolic music understanding," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[14] N. Fradet, J.-P. Briot, N. Gutowski, A. Choffrut, and G. Hadjeres, "Byte pair encoding for symbolic music," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2023.

[15] A. Poltronieri, M. Spanio, S. Chieppa, X. Serra, and M. Rocamora, "Does notation matter? musicxml tokenisation for symbolic music generation and understanding," in *Proceedings of the 27th International Society for Music Information Retrieval Conference (ISMIR 2026)*, 2026, accepted for publication.

[16] P. Pasquier, V. Cema, T. Paquet, and M. E. Yacoubi, "MIDI-GPT: A controllable generative model for computer-assisted multitrack music composition," *arXiv preprint arXiv:2501.17011*, 2025.

[17] Y. Wang, W. Liang, J. Han, G. Zhang, Z. Wang, L. Zhang, Y. Ding, W. Gao, and T.-S. Chua, "NotaGen: Advancing musicality in symbolic music generation with large language model training paradigms," *arXiv preprint arXiv:2504.00572*, 2025.

[18] "ABC standard v2.1," [Online]. Available: https://abcnotation.com/wiki/abc:standard:v2.1, accessed: 2026.

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[20] D. Guo, S. Ren, S. Lu, Z. Feng, D. Tang, S. Liu, L. Zhou, N. Duan, A. Svyatkovskiy, S. Fu, M. Tufano, S. K. Deng, C. Clement, D. Drain, N. Sundaresan, J. Yin, D. Jiang, and M. Zhou, "GraphCodeBERT: Pre-training code representations with data flow," in *Proc. International Conference on Learning Representations (ICLR)*, 2021.

[21] G. Dagan, O. Lieber, and R. Tsarfaty, "Getting the most out of your tokenizer for pre-training and domain adaptation," *arXiv preprint arXiv:2402.01035*, 2024.

[22] I. Tenney, D. Das, and E. Pavlick, "BERT rediscovers the classical NLP pipeline," in *Proc. Association for Computational Linguistics (ACL)*, 2019.