# BERT-like Pre-training for Symbolic Piano Music Classification Tasks

**Yi-Hui Chou***
Carnegie Mellon University, United States
yihuic@andrew.cmu.edu

**I-Chun Chen***
National Tsing Hua University, Taiwan
icchen0101@elsa.cs.nthu.edu.tw

**Chin-Jui Chang**
Research Center for IT Innovation, Academia Sinica, Taiwan
csc63182@citi.sinica.edu.tw

**Joann Ching**
Research Center for IT Innovation, Academia Sinica, Taiwan
joann8512@citi.sinica.edu.tw

**Yi-Hsuan Yang**
National Taiwan University, Taiwan
yhyangtw@ntu.edu.tw

## Abstract

This article presents a benchmark study of symbolic piano music classification using the masked language modelling approach of the Bidirectional Encoder Representations from Transformers (BERT). Specifically, we consider two types of MIDI data: *MIDI scores*, which are musical scores rendered directly into MIDI with no dynamics and precisely aligned with the metrical grids notated by their composers and *MIDI performances*, which are MIDI encodings of human performances of musical scoresheets. With five public-domain datasets of single-track piano MIDI files, we pre-train two 12-layer Transformer models using the BERT approach, one for MIDI scores and the other for MIDI performances, and fine-tune them for four downstream classification tasks. These include two note-level classification tasks (melody extraction and velocity prediction) and two sequence-level classification tasks (style classification and emotion classification). Our evaluation shows that the BERT approach leads to higher classification accuracy than recurrent neural network (RNN)-based baselines.

**Keywords:** Pre-trained model, Transformer, symbolic-domain music classification, piano music, melody recognition, velocity prediction, style classification, emotion classification

## 1 Introduction

In the literature on machine learning, a prominent approach to overcome the labelled data scarcity issue is to adopt "transfer learning" and divide the learning problem into two stages (Han *et al.*, 2021): a *pre-training* stage that establishes a model capturing general knowledge from one or multiple source tasks and a *fine-tuning* stage that transfers the captured knowledge to target tasks. Model pre-training can be done in two ways: using a labelled dataset (Choi *et al.*, 2018; J. Kim *et al.*, 2019; H.-H. Wu *et al.*, 2021), such as training a VGG-like model over millions of human-labelled clips of general sound events and then fine-tuning it on instrument classification (Gururani *et al.*, 2019); or using an unlabelled dataset with a self-supervised training strategy. The latter is in particular

*The first two authors contribute equally to the paper.

popular in the field of natural language processing (NLP), where pre-trained models (PTMs) using Transformers (Vaswani *et al.*, 2017) have achieved state-of-the-art results on almost all NLP tasks, including generative and discriminative ones (Han *et al.*, 2021).

This article presents an empirical study of applying PTMs to symbolic-domain piano music classification tasks. In particular, inspired by the growing trend of treating MIDI music as a "language" in deep generative models for symbolic music (C.-Z. A. Huang *et al.*, 2019; Y.-S. Huang & Yang, 2020; P. Lu *et al.*, 2023; Payne, 2019; S.-L. Wu & Yang, 2023), we employ a Transformer-based network pre-trained by a self-supervised training strategy called "masked language modelling" (MLM), which has been widely used in BERT-like PTMs in NLP (Conneau & Lample, 2019; Devlin *et al.*, 2019; Joshi *et al.*, 2020; Liu *et al.*, 2019; Z. Yang *et al.*, 2019). Despite the wide-scale adoption of BERT, we are aware of only two publications that employ BERT-like PTMs for symbolic music classification (Tsai & Ji, 2020; Zeng *et al.*, 2021). The first work (Tsai & Ji, 2020) deals with optically scanned sheet music, while we use MIDI inputs. The second work (Zeng *et al.*, 2021) uses a diverse set of multi-track MIDI files, while we intend to focus on piano music only. We discuss how our work differs from these two existing works in more detail in Section 2.

We evaluate PTMs on four piano music classification tasks. These include two *note-level classification tasks*, i.e., melody extraction (Y.-W. Hsiao & Su., 2021; Simonetta *et al.*, 2019) and velocity prediction (Jeong, Kwon, Kim, Lee, & Nam, 2019; Jeong, Kwon, Kim, & Nam, 2019; Widmer, 1994) and two *sequence-level classification tasks*, i.e., style classification (S. Kim *et al.*, 2020; Kong *et al.*, 2020) and emotion classification (Grekow & Raś, 2009; Y. Lin *et al.*, 2013; Panda *et al.*, 2013, 2018). We use five datasets in this work, amounting to 4,166 pieces of piano MIDI. We give details of the datasets and tasks in Sections 3 and 4.

As the major contribution of this article, we report a performance study of variants of PTM for this diverse set of classification tasks, comparing the proposed approach (Section 6) with recurrent neural network (RNN)-based baselines (Section 5). Results reported in Section 7 show that the "pre-train and fine-tune" strategy does lead to higher accuracy than the RNN baselines. Moreover, we consider two types of MIDI data and compare the performance of the resulting PTMs. Specifically, following Oore *et al.* (2018), we differentiate two types of MIDI files, *MIDI scores*, which are musical scoresheets rendered directly into MIDI with no dynamics and exactly according to the written metrical grid, and *MIDI performances*, which are MIDI encodings of human performances of musical scoresheets. All the 4,166 pieces we have are MIDI performances, but we can obtain the corresponding MIDI-score version of them by dropping performance-related information. Accordingly, we build two PTMs, one for MIDI scores and the other for MIDI performances and evaluate their performance respectively on the downstream tasks. While the MIDI-score version can be applied to a wider array of tasks involving those with or without performance-related information, the MIDI-performance version can likely perform better for tasks that involve human performance of piano scores, such as style classification and emotion classification. Therefore, such a performance comparison is relevant.

As the secondary contribution, we share the code and release checkpoints of the pre-trained and fine-tuned models publicly in our GitHub repository[2] with an open-source licence. Together with the fact that all the datasets employed in this work are publicly available, our research can be taken as a new testbed of PTMs in general and a new public benchmark for machine learning-based classification of MIDI music.

## 2 Related Work on Pre-trained Models for MIDI

Machine learning has been applied to music in symbolic formats such as MIDI. Exemplary tasks include symbolic-domain music genre classification (Correa & Rodrigues, 2016; Ferraro & Lemstrom, 2018), composer classification (S. Kim *et al.*, 2020; Kong *et al.*, 2020) and melody note identification (Y.-W. Hsiao & Su., 2021; Simonetta *et al.*, 2019). However, labelled datasets for symbolic-domain music data tend to be small in size in general (Hamanaka *et al.*, 2014; Harasim *et al.*, 2020; Simonetta *et al.*, 2019), posing challenges to train effective supervised machine learning models that generalise well.

To our best knowledge, the work of Tsai and Ji (2020) represents the first attempt to use PTMs for symbolic-domain music classification. They showed that either a RoBERTa-based Transformer

[2]https://github.com/wazenmai/MIDI-BERT

Table 1: Public datasets used for this article. All the datasets are used for pre-training, while three are also used for downstream classification tasks. Average note pitch is in MIDI number. The symbol "#" stands for "number of".

| Dataset | Downstream Classification (CLS) Tasks | Pieces | Duration (hours) | Avg. note pitch | Avg. note duration (in 𝅘𝅥𝅯) | Avg. #notes per bar | Avg. #bars per piece |
|---|---|---|---|---|---|---|---|
| Pop1K7 | - | 1,747 | 108.8 | E4 | 8.5 | 16.9 | 103.3 |
| $ASAP_{4/4}$ | - | 65 | 3.5 | D4# | 2.9 | 23.0 | 95.9 |
| $POP909_{4/4}$ | melody, velocity | 865 | 59.7 | D4# | 6.1 | 17.4 | 94.9 |
| Pianist8 | style | 411 | 31.9 | D4# | 9.6 | 17.0 | 108.9 |
| EMOPIA | emotion | 1,078 | 12.0 | C4# | 10.0 | 17.9 | 14.8 |

encoder PTM (Liu *et al.*, 2019) or a GPT2-based Transformer encoder PTM (Radford *et al.*, 2019) outperforms non-pre-trained baselines for a 9-class symbolic-domain composer classification task. Pre-training boosts the classification accuracy for the GPT2 model greatly from 46% to 70%. However, the symbolic data format considered in their work is "sheet music image" (Tsai & Ji, 2020), which are images of musical scores. This data format has been much less used than MIDI in the literature.

Zeng *et al.* (2021) presented *MusicBERT*, a PTM tailored for symbolic MIDI data. *MusicBERT* was trained on a non-public dataset of over one million multi-track MIDI pieces. The authors showcased the efficacy of *MusicBERT* by applying it to two generative music tasks, melody completion and accompaniment suggestion and two sequence-level discriminative tasks, including genre and style classification. In comparison to non PTM-based baselines, *MusicBERT* consistently led to better performance. Our work differs from theirs in the following aspects. First, our pre-training corpus is much smaller (only 4,166 pieces) but all publicly available, less diverse but more dedicated (to piano music). Second, we aim at establishing a benchmark for symbolic music classification and include not only sequence-level but also note-level tasks. Furthermore, the labelled data we employ for our downstream tasks is comparatively modest, with each dataset containing fewer than 1,000 annotated pieces. This differs from *MusicBERT*'s dataset, referred to as the TOP-MAGD dataset (Ferraro & Lemstrom, 2018), which comprises over 20,000 annotated pieces—a considerably extensive collection rarely encountered in symbolic music tasks. Finally, their token representation is designed for multi-track MIDI, while ours is for single-track piano MIDI, each MIDI file is an individual movement of a longer work.

## 3 Datasets and Data Representation

### 3.1 Piano MIDI Datasets

We collected four existing public-domain piano MIDI datasets, including Pop1K7, ASAP, POP909, EMOPIA and compiled a new dataset, named Pianist8. To simplify the token representation, we consider only MIDI files that specify 4/4 metre.[3] We list some important statistics of these five datasets in Tab. 1 and provide their details below.

- The **Pop1K7** dataset developed by W.-Y. Hsiao *et al.* (2021)[4] is composed of machine transcriptions of 1,747 audio recordings of piano covers (i.e., a new recording by someone other than the original artist or composer of a commercially released song) of Japanese anime, Korean and Western pop music, amounting to over 100 hours worth of data. The transcription was done with the "onsets-and-frames" RNN-based piano transcription model (Hawthorne *et al.*, 2018) (which is reported to attain a 95.32and the RNN-based downbeat and beat tracking model from the Madmom library (Böck *et al.*, 2016). This dataset is the largest among the five, constituting half of our training data. We only use it for pre-training.
- **ASAP**, the aligned scores & performances dataset compiled by Foscarin *et al.* (2020), [5] contains 1,068 MIDI performances of 222 Western classical music compositions from

[3]We note that the metre can be wrong due to errors in automatic music transcription, leading to noise in the data. Future work can be done to improve this. Moreover, future work can be done to use a more complicated token representation such as that proposed by Pati *et al.* (2019) to include other time signatures.

[4]https://github.com/YatingMusic/compound-word-transformer

[5]https://github.com/fosfrancesco/asap-dataset

15 composers, along with the MIDI performances of the 222 pieces compiled from the MAESTRO dataset (Hawthorne *et al.*, 2019). We consider it as an additional dataset for pre-training, using only the MIDI that specifies 4/4 metre with no metre change at all throughout the piece. This leaves us with 65 pieces of MIDI files, which last for 3.5 hours in total. Tab. 1 shows that, being the only classical dataset among the five, ASAP features shorter average note duration and larger number of notes per bar.

- **POP909** comprises piano covers of 909 pop songs compiled by Wang *et al.* (2020).[6] It is the only dataset among the five that provides melody, non-melody labels for each note. Specifically, each note is labelled with one of the following three classes: `vocal melody` (piano notes corresponding to the lead vocal melody line in the original pop song, usually during the verse and chorus parts); `instrumental melody` (piano notes corresponding to the secondary melody line played by the instruments in the original pop song, usually during the intro, bridge, outro parts); and `accompaniment` (including arpeggios, broken chords and many other textures).[7] As it is a MIDI performance dataset, it also comes with velocity information. Therefore, we use it for the melody classification and velocity prediction tasks. We discard pieces that do not specify 4/4 metre, ending up with 865 pieces for this dataset.
- **Pianist8** consists of eight artists' performances of piano music that we downloaded from YouTube for training and evaluating symbolic-domain style classification.[8] The artists are Richard Clayderman (pop), Yiruma (pop), Herbie Hancock (jazz), Ludovico Einaudi (contemporary), Hisaishi Joe (contemporary), Ryuichi Sakamoto (contemporary), Bethel Music (religious) and Hillsong Worship (religious). The artists are also the composers of the pieces, except for Richard Clayderman, Bethel Music and Hillsong Worship. The dataset contains a total of 411 pieces, with a fairly balanced number of pieces per artist. Each audio file is paired with its MIDI performance, which is machine-transcribed by the piano transcription model proposed by Kong *et al.* (2021).
- **EMOPIA** is a dataset of pop piano music collected recently by Hung *et al.* (2021) from YouTube for research on emotion-related tasks.[9] It has 1,087 clips (each around 30 seconds) segmented from 387 songs, covering Japanese anime, Korean & Western pop song covers, movie soundtracks and personal compositions. The emotion of each clip has been labelled using the following 4-class taxonomy: `HAHV` (high arousal high valence); `LAHV` (low arousal high valence); `HALV` (high arousal low valence); and `LALV` (low arousal low valence). This taxonomy is derived from Russell's valence-arousal model of emotion (Russell, 1980), where *valence* indicates whether the emotion is positive or negative and *arousal* denotes whether the emotion is high (e.g., angry) or low (e.g., sad) (Y.-H. Yang & Chen, 2011). The MIDI performances of these clips are similarly machine-transcribed from the audio recordings by the model of Kong *et al.* (2021). We use this dataset for the emotion classification task. As Tab. 1 shows, the average length of the pieces in the EMOPIA dataset is the shortest among the five, since they are actually clips manually selected by dedicated annotators (Hung *et al.*, 2021) to ensure that each performance expresses a single emotion.

All five datasets consist of MIDI performances. As mentioned in the introduction, we intend to build two PTMs, one for MIDI scores and the other for MIDI performances. We obtain the MIDI-score version of each performance by dropping velocity and tempo information, temporally quantising the onset time and duration of each the notes to a semiquaver resolution.

### 3.2 Token Representation

Similar to text, a piece of music in MIDI can be considered as a sequence of musical events or "tokens". However, what makes music different is that musical notes are associated with a temporal length (i.e., note duration) and multiple notes can be played at the same time. Therefore, to represent music, we need *note-related* tokens describing, for example, the pitch and duration of the notes, as well as *metric-related* tokens placing the notes over a time grid.

[6] https://github.com/music-x-lab/POP909-Dataset

[7] POP909 originally refers to `vocal melody` as `melody` and `instrumental melody` as `bridge` in their work (Wang *et al.*, 2020). We opt for our new naming to highlight the fact that the latter is also a type of melody.

[8] https://zenodo.org/record/5089279

[9] https://annahung31.github.io/EMOPIA/

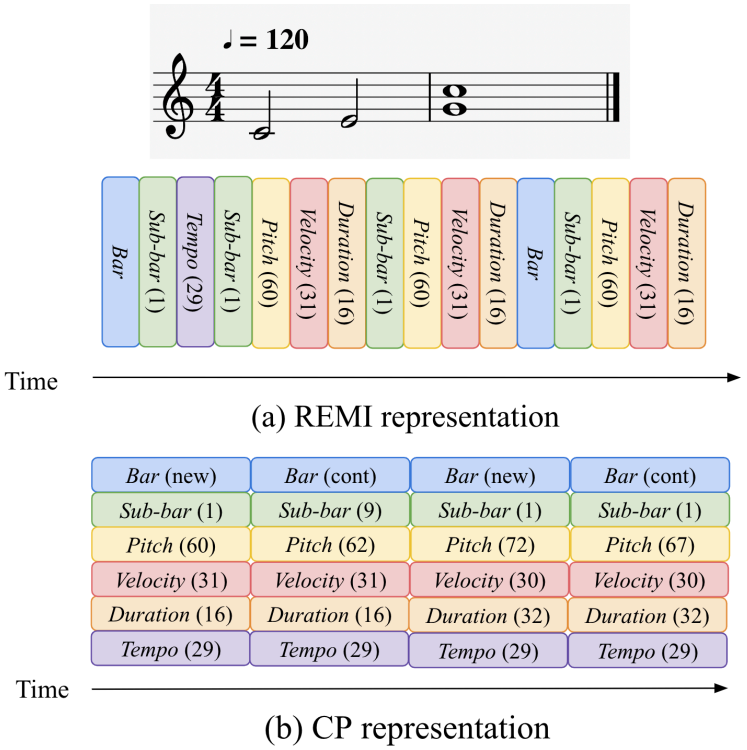



Figure 1: An example of a piece of score encoded using the proposed simplified version of the (a) REMI and (b) CP representations, using seven types of tokens, `Bar`, `Sub-bar`, `Pitch`, `Velocity`, `Duration`, `Tempo` and `Pad` (not shown here), for piano-only MIDI performance. The text inside parentheses indicates the value each token takes. While each time step corresponds to a single token in REMI, each time step would correspond to a *super token* that assembles four tokens in total in CP. Without such a token grouping, the sequence length (in terms of the number of time steps) of REMI is longer than that of CP (in this example, 16 *versus* 4). Please note that the actual scores employed in our work are not as simple as this example as they are polyphonic.

In the literature, a variety of token representations for MIDI have been proposed, differing in many aspects such as the MIDI data being considered (e.g., melody (Waite *et al.*, 2016), lead sheet (S.-L. Wu & Yang, 2020), piano (C.-Z. A. Huang *et al.*, 2019) and multi-track music (Dong *et al.*, 2023; Payne, 2019)), the temporal resolution of the time grid and the way the advancement in time is notated (Y.-S. Huang & Yang, 2020). Auxiliary tokens describing, for example, the chord progression (Y.-S. Huang & Yang, 2020) or grooving pattern (Chen *et al.*, 2020) underlying a piece can also be added.

In this work, we adopt the beat-based REMI token representation proposed by Y.-S. Huang and Yang (2020) to place musical notes over a discrete time grid comprising 16 equidistant sample points per bar. In addition to REMI, we experiment with the "token grouping" idea of the *compound word* (CP) representation (W.-Y. Hsiao *et al.*, 2021), to reduce the length of the token sequences. We depict the two adopted token representations in Fig. 1 and provide some details below.

#### 3.2.1 REMI Token Representation

The REMI representation (Y.-S. Huang & Yang, 2020) for MIDI performances uses `Bar` and `Sub-bar` tokens to represent the advancement in time. The former marks the beginning of a new bar, while the latter points to a discrete position within a bar. Specifically, as we divide a bar into 16 equidistant sample points, the `Sub-bar` tokens can take values from 1 to 16; e.g., `Sub-bar(1)` indicates the position corresponding to the first sample point in a bar, i.e., the first beat in 4/4 time signature,

whereas `Sub-bar(9)` indicates the third beat.[10] As depicted in Fig. 1(a), we use a `Sub-bar` token before each musical note, which comprises two consecutive tokens of `Pitch` and `Duration`. In other words, the `Sub-bar` token indicates the onset time of a note played at a certain MIDI pitch (i.e., the value taken by the `Pitch` token), whose duration is indicated by the `Duration` token, in multiples of demisemiquavers. For example, `Duration(1)` and `Duration(32)` correspond to a thirty-second note and a whole note, respectively. For MIDI performances, a musical note is represented by not only `Pitch` and `Duration` tokens but also a `Velocity` token that indicates how hard this note was pressed by key. Moreover, we use the `Tempo` token to specify the tempo information of the songs. It is placed behind the `Sub-bar` token to imply when the song would perform with the tempo. We only add the tempo token at the beginning of the song and the timing when tempo changes. For MIDI scores, the `Velocity` and `Tempo` tokens are simply dropped.

#### 3.2.2 CP Token Representation

Fig. 1(a) shows that, except for `Bar`, the other tokens in a REMI sequence always occur consecutively in groups, in the order of `Sub-bar`, `Pitch`, `Duration`. We can further differentiate `Bar(new)` and `Bar(cont)`, representing respectively the beginning of a new bar and a continuation of the current bar and always have one of them before a `Sub-bar` token. This way, the tokens would always occur in a group of four for MIDI scores. For MIDI performances, six tokens would be grouped together, including `Velocity` and `Tempo`. Following the logic of `Bar`, if there is no tempo change, we simply repeat the tempo value. Instead of feeding the token embedding of each of them individually to the Transformer, we can combine the token embedding of either the four tokens for MIDI scores or six tokens for MIDI performances in a group by concatenation and let the Transformer model process them jointly, as depicted in Fig. 1(b). We can also modify the output layer of the Transformer so that it predicts multiple tokens at once with different heads. These constitute the main ideas of the CP representation (W.-Y. Hsiao *et al.*, 2021), which has at least the following two advantages over its REMI counterpart: 1) the number of time steps needed to represent a MIDI piece is much reduced, since the tokens are merged into a "super token" (a.k.a. a "compound word" (W.-Y. Hsiao *et al.*, 2021)) representing four tokens at once; 2) the self-attention in Transformer is operated over the super tokens, which might be musically more meaningful as each super token jointly represents different aspects of a musical note. Therefore, we experiment with both REMI and CP in our experiments.

#### 3.2.3 On Zero-padding

To train Transformers, it is required that all input sequences have the same length. For both REMI and CP, we divide the token sequence for each entire piece into a number of shorter sequences with equal sequence length 512, zero-padding those at the end of a piece to 512 with an appropriate number of `Pad` tokens. Because of the token grouping, a CP sequence for the Pop1K7 dataset would cover around 25 bars on average, whereas a corresponding REMI sequence covers only 9 bars on average.

For MIDI scores, our final token vocabulary for REMI contains 16 unique `Sub-bar` tokens, 86 `Pitch` tokens, 64 `Duration` tokens, one `Bar` token, one `Pad` token and one `Mask` token, in total 169 tokens. For CP, we do not use a `Pad` token but represent a zero-padded super token by `Bar(Pad)`, `Sub-bar(Pad)`, `Pitch(Pad)` and `Duration(Pad)`. We do similarly for a masked super token, using `Bar(Mask)`, etc. We need an additional bar-related token `Bar(cont)` for CP, so the vocabulary size for CP is $169-2+8+1=176$. For MIDI performances, the vocabulary sizes are 299 and 310 using the REMI and CP representations, respectively.

## 4 Task Specification

Throughout this article, we refer to *note-level classification tasks* as tasks that perform a prediction for each individual note in a music sequence and *sequence-level tasks* as tasks that require a single prediction for an entire music sequence. We consider two note-level tasks and two sequence-level tasks in our experiments, as elaborated below.

[10] We note that Y.-S. Huang and Yang (2020) originally referred to such `Sub-bar` tokens as `Position` tokens, while Shih *et al.* (2022) and S.-L. Wu and Yang (2023) call them `Sub-beat` tokens. We prefer our naming for it is musically more accurate—our `Sub-bar` tokens are subdivisions of a bar (i.e., dividing a bar into 16 points), not subdivisions of a beat (i.e., not dividing a beat into 16 points).

### 4.1 Symbolic-domain Melody Extraction

For symbolic-domain melody extraction, initial methodologies predominantly adopted rule-based approaches. These rule-based methods encompassed techniques such as utilising pitch contour characteristics (Salamon & Gomez, 2012), as well as the implementation of the "skyline" algorithm (Chai & Vercoe, 2001). In recent years, deep learning-based approaches utilising convolutional neural networks (CNN) have been adopted (Y.-W. Hsiao & Su., 2021; Simonetta *et al.*, 2019), We will review such CNN-based methods in Section 5, highlighting their specific details and implementation. Similar to Simonetta *et al.* (2019), we regard *melody extraction* as a task that identifies the melody notes in single-track [11] homophonic or polyphonic music. Utilising the POP909 dataset of Wang *et al.* (2020), we can develop a model that classifies each Pitch event into vocal melody, instrumental melody or accompaniment, with classification accuracy (ACC) serving as the evaluation metric.[12]

Specifically, we consider two formulations of the task. Firstly, we adhere to the original configuration of POP909 and perform **three-class** melody classification, classifying each Pitch into three categories: vocal melody, instrumental melody or accompaniment. Secondly, we merge vocal melody and instrumental melody into a general "melody" category (while accompaniment is designated as "non-melody") and perform **two-class** classification. Doing so allows for a direct comparison with established baselines, such as the skyline algorithm and the baseline introduced in Section 5. For detailed results and a thorough examination, please refer to Section 7.1.

### 4.2 Symbolic-domain Velocity Prediction

Dynamics is an important element in music, as they are often used by musicians to add excitement and emotion to songs. Given that the tokens we choose do not contain performance information, it is interesting to see how a machine model would "perform" a piece by deciding these volume changes, a task that is essential in *performance generation* (Jeong, Kwon, Kim, Lee, & Nam, 2019; Jeong, Kwon, Kim, & Nam, 2019; Widmer, 1994) or *expressive performance modelling* (Friberg *et al.*, 2006; Friberg *et al.*, 2007). In the realm of MIDI, velocity is a parameter that scales the intensity or volume at which a sound sample is played back, with the value ranging from 0 to 127. Default MIDI velocity values are associated with dynamic indications. Apple's Logic Pro 9 user manual correlates traditional volume indicators (`pp`, `p`, `mp`, `mf`, `f`, `ff` and `fff`) with specific MIDI velocity values (16, 32, 48, 64, 80, 96, 112 and 127), respectively.[13] In our work, we define and classify this information into six categories: `pp` (0–31), `p` (32–47), `mp` (48–63), `mf` (64–79), `f` (80–95) and `ff` (96–127). Our definition aligns with the Logic Pro 9 specifications, except that we treat `fff` as equivalent to `ff`. Our objective can be treated as a note-level classification task, aiming to classify `Pitch` events into six classes using the POP909 dataset (Wang *et al.*, 2020).

### 4.3 Symbolic-domain Style Classification

Genre classification (Correa & Rodrigues, 2016) can be considered as a type of style classification. While genre classification categorises music based on shared musical attributes and conventions, style classification seeks to capture the nuanced stylistic variations within either a specific genre, composer or performer, accounting for the diverse artistic choices and performance practices that shape musical expressions. We could relatively easily find out which type of music we are listening to based on the similar patterns in that genre, while needing more musical insights to recognise the composer's or performer's style. Deep learning-based composer classification in MIDI has been attempted by S. Kim *et al.* (2020) and Kong *et al.* (2020), both treating MIDI pieces as 2D-representation matrices (via the piano-roll representation) and using CNN classifiers. Our work differs from theirs in that: 1) we encode MIDI pieces as token sequences, 2) we employ PTM, 3) we consider non-classical music

[11] It is common for MIDI files to consist of multiple tracks. We refer to "single-track" as MIDI files containing only one track, which is in contrast to multi-track MIDI files that have multiple tracks.

[12] We note that there is a task closely related to melody extraction, called *melody track identification*. The goal of this task is to distinguish the melody track from other non-melody tracks present in a multi-track MIDI file (Jiang & Dannenberg, 2019; Madsen & Widmer, 2007). While melody extraction is a note-level classification task, melody track identification is a track-level task. The latter is also an important symbolic music classification task, but we do not consider it here for we exclusively focus on piano-only data.

[13] https://help.apple.com/logicpro/mac/9.1.6/en/logicpro/usermanual/ (page 468 in the user manual; accessed 2023-06-22)

pieces and 4) our task is about style classification because not all the pianists in Pianist8 composed the pieces they performed.

### 4.4 Symbolic-domain Emotion Classification

Emotion classification in MIDI has been approached by a few researchers, mostly using hand-crafted features and non-deep learning classifiers (Grekow & Raś, 2009; Y. Lin *et al.*, 2013; Panda *et al.*, 2013, 2018). Some researchers work on MIDI alone, while others use both audio and MIDI in multi-modal emotion classification (Panda *et al.*, 2013). The only deep learning-based approach we are aware of is presented by Hung *et al.* (2021), using an RNN-based classifier called "Bi-LSTM-Attn" (Z. Lin *et al.*, 2017) but without employing PTMs, which is also used as a baseline in our experiment; see Section 5.

## 5 Baseline Model

For the note-level classification tasks, we use an RNN model as our baseline that consists of three bi-directional long short-term memory (Bi-LSTM) layers, each with 256 neurons and a feed-forward layer for classification, since such a network has led to state-of-the-art result in many audio-domain music classification tasks, like beat tracking (Böck *et al.*, 2016; Chiu *et al.*, 2021) and pitch estimation (Hawthorne *et al.*, 2018). All of our downstream tasks can be viewed as a multi-class classification problem. Given a REMI sequence, a Bi-LSTM model makes a prediction for each `Pitch` token, ignoring all the other types of tokens (i.e., `Bar`, `Sub-bar`, `Duration` and `Pad`). For CP, the Bi-LSTM model simply makes a prediction for each super token, again ignoring the zero-padded ones.

For the sequence-level tasks, which require only a prediction for an entire sequence, we follow Hung *et al.* (2021) and choose the Bi-LSTM-Attn model from Z. Lin *et al.* (2017) as our baseline, which was originally proposed for sentiment classification in NLP. The model combines LSTM with a self-attention module for temporal aggregation. Specifically, it uses a Bi-LSTM layer to convert the input sequence of tokens into a sequence of embeddings, which can be considered as feature representations of the tokens and then fuses these embeddings into one sequence-level embedding according to the weights assigned by the attention module to each token-level embedding. The sequence-level embedding then goes through two dense layers for classification. We use the token-level embedding for all the tokens here.

For melody extraction, we implement additionally the skyline algorithm (Chai & Vercoe, 2001) and a CNN-based method (Simonetta *et al.*, 2019) for performance comparison. The skyline algorithm can only perform "melody *versus* non-melody" two-class classification for it cannot distinguish between vocal melody and instrumental melody—it uses the simple rule of taking the note with the highest pitch among the concurrent notes as the melody, while avoiding temporally overlapping notes (Chai & Vercoe, 2001).

The CNN method (Simonetta *et al.*, 2019) uses piano-roll, a 2D representation where the x-axis represents symbolic time and the y-axis represents pitch, to represent MIDI. Their CNN learns to predict the probability that each note belongs to the melody line. Then, a clustering algorithm is used to find a threshold for each piece adaptively. Finally, the Bellman-Ford algorithm is adopted to pick a strictly monophonic melody line. In contrast, we do not have postprocessing steps such as thresholding or clustering in our BERT-based model and the RNN baseline.

The source code for Simonetta's model (Simonetta *et al.*, 2019) is available online[14] but we make the following modifications to improve the model's performance: we use binary cross-entropy loss instead of mean error loss, sigmoid rather than ReLU activations, an Adam optimizer with learning rate 1e-4 and dropout to prevent overfitting. We share the re-implemented version online.[15]

As an additional baseline for style and emotion classification, we implemented the ResNet50-based CNN model from S. Kim *et al.* (2020), which represents the state-of-the-art for composer classification, based on the authors' code.[16]

[14]https://github.com/LIMUNIMI/Symbolic-Melody-Identification
[15]https://github.com/sophia1488/symbolic-melody-identification
[16]https://github.com/KimSSung/Deep-Composer-Classification

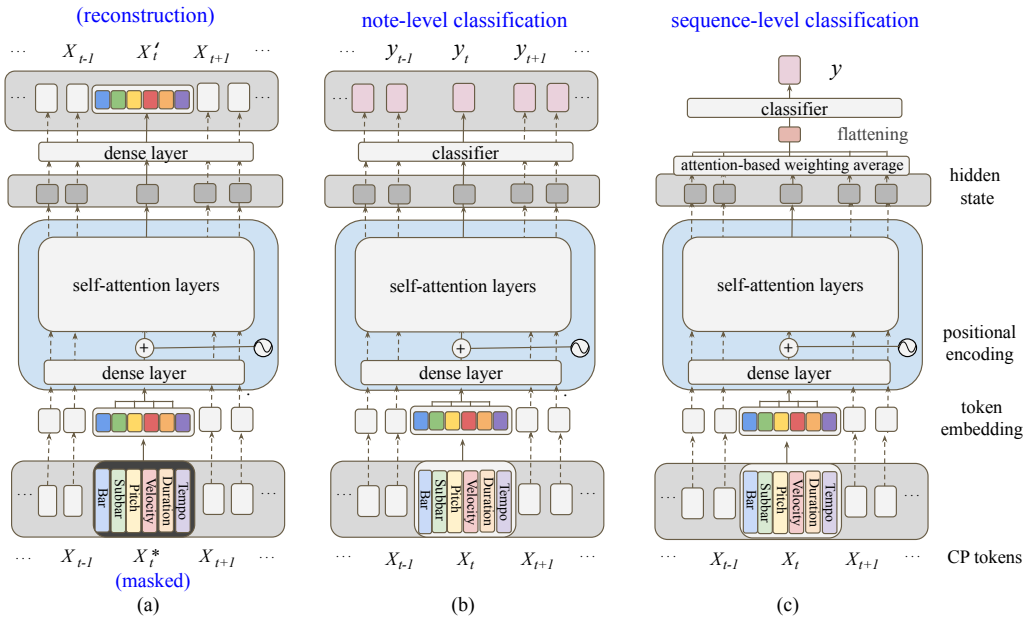



Figure 2: Illustration of the (a) pre-training procedure of our model for a CP sequence, where the model learns to predict (reconstruct) randomly-picked super tokens masked in the input sequence (each consisting of four tokens, as the example one shown in the middle with time step $t$); and (b), (c) the fine-tuning procedure for note-level and sequence-level classification. Apart from the last few output layers, both pre-training and fine-tuning use the same architecture.

## 6 BERT Pre-training and Fine-tuning

We now present our PTM, a pre-trained Transformer encoder with 111 million parameters for piano MIDI music. We adopt as the model backbone the $\text{BERT}_{\text{BASE}}$ model (Devlin *et al.*, 2019), a classic multi-layer bi-directional Transformer encoder with 12 layers of multi-head self-attention, each with 12 heads and the dimension of the hidden space of the self-attention layers being 768. Below, we first describe the pre-training strategy, then the fine-tuning method for the downstream tasks.

### 6.1 Pre-training

For PTMs, an unsupervised or self-supervised, pre-training task is needed to set the objective function for learning. We employ the masked language modelling (MLM) pre-training strategy of BERT, randomly masking 15% tokens of an input sequence and the Transformer will reconstruct these masked tokens from the context of the visible tokens by minimising the cross-entropy loss. As a self-supervised method, MLM needs no labelled data relating to the downstream tasks for pre-training. Following BERT, among all the masked tokens, we replace 80% by `MASK` tokens, 10% by a randomly chosen token and leave the last 10% unchanged. Doing so has the effect of helping mitigate the mismatch between pre-training and fine-tuning as `MASK` tokens do not appear at all during fine-tuning. For REMI, we mask the individual tokens at random. For CP, we mask the super tokens—when we mask a super token, we have to reconstruct all the four tokens composing it by different output heads (W.-Y. Hsiao *et al.*, 2021), as shown in Fig. 2(a).

There are three steps for processing the input token. First, each input token $X_i$ is converted into a token embedding $E_i$ through an embedding layer. Second, it is augmented by addition with a relative positional encoding (Z. Huang *et al.*, 2020) that is related to its time step. Third, it is then fed $E_i$ to the stack of 12 self-attention layers to get a "contextualised" representation known as a *hidden vector* at the output of the self-attention stack. Because of the bi-directional self-attention layers, the hidden vector is contextualized in the sense that it has attended to information from all the other tokens from the same sequence. Finally, the hidden vector of a masked token is fed into a dense layer to predict what the missing super token is. As our network structure is rather standard, we refer readers to the literature (Devlin *et al.*, 2019; Vaswani *et al.*, 2017; S.-L. Wu & Yang, 2023) for details and

the mathematical underpinnings due to space limits. Because the vocabulary sizes for the four token types are different, we weight the training loss associated with tokens of different types in proportion to the corresponding vocabulary size of REMI and CP, to facilitate model training.

We note that the original BERT article also used another self-supervised task called "next sentence prediction" (NSP) (Devlin *et al.*, 2019) together with MLM for pre-training. We do not use NSP for our model since it was later shown to be not that useful (Conneau & Lample, 2019; Joshi *et al.*, 2020; Z. Yang *et al.*, 2019); moreover, NSP requires a clear definition of "sentences", which is not well-defined for our MIDI sequences. As a result, we do not use tokens such as `CLS`, `SEP` and `EOS` used by BERT for making the boundary of the sentences.[17]

### 6.2 Fine-tuning

It has been widely shown in NLP and related fields (Brandes *et al.*, 2022; Y.-S. Chuang *et al.*, 2020; J. Lu *et al.*, 2019; Sun *et al.*, 2019) that, by storing knowledge in huge numbers of parameters and carrying out task-specific fine-tuning, the knowledge implicitly encoded in the parameters of a PTM can be transferred to benefit a variety of downstream tasks (Han *et al.*, 2021). For fine-tuning, we extend the architecture shown in Fig. 2(a) by modifying the last few layers in two different ways, one for each of the two types of downstream tasks.

Fig. 2(b) shows the fine-tuning architecture for note-level classification. While the Transformer uses the hidden vectors to recover the masked tokens during pre-training, it has to predict the label of an input token during fine-tuning, by learning from the labels provided in the training data of the downstream task in a supervised way. To achieve this, we feed the hidden vectors to a stack of dense layers, a ReLU activation layer and finally another dense layer for the output classification, with 10% dropout probability. We note that this classifier design is fairly simple, as we expect much knowledge regarding the downstream task can already be extracted from the preceding self-attention layers.

Fig. 2(c) shows the fine-tuning architecture for sequence-level classification. Being inspired by the Bi-LSTM-Attn model (Z. Lin *et al.*, 2017), we employ an attention-based weighting average mechanism to convert the sequence of 512 hidden vectors for an input sequence to one single vector before feeding it to the classifier layer, which comprises two dense layers. We note that, unlike the baseline models introduced in Section 5, we do not use RNN layers in our models. An alternative approach is to add the `CLS` token to our sequences and simply use its hidden vector as the input to the classifier layer. We do not explore this alternative since we do not have `CLS` tokens.

### 6.3 Implementation Details

Our implementation is based on the open-source library HuggingFace (Wolf *et al.*, 2020). As mentioned in Section 3.1, we use Pop1K7 and ASAP for pre-training and the other three datasets (i.e., POP909, Pianist8 and EMOPIA) for the downstream tasks. From the combination of Pop1K7 and ASAP, we use 85% of them for pre-training as described in Section 6.1 and the rest as the validation set. We train with a batch size of 12 sequences for at most 500 epochs (i.e., around 500K iterations for REMI and 1M iterations for CP), using the AdamW optimizer with learning rate $2e{-}5$ and weight decay rate 0.01. If the validation cross-entropy loss does not improve for 30 consecutive epochs, we stop the training process early. For pre-training, we can improve the validation "cloze" accuracy from 70.4% for REMI to 78.73% for CP. We observe that pre-training using the CP representation converges in 2.5 days on four GeForce GTX 1080-Ti GPUs, which is about 2.5 times faster than the case of REMI. Moreover, a smaller batch size degrades overall performance, including downstream classification accuracy. Because each sequence has 512 super tokens, we have 6,144 super tokens per batch.

For fine-tuning, we create training, validation and test splits for each of the three datasets of the downstream tasks with a 8:1:1 ratio at the piece level (i.e., all the 512-token sequences from the same piece are in the same split). With the same batch size of 12, we fine-tune our pre-trained model for each task independently for at most 10 epochs, stopping early when there is no improvement for three consecutive epochs. Compared to pre-training, fine-tuning is less computationally expensive. All the

[17] `CLS` marks the beginning of a sentence, `SEP` the boundary between two consecutive sentences (useful for the so-called "next sentence prediction task" (Devlin *et al.*, 2019)) and `EOS` the end of a sentence.

Table 2: The testing classification accuracy (in %) of different combinations of MIDI token representations and models for four downstream tasks: three-class melody classification, velocity prediction, style classification and emotion classification. "CNN" represents the ResNet50 model used by S. Kim *et al.* (2020), which only supports sequence-level tasks. "RNN" denotes the baseline models introduced in Section 5, representing the Bi-LSTM model for the first two (note-level) tasks and the Bi-LSTM-Attn model (Z. Lin *et al.*, 2017) for the last two (sequence-level) tasks.

| Token | Model | Melody | Velocity | Style | Emotion |
|---|---|---|---|---|---|
| REMI | CNN (S. Kim *et al.*, 2020) | — | — | 51.35 | 60.00 |
| | RNN (Z. Lin *et al.*, 2017) | 89.96 | 44.56 | 51.97 | 53.46 |
| | Our model (`score`) | 90.97 | 49.02 | 67.19 | 67.74 |
| | Our model (`performance`) | 89.23 | — | 75.30 | 66.18 |
| CP | CNN (S. Kim *et al.*, 2020) | — | — | 67.57 | 60.00 |
| | RNN (Z. Lin *et al.*, 2017) | 88.66 | 43.77 | 60.32 | 54.13 |
| | Our model (`score`) | **96.15** | **52.11** | 67.46 | 64.22 |
| | Our model (`performance`) | 95.83 | — | **81.75** | 70.64 |
| OctupleMIDI | *MusicBERT* (Zeng *et al.*, 2021) | — | — | 37.25 | **77.78** |

results reported in our work can be reproduced with four GeForce GTX 1080-Ti GPUs within 30 minutes.

In our experiments, we used the same pre-trained model parameters to initialise the models for different downstream tasks. During fine-tuning, we fine-tune the parameters of all the layers, including the self-attention and token embedding layers.

## 7 Performance Study

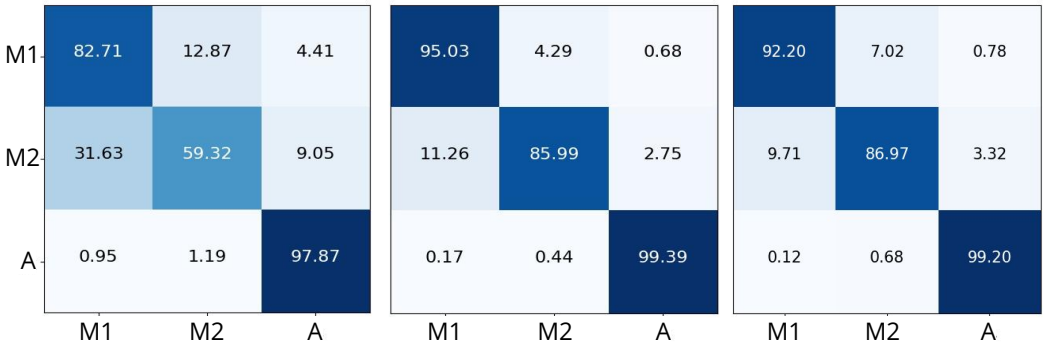
In what follows, we use "our model (`score`)" to indicate the result when MIDI scores are considered and similarly "our model (`performance`)" for MIDI performances. Since MIDI performance contains `velocity` information, we do not evaluate on the velocity prediction task for fairness. We note that, while "our model (`score`)" and "our model (`performance`)" adopt different token representations, we consider it valid to compare their performance as their training and test data are respectively from the same sets of music pieces.

Tab. 2 lists the testing accuracy achieved by the baseline models and the proposed ones for four downstream tasks. We see that "our model (`score`)" outperforms the Bi-LSTM or Bi-LSTM-Attn baselines in all tasks consistently, using either the REMI or CP representation. In particular, the combination of our model (score) and CP, referred to as "our model (score)+CP" hereafter, exhibits the highest accuracy in the two note-level tasks. Additionally, the combination of our model (performance) and CP, denoted as "our model (performance)+CP", achieves the best result in the style classification task, while demonstrating a notable improvement in accuracy compared to REMI for emotion classification. We also observe that our models outperform Bi-LSTM+CP with just 1 or 2 epochs of fine-tuning, validating the strength of PTMs on symbolic-domain music classification tasks.

To facilitate a comprehensive evaluation, we additionally incorporate an officially released version of *MusicBERT* (Zeng *et al.*, 2021) in the sequence-classification tasks. Specifically, we use the model checkpoint *MusicBERT*-small,[18] which is pre-trained on the Lakh MIDI (LMD) dataset (Raffel & Ellis, 2018), which contains about 100K songs.[19] The results show that *MusicBERT* achieves a testing accuracy of 37.25% for style classification and 77.78% for emotion classification. Specifically, in the style classification task, *MusicBERT* exhibits clear signs of overfitting and falls short in performance when compared to our model (81.75%). This outcome can be attributed to the limited size of the Pianist8 dataset, comprising only 411 songs. Conversely, in the emotion classification task,

[18] https://github.com/microsoft/muzic/tree/main/musicbert

[19] There is another implementation named *MusicBERT*-base, which is trained on the Million MIDI Dataset (Zeng *et al.*, 2021), which is ten times larger than LMD.




(a) Bi-LSTM + CP (b) Our model (score) + CP (c) Our model (performance) + CP

Figure 3: Confusion tables (in %) for two models for three-class melody classification, calculated on the test split of $\text{POP909}_{4/4}$. Each row represents the percentage of notes in an actual class while each column represents a predicted class. Notation—"M1": `vocal melody`, "M2": `instrumental melody`, "A": `accompaniment`.

Table 3: Testing metrics (in %) of "our model (performance) +CP" and other baseline methods for the two-class "melody *versus* non-melody" classification task using POP909, viewing `vocal melody` and `instrumental melody` as "melody" and `accompaniment` as "non-melody".

| Model | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Skyline (Chai & Vercoe, 2001) | 79.52 | 81.42 | 56.57 | 66.76 |
| Simonetta *et al.*'s CNN (Simonetta *et al.*, 2019) | 92.08 | 88.95 | 89.30 | 89.13 |
| Our model (performance) + CP | **99.06** | **98.68** | **98.72** | **98.70** |

*MusicBERT* demonstrates impressive performance, surpassing our model (70.64%) by a significant margin. This finding is intriguing and suggests that the application of large-scale pre-training may yield substantial benefits in classifying the emotional content of a MIDI piece.

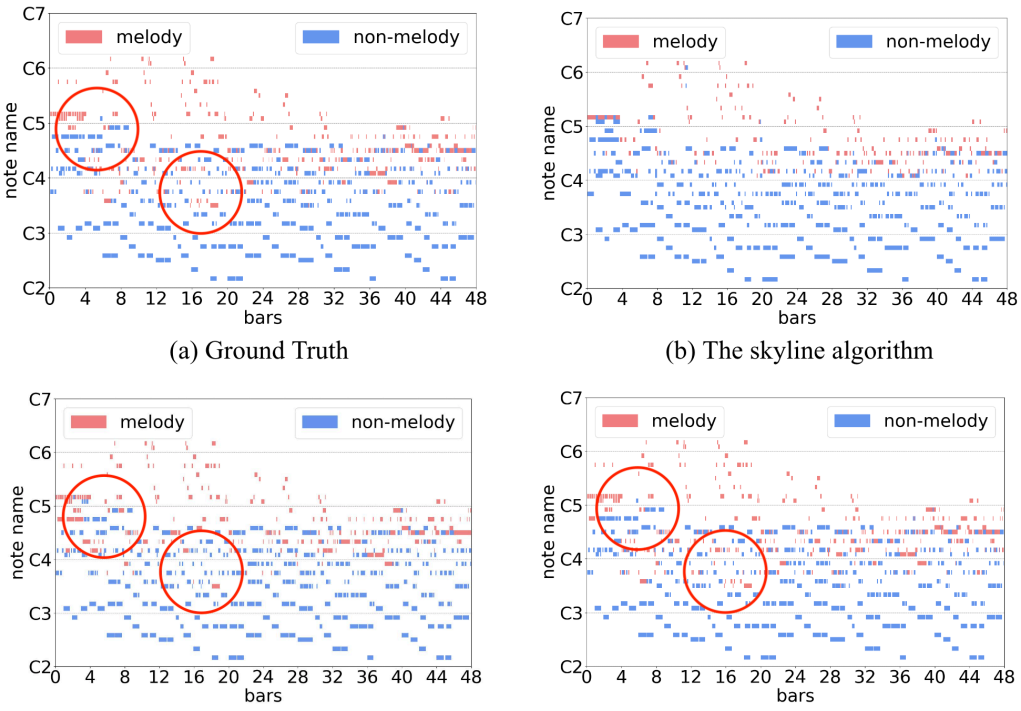
Tab. 2 also shows that the CP token representation tends to outperform the REMI one across different tasks for both the baseline models and the PTM-based models, demonstrating the importance of token representation for music applications. To study whether the accuracy gain comes simply from a longer musical context enjoyed by CP, we also train "our model (`performance`)+CP" with a sequence of length 128, obtaining 95.43, 80.32 and 64.04 accuracies for three-class melody classification, style classification and emotion classification, respectively. We note a sequence of length 512 for REMI contains approximately the same amount of information for a sequence of 147 supertokens for CP. Still, using the CP token representation in general leads to better performance even with less information.

Tab. 2 also shows that "our model (`performance`)+CP" outperforms "our model (`score`)+CP" greatly for the two sequence-level tasks, style classification and emotion classification. This matches our intuition as the two tasks are highly related to performance styles and expressions of the piano pieces.

We take a closer look at the performance of the evaluated models, in particular Bi-LSTM+CP (or Bi-LSTM-Attn+CP), "our model (`score`)+CP" and "our model (`performance`)+CP" in different tasks in what follows.

### 7.1 Melody

Fig. 3 presents the normalised confusion tables for three-class melody classification, illustrating distinct performance characteristics among the models. We note that the baseline exhibits a tendency to conflate vocal melody (M1) and instrumental melody (M2), whereas our model outperforms the RNN-based model, enhancing the overall accuracy by almost 8% (from 88.66% to 96.15%). A closer




(a) Ground Truth

(b) The skyline algorithm

(c) Simonetta et al.'s CNN

(d) Our model (performance) + CP

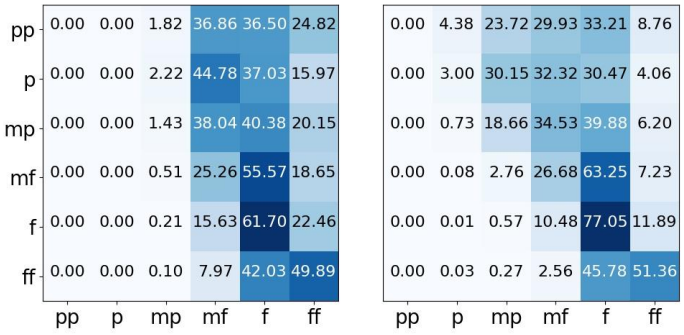
Figure 4: The melody/non-melody classification result for "POP909-596.mid" by (b) "skyline" (Chai & Vercoe, 2001), (c) Simonetta *et al.*'s CNN (Simonetta *et al.*, 2019) and (d) our model (performance) + CP. Directing attention to the red circled region within the pianoroll representation, it is evident that the CNN baseline faces challenges in effectively distinguishing between melody and accompaniment, particularly when note pitches reside within the C4 to C5 range during the initial phase. This is especially pronounced in low-pitch scenarios, where the CNN baseline struggles with accurate classification. In contrast, our model exhibits a notably improved predictive accuracy, closely aligning with the ground truth representation. To supplement this information , the generated melody audio files and pianoroll figures are available in our repository.

examination reveals our model's superior ability to differentiate between vocal and instrumental melodies compared to the RNN baseline with minimal finetuning. This task is particularly challenging given the nature of the POP909 dataset, which exclusively features pop songs sung by humans. Consequently, the separation of vocal and instrumental melodies relies on the criterion of human vocalisation (which is absent in MIDI data), potentially leading to instances where notes between phrases are designated as instrumental melody despite sharing pitch and melodic characteristics with vocal melody.

Interestingly, an intriguing observation emerges as "our model (score) + CP" demonstrates a more effective distinction between vocal and instrumental melodies than "our model (performance) + CP". This phenomenon suggests that even without velocity information, our model can discern segments designated for singing *versus* those serving as preludes, interludes or fills.

Tab. 3 compares our model with the "skyline" algorithm (Chai & Vercoe, 2001) and the CNN-based baselines (Simonetta *et al.*, 2019) for the two-class "melody *versus* non-melody" melody classification task. As the dataset is highly unbalanced (i.e., the melody notes are much fewer than the accompaniment notes), we also report the precision, recall and F1 scores. It turns out that our model greatly outperforms other baselines across all the metrics, reaching 99.04% classification accuracy. A qualitative example demonstrating the superiority of the proposed model over the the skyline algorithm can be found in Fig. 4, using a randomly chosen testing piece from $\text{POP909}_{4/4}$.

Moreover, we have extended the application of our melody extraction model to compositions from the Pianist8 dataset. Given the absence of manual labels for melody notes within this dataset,




(a) Bi-LSTM + CP (b) Our model (score) + CP

Figure 5: Confusion tables (in %) for velocity prediction, calculated on the test split of $\text{POP909}_{4/4}$. Each row represents the percentage of notes in an actual class while each column represents a predicted class.

we encourage readers to evaluate the results by listening to the prediction outputs.[20] We provide three versions of the melody MIDI file for each original song, generated respectively by the skyline algorithm, Simonetta *et al.*'s CNN and "our model (performance) + CP". Taking "Clayderman_Yesterday_Once_More.mid" as an example, the melody generated by the skyline algorithm exhibits stiffness and lacks intricate details, retaining only the treble. The CNN version demonstrates considerable improvement over the skyline algorithm. However, a noticeable intermittent quality persists throughout the entire song, with some cohesive melodies omitted. Our model achieves commendable performance, successfully extracting the majority of the main melody and presenting a discernible melodic progression. It is worth highlighting the efficiency of our model, as it requires less than one hour for fine-tuning under the same hardware conditions that necessitate a full day of training for the CNN baseline on the POP909 training set.

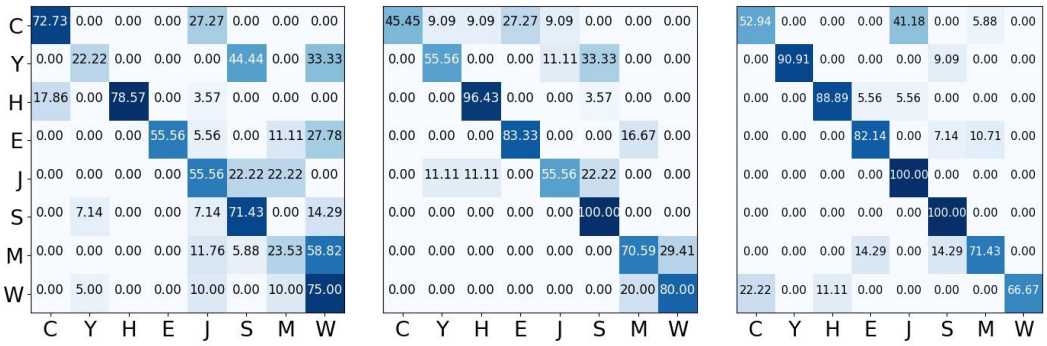
### 7.2 Velocity

Tab. 2 shows that the accuracy on our 6-class velocity classification task is not high, reaching 52.11% at best. This may be due to the fact that velocity is rather subjective, meaning that musicians can perform the same music piece fairly differently. Moreover, we note that the data is highly imbalanced, with the latter three classes (`mf`, `f`, `ff`) taking up nearly 90% of all labelled data. The confusion tables presented in Fig. 5 show that Bi-LSTM tends to classify most of the notes into `f`, the most popular class among the six. This is less the case for our model, but the prediction of p and pp, i.e., the two with the lowest dynamics, remains challenging. For future work, data augmentation is a potential solution to mitigate the impact of data imbalance.

### 7.3 Style

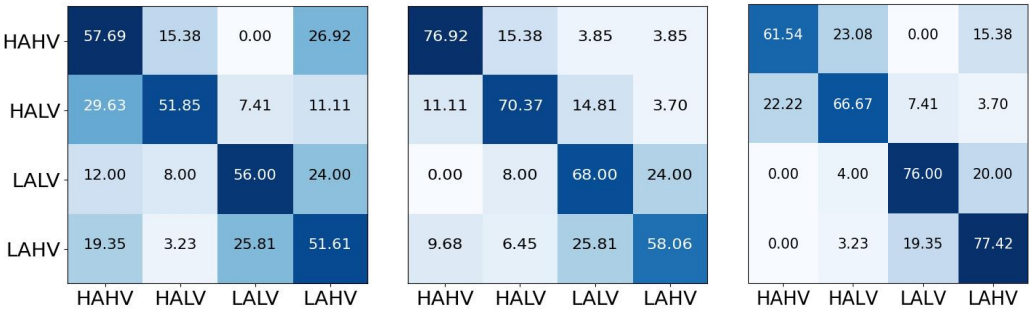
Tab. 2 shows that our model greatly outperforms Bi-LSTM-Attn (Z. Lin *et al.*, 2017) and the CNN baseline (S. Kim *et al.*, 2020) by 10–20% regardless of the token representation taken, reaching 81.75% testing accuracy at best for this 8-class classification problem. In addition, we see a large performance gap between REMI and CP in this task, the largest among the four tasks. Fig. 6 further shows that, both the baseline and "our model (`score`)+CP" confuse artists in similar genres and that our model performs fairly well in recognising Herbie Hancock and Ryuichi Sakamoto. In contrast, by considering velocity and tempo information, "our model (`performance`)+CP" gains lots of precision on classifying songs in pop and contemporary genres, boosting the classification accuracy from 67.46 (score) to 81.75 (performance).

[20] https://github.com/wazenmai/MIDI-BERT/tree/CP/melody_extraction/audio




(a) Bi-LSTM + CP (b) Our model (score) + CP (c) Our model (performance) + CP

Figure 6: Confusion tables (in %) for style classification on the test split of Pianist8. Each row shows the percentage of sequences of a class predicted as another class. Notation—"C": R. Clayderman (pop), "Y": Yiruma (pop), "H": H. Hancock (jazz), "E": L. Einaudi (contemporary), "J": H. Joe (contemporary), "S": R. Sakamoto (contemporary), "M": Bethel Music (religious) and "W": Hillsong Worship (religious).




(a) Bi-LSTM + CP (b) Our model (score) + CP (c) Our model (performance) + CP

Figure 7: Confusion tables for emotion classification; in % of sequences on the test split of EMOPIA. Each row represents the percentage of notes in an actual class while each column represents a predicted class.

### 7.4 Emotion

Tab. 2 shows that our model outscores Bi-LSTM-Attn by around 14% and the CNN baseline (S. Kim *et al.*, 2020) by around 7% in both REMI and CP for this 4-class classification problem, reaching 70.64% testing accuracy at best. There is little performance difference between REMI and CP in this task. Fig. 7 further shows that the evaluated models can fairly easily distinguish between high arousal and low arousal pieces (i.e., "HAHV, HALV" *versus* "LALV, LAHV"), but they have a much harder time along the valence axis (e.g., "HAHV" *versus* "HALV" and "LALV" *versus* "LAHV"). We see less confusion from the result of "our model (`score`)+CP". By considering velocity and tempo, "our model (`performance`)+CP" can further classify variance difference in low-arousal songs, though there is still room for improvement.

## 8 Conclusion

In this article, we presented a large-scale pre-trained model for musical data in the MIDI format. We employed five public-domain piano MIDI datasets for BERT-like masking-based pre-training and evaluated the pre-trained model on four challenging downstream symbolic music classification tasks, most with less than 1K labelled MIDI pieces. Our experiments validate the effectiveness of pre-training for both note-level and sequence-level classification tasks.

This work can be extended in many ways. First, to employ other pre-training strategies or architectures (Han *et al.*, 2021). Second, to employ Transformer models with linear computational complexity (Choromanski *et al.*, 2021; Liutkus *et al.*, 2021), so as to use the whole MIDI pieces (instead of segments) at pre-training.[21] Third, to include other time signatures and increase the amount of non-pop piano scores. Fourth, to extend the corpus and the token representation from single-track piano to multi-track MIDI, like the work done by Zeng *et al.* (2021). Finally, to consider more downstream tasks such as symbolic-domain music segmentation (Hamanaka *et al.*, 2014; Kranenburg, 2020), chord recognition (Harasim *et al.*, 2020), score passage matching (Sutcliffe *et al.*, 2019) and beat tracking (Y.-C. Chuang & Su, 2020). We have released the code publicly, which may hopefully help facilitate such endeavours.

## References


Böck, S., Korzeniowski, F., Schlüter, J., Krebs, F., & Widmer, G. (2016). Madmom: A new python audio and music signal processing library. *Proceedings of the 24th ACM International Conference on Multimedia*, 1174–1178. https://doi.org/10.1145/2964284.2973795

Brandes, N., Ofer, D., Peleg, Y., Rappoport, N., & Linial, M. (2022). ProteinBERT: a universal deep-learning model of protein sequence and function. *Bioinformatics*, *38*(8), 2102–2110. https://doi.org/10.1093/bioinformatics/btac020

Chai, W., & Vercoe, B. (2001). Melody retrieval on the web. *Multimedia Computing and Networking 2002*, *4673*, 226–241. https://doi.org/10.1117/12.449982

Chen, Y.-H., Huang, Y.-S., Hsiao, W.-Y., & Yang, Y.-H. (2020). Automatic composition of guitar tabs by transformers and groove modeling. *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 756–763. https://doi.org/10.5281/zenodo.4245542

Chiu, C.-Y., Su, A. W.-Y., & Yang, Y.-H. (2021). Drum-aware ensemble architecture for improved joint musical beat and downbeat tracking. *IEEE Signal Processing Letters*, *28*, 1100–1104. https://doi.org/10.1109/LSP.2021.3084504

Choi, K., Fazekas, G., Sandler, M. B., & Cho, K. (2018). Transfer Learning for Music Classification and Regression Tasks. *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 141–149. https://doi.org/10.5281/zenodo.1418015

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., & Weller, A. (2021). Rethinking attention with Performers. *9th International Conference on Learning Representations, ICLR 2021*. https://openreview.net/forum?id=Ua6zuk0WRH

Chuang, Y.-C., & Su, L. (2020). Beat and downbeat tracking of symbolic music data using deep recurrent neural networks. *Proc. Asia Pacific Signal and Information Processing Association Annual Summit and Conf.*, 346–352.

Chuang, Y.-S., Liu, C.-L., Lee, H.-Y., & Lee, L.-S. (2020). SpeechBERT: An audio-and-text jointly learned language model for end-to-end spoken question answering. *Proc. INTERSPEECH*, 4168–4172. https://doi.org/10.21437/Interspeech.2020-1570

Conneau, A., & Lample, G. (2019). Cross-lingual language model pretraining. *Proc. Int. Conf. Neural Information Processing Systems*, *32*, 7059–7069. http://papers.nips.cc/paper/8928-cross-lingual-language-model-pretraining

Correa, D. C., & Rodrigues, F. A. (2016). A survey on symbolic data-based music genre classification. *Expert Systems*, *60*(30), 190–210. https://doi.org/10.1016/j.eswa.2016.04.008

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proc. Conf. North American Chapter of the Association for Computational Linguistics*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Dong, H.-W., Chen, K., Dubnov, S., McAuley, J., & Berg-Kirkpatrick, T. (2023). Multitrack music Transformer. *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, 1–5. https://doi.org/10.1109/ICASSP49357.2023.10094628

Ferraro, A., & Lemstrom, K. (2018). On large-scale genre classification in symbolically encoded music by automatic identification of repeating patterns. *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, 34–37. https://doi.org/10.1145/3273024.3273035


[21] We note that the use of linear Transformers for symbolic music generation has been attempted before W.-Y. Hsiao *et al.* (2021).

Foscarin, F., Mcleod, A., Rigaux, P., Jacquemard, F., & Sakai, M. (2020). ASAP: A dataset of aligned scores and performances for piano transcription. *Proceedings of the 21st International Society for Music Information Retrieval*, 534–541. https://doi.org/10.5281/zenodo.4245490

Friberg, A., Bresin, R., & Sundberg, J. (2006). Overview of the KTH rule system for musical performance. *Advances in Cognitive Psychology*, *2*(2-3), 145–161. https://doi.org/10.2478/v10053-008-0052-x

Friberg, A., Schoonderwaldt, E., & Juslin, P. (2007). CUEX: An algorithm for automatic extraction of expressive tone parameters in music performance from acoustic signals. *Acta Acustica united with Acustica*, *93*, 411–420.

Grekow, J., & Raś, Z. W. (2009). Detecting emotions in classical music from MIDI files. *Proc. Int. Symposium on Methodologies for Intelligent Systems*, 261–270. https://doi.org/10.1007/978-3-642-04125-9_29

Gururani, S., Sharma, M., & Lerch, A. (2019). An attention mechanism for musical instrument recognition. *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 83–90. https://doi.org/10.48550/arXiv.1907.04294

Hamanaka, M., Hirata, K., & Tojo, S. (2014). Musical structural analysis database based on GTTM. *Proceedings of the 15th International Society for Music Information Retrieval Conference*, 325–330.

Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., Qiu, J., Yao, Y., Zhang, A., Zhang, L., Han, W., Huang, M., Jin, Q., Lan, Y., Liu, Y., Liu, Z., Lu, Z., Qiu, X., Song, R., … Zhu, J. (2021). Pre-trained models: Past, present and future. *AI Open*, *2*, 225–250. https://doi.org/10.1016/j.aiopen.2021.08.002

Harasim, D., Finkensiep, C., Ericson, P., O'Donnell, T. J., & Rohrmeier, M. (2020). The Jazz Harmony Treebank. *Proceedings of the 21th International Society for Music Information Retrieval Conference*, 207–215. https://doi.org/10.5281/zenodo.4245406

Hawthorne, C., Elsen, E., Song, J., Roberts, A., Simon, I., Raffel, C., Engel, J., Oore, S., & Eck, D. (2018). Onsets and Frames: Dual-Objective Piano Transcription. *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 50–57. https://doi.org/10.5281/zenodo.1492341

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., & Eck, D. (2019). Enabling factorized piano music modeling and generation with the MAESTRO dataset. *7th International Conference on Learning Representations, ICLR 2019*. https://openreview.net/forum?id=r1lYRjC9F7

Hsiao, W.-Y., Liu, J.-Y., Yeh, Y.-C., & Yang, Y.-H. (2021). Compound Word Transformer: Learning to compose full-song music over dynamic directed hypergraphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*, 178–186. https://doi.org/10.1609/aaai.v35i1.16091

Hsiao, Y.-W., & Su., L. (2021). Learning note-to-note affinity for voice segregation and melody line identification of symbolic music data. *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, 285–292. https://doi.org/10.5281/zenodo.5624479

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N. M., Simon, I., Hawthorne, C., Dai, A. M., Hoffman, M. D., Dinculescu, M., & Eck, D. (2019). Music Transformer: Generating music with long-term structure. *7th International Conference on Learning Representations, ICLR 2019*. https://openreview.net/forum?id=rJe4ShAcF7

Huang, Y.-S., & Yang, Y.-H. (2020). Pop Music Transformer: Beat-based modeling and generation of expressive pop piano compositions. *Proc. ACM Multimedia*, 1180–1188. https://doi.org/10.1145/3394171.3413671

Huang, Z., Liang, D., Xu, P., & Xiang, B. (2020). Improve Transformer models with better relative position embeddings. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 3327–3335. https://doi.org/10.18653/v1/2020.findings-emnlp.298

Hung, H.-T., Ching, J., Doh, S., Kim, N., Nam, J., & Yang, Y.-H. (2021). EMOPIA: A multi-modal pop piano dataset for emotion recognition and emotion-based music generation. *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, 318–325. https://doi.org/10.5281/zenodo.5090631

Jeong, D., Kwon, T., Kim, Y., Lee, K., & Nam, J. (2019). VirtuosoNet: A hierarchical RNN-based system for modeling expressive piano performance. *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 908–915. https://doi.org/10.5281/zenodo.3527962

Jeong, D., Kwon, T., Kim, Y., & Nam, J. (2019). Graph neural network for music score data and modeling expressive piano performance. *Proc. Int. Conf. Machine Learning*, 3060–3070.

Jiang, Z., & Dannenberg, R. B. (2019). Melody identification in standard MIDI files. *Proc. Sound and Music Computing Conf.*, 65–71. https://doi.org/10.5281/zenodo.3249256

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., & Levy, O. (2020). SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, *8*, 64–77. https://doi.org/10.1162/tacl_a_00300

Kim, J., Urbano, J., Liem, C., & Hanjalic, A. (2019). One deep music representation to rule them all? A comparative analysis of different representation learning strategies. *Neural Computing & Applications*, *32*, 1067–1093. https://doi.org/10.1007/s00521-019-04076-1

Kim, S., Lee, H., Park, S., Lee, J., & Choi, K. (2020). Deep composer classification using symbolic representation. *CoRR*, *abs/2010.00823*. https://arxiv.org/abs/2010.00823

Kong, Q., Choi, K., & Wang, Y. (2020). Large-scale MIDI-based composer classification. *arXiv preprint arXiv:2010.14805*.

Kong, Q., Li, B., Song, X., Wan, Y., & Wang, Y. (2021). High-resolution piano transcription with pedals by regressing onsets and offsets times. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, *29*, 3707–3717. https://doi.org/10.1109/TASLP.2021.3121991

Kranenburg, P. V. (2020). Rule mining for local boundary detection in melodies. *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 271–278. https://doi.org/10.5281/zenodo.4245422

Lin, Y., Chen, X., & Yang, D. (2013). Exploration of music emotion recognition based on MIDI. *Proceedings of the 14th International Society for Music Information Retrieval Conference*, 221–226. https://doi.org/10.5281/zenodo.1416604

Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017). A structured self-attentive sentence embedding. *5th International Conference on Learning Representations, ICLR 2017*. https://openreview.net/forum?id=BJC_jUqxe

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

Liutkus, A., Cífka, O., Wu, S.-L., Şimşekli, U., Yang, Y.-H., & Richard, G. (2021). Relative positional encoding for Transformers with linear complexity. *Proceedings of the 38th International Conference on Machine Learning*, *139*, 7067–7079. http://proceedings.mlr.press/v139/liutkus21a.html

Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Proc. Int. Conf. Neural Information Processing Systems*, 13–23. https://doi.org/10.48550/arXiv.1908.02265

Lu, P., Xu, X., Kang, C., Yu, B., Xing, C., Tan, X., & Bian, J. (2023). MuseCoco: Generating symbolic music from text. *arXiv preprint arXiv:2306.00110*.

Madsen, S. T., & Widmer, G. (2007). A complexity-based approach to melody track identification in MIDI files. *Proc. Int. Workshop on Artificial Intelligence and Music*.

Oore, S., Simon, I., Dieleman, S., Eck, D., & Simonyan, K. (2018). This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, *32*, 955–967. https://doi.org/10.1007/s00521-018-3758-9

Panda, R., Malheiro, R., & Paiva, R. P. (2018). Musical texture and expressivity features for music emotion recognition. *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 383–391. https://doi.org/10.5281/zenodo.1492431

Panda, R., Malheiro, R., Rocha, B., Oliveira, A., & Paiva, R. P. (2013). Multi-modal music emotion recognition: A new dataset, methodology and comparative analysis. *Proc. Int. Symposium on Computer Music Multidisciplinary Research*, 570–582. https://hdl.handle.net/10316/94095

Pati, A., Lerch, A., & Hadjeres, G. (2019). Learning to traverse latent spaces for musical score inpainting. *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 343–351. https://doi.org/10.48550/arXiv.1907.01164

Payne, C. (2019). "MuseNet." *OpenAI, 25 Apr. 2019*. https://openai.com/research/musenet

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

Raffel, C., & Ellis, D. P. W. (2018). Extracting ground-truth information from midi files: A MIDIfesto. *Proceedings of the 17th International Society for Music Information Retrieval Conference*, 796–802. https://doi.org/10.5281/zenodo.1418233

Russell, J. (1980). A circumplex model of affect. *Journal of Personality and Social Psychology*, *39*, 1161–1178. https://doi.org/10.1037/h0077714

Salamon, J., & Gomez, E. (2012). Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(6), 1759–1770. https://doi.org/10.1109/TASL.2012.2188515

Shih, Y.-J., Wu, S.-L., Zalkow, F., Muller, M., & Yang, Y.-H. (2022). Theme Transformer: Symbolic music generation with theme-conditioned Transformer. *IEEE Transactions on Multimedia*, 1–1. https://doi.org/10.1109/TMM.2022.3161851

Simonetta, F., Chacón, C. E. C., Ntalampiras, S., & Widmer, G. (2019). A convolutional approach to melody line identification in symbolic scores. *Proceedings of the 20th International Society for Music Information Retrieval Conference.*, 924–931. https://doi.org/10.5281/zenodo.3527966

Sun, C., Myers, A., Vondrick, C., Murphy, K., & Schmid, C. (2019). VideoBERT: A joint model for video and language representation learning. *Proc. IEEE/CVF International Conference on Computer Vision*, 7464–7473. https://doi.org/10.1109/ICCV.2019.00756

Sutcliffe, R., Hovy, E., Collins, T., Wan, S., Crawford, T., & Root, D. L. (2019). Searching for musical features using natural language queries: The C@merata evaluations at MediaEval. *Language Resources and Evaluation*, *53*(1), 87–140. https://doi.org/10.1007/s10579-018-9422-2

Tsai, T., & Ji, K. (2020). Composer style classification of piano sheet music images using language model pretraining. *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 176–183. https://doi.org/10.48550/arXiv.2007.14587

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 5998–6008. https://doi.org/10.48550/arXiv.1706.03762

Waite, E., Eck, D., Roberts, A., & Abolafia, D. (2016). Project Magenta: Generating long-term structure in songs and stories. *Google Brain Blog*.

Wang, Z., Chen, K., Jiang, J., Zhang, Y., Xu, M., Dai, S., Bin, G., & Xia, G. (2020). POP909: A pop-song dataset for music arrangement generation. *Proceedings of 21st International Conference on Music Information Retrieval*, 38–45. https://doi.org/10.48550/arXiv.2008.07142

Widmer, G. (1994). The synergy of music theory and AI: Learning multi-level expressive interpretation. *Proceedings of the 12th National Conference on Artificial Intelligence*, *12*, 114–119. http://www.aaai.org/Library/AAAI/1994/aaai94-018.php

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., … Rush, A. (2020). Transformers: State-of-the-art natural language processing. *Proc. Conf. Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45.

Wu, H.-H., Kao, C.-C., Tang, Q., Sun, M., McFee, B., Bello, J. P., & Wang, C. (2021). Multi-task self-supervised pre-training for music classification. *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, 556–560. https://doi.org/10.1109/ICASSP39728.2021.9414405

Wu, S.-L., & Yang, Y.-H. (2020). The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures. *Proc. Int. Soc. Music Information Retrieval Conf.*, 142–149. https://doi.org/10.48550/arXiv.2008.01307

Wu, S.-L., & Yang, Y.-H. (2023). MuseMorphose: Full-song and fine-grained piano music style transfer with just one Transformer VAE. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *31*, 1953–1967. https://doi.org/10.1109/TASLP.2023.3270726

Yang, Y.-H., & Chen, H. H. (2011). *Music Emotion Recognition* (1st ed.). CRC Press.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 5753–5763. https://doi.org/10.48550/arXiv.1906.08237

Zeng, M., Tan, X., Wang, R., Ju, Z., Qin, T., & Liu, T.-Y. (2021). MusicBERT: Symbolic music understanding with large-scale pre-training. *Findings of the Association for Computational Linguistics*, 791–800. https://doi.org/10.18653/v1/2021.findings-acl.70