AI-powered Support System for Music Composition

Phi-Hung Ngo[1,2][0009−0002−8088−6528],
Quoc-Vuong Pham[1,2][0009−0008−6737−9795], and
Duy-Hoang Tran[1,2][0000−0001−9669−4657]

[1] Faculty of Information Technology, University of Science, Ho Chi Minh city, Vietnam
[2] Vietnam National University, Ho Chi Minh city, Vietnam

Abstract. Text-to-music generation is a task that creates musical compositions from textual descriptions, allowing users to easily and intuitively generate music using natural language. There are two main approaches: text-to-audio-music and text-to-symbolic-music; the latter has the advantage of easy editing of musical elements. This paper proposes a text-to-MIDI generation system that uses musical attributes as a bridge. In contrast to a large single-stage model, two compact models are proposed: text-to-attribute understanding and attribute-to-MIDI generation. For text-to-attribute, BERT is used to extract musical attribute values from plain text in multiple languages. For attribute-to-MIDI, GPT-2 combined with Low-Rank Adaptation is used to optimize training cost and efficiency. Furthermore, a post-processing method based on music theory principles is proposed to ensure the accuracy and integrity of the generated music. With approximately 200 million parameters, the proposed model shows a 9.84% improvement in Average Sample-wise Accuracy compared to the MuseCoCo model trained on the same dataset. Our source code is available at this link[3].

Keywords: AI music generator · text-to-midi · symbolic music · large language mode.

1 Introduction

Text-to-music generation is the task of creating musical compositions based on textual descriptions. It allows users to generate music more easily and intuitively by using natural language as the interface. There are two main approaches to this problem: text-to-audio-music [3, 8, 1, 6] and text-to-symbolic-music [13, 19, 17, 11]. The approach of generating audio music from textual descriptions often has limitations: 1) it does not allow detailed editing, which leads to a lack of creative freedom in using the generated music for composition; 2) it poses difficulties in validating musical features such as instruments, time signature, and pitch range in the generated music; and 3) it requires a large amount of

[3] https://github.com/HCMUS-Thesis-AI-for-Music-Composition/Brainstorming

training data as well as expensive model training costs. In contrast, symbolic music is defined as music stored in a notation-based format, such as the Musical Instrument Digital Interface (MIDI), which contains explicit information about note onsets and pitch on individual tracks for different instruments [10]. MIDI offers flexibility and efficiency for user-editable modifications after the music has been created, with smaller data sizes and easier extraction of musical features compared to audio music. This allows for more detailed and precise control over musical attributes, enabling the creation of longer and more customized compositions. In addition to easily meeting user requirements, the MIDI music generation model also requires fewer computational resources during both training and inference compared to audio music generation model. This motivates us to choose the text-to-symbolic-music approach, especially text-to-MIDI.

Beyond the audio-music versus symbolic-music issue, a common challenge in text-to-music models is helping the model understand subjective descriptions provided by users, such as *"A piece of music with a moderate tempo that feels relaxing."*. Converting these subjective descriptions into specific technical parameters such as tempo, key, or time signature is challenging due to the ambiguity and inconsistency of textual descriptions. This complexity makes it difficult to control and adapt the model to specific user desires, especially for those without deep technical knowledge of music. To address this issue, we have adopted an architecture of MuseCoCo [11] that consists of two compact models instead of a large single-stage model. Using musical attributes as a bridge, the task of generating symbolic music from text descriptions is divided into two stages: understanding musical attributes from input text (text-to-attribute) and generating music from these attributes (attribute-to-music). This approach allows each model to be more streamlined while maintaining performance, optimizing resources, and increasing processing efficiency. Our research includes the following proposed methods and contributions:

- For the text-to-attribute model, we applied prompt engineering techniques to increase the variety of text descriptions with different contexts provided by users. We reuse 4,815 English music description templates from MusicCoCo and use prompt engineering to enrich the training data, increasing the number of training templates to 14,900. We also develop a multilingual model that supports both English and Vietnamese.
- For the attribute-to-music model, we apply the large language model GPT-2 [14] combined with the Low-Rank Adaptation (LoRA) technique [7] to optimize resources and training costs. With only 200 million parameters, this solution yields 12% better accuracy based on the Average Sample-wise Accuracy (ASA) metric [11] compared to the MuseCoCo[11] model trained on the same dataset.
- In addition, we proposed a post-processing technique to validate the correctness of the output from the attribute-to-music model and used rule-based music interpolation techniques to correct errors in the generated music segments.

2 Related Work

The field of text-to-music has developed rapidly in recent years, and a wide range of diverse and rich research has emerged. The research can be divided into two main approaches: text-to-audio-music and text-to-symbolic-music generation.

In the first approach, scientists and engineers have used large-scale language models and architectures, such as Transformers [15] and Generative Adversarial Networks [5], to automatically transform text descriptions into audio music. Notable work in this area includes MuLan [8], an advanced model capable of efficiently and creatively generating music from textual descriptions. MuLan uses machine learning and natural language processing techniques to transform emotional, stylistic, and rhythmic descriptions into complex musical segments. Another important model is MusicLM [1], developed by Google Research, which produces high-quality music based on textual descriptions. In addition, MUGEN [6] is a model designed to understand and generate sound for video games, using video input of a game scene and descriptive text. However, the approach of generating audio music from text often has limitations, such as the inability to allow detailed editing, which leads to a lack of creative freedom in using the generated music for composition. It also requires bulky training data and poses difficulties in validating musical features, such as instruments, time signature, and pitch range, in the generated music.

The second approach focuses on generating symbolic music that can be more easily interpreted and manipulated by both humans and machines. BUTTER [19] is a framework for learning music-sentence representations that introduces methods for disentangling musical representations and aligning them across modalities. It initially generates music in ABC notation from text descriptions that include four musical keywords: key, meter, style, and others. However, its capabilities are limited by the folk song datasets and a limited number of musical factors, preventing it from generating a wide range of symbolic music. The work in [17] explores the use of publicly available pre-trained natural language processing checkpoints, such as BERT, GPT-2, and BART, in the task of text-to-symbolic-music generation. Despite being fine-tuned with over 200,000 pairs of text and ABC notation music, the model struggles to accurately align musical attribute values in the text with the generated music and is limited to producing music with solo tracks only. MuseCoCo[11] uses a two-model architecture to generate symbolic music from textual descriptions with high accuracy. The approach breaks down the task into two stages: the text-to-attribute understanding stage and the attribute-to-music generation stage. We refer to MuseCoCo[11]'s architecture, but our research has several differences: 1) enriching the training data, 2) applying a more lightweight training model while ensuring performance, and 3) proposing a post-processing technique to correct musically incorrect segments in the generated music.

3 Proposed Method

3.1 Overview

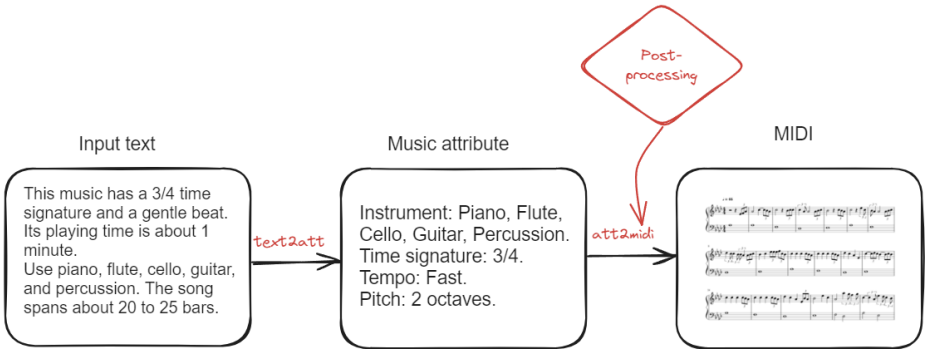

Fig. 1. Overview architecture

The proposed architecture is described in Figure 1, which can be divided into three main stages: understanding the text description, generating symbolic music, and post-processing the generated music.

With the corresponding **text2att** model, the first stage focuses on the task of identifying musical attributes based on the input music description text by label identification and classification. These labels represent either qualitative attributes, such as fast, slow, and instruments used, or quantitative attributes, such as pitch range, number of octaves, and length of music segment. The data format for the **text2att** model is music description templates in multiple languages, containing placeholders for the identified qualitative and quantitative labels. Here is an example of an English template:

"The song has a fast tempo and a [TIME_SIGNATURE] time signature. It is bright at its start but then turns dark. The tune is played with [INSTRUMENTS]."

The second stage, with the corresponding **att2midi** model, focuses on the task of music generation. The input consists of the labels identified in the first stage. A prompt corresponding to these labels is created to help the model determine the characteristics of the music segment to be generated. The data format for the **att2midi** model is "source - target" pairs, in other words, "command - music". Each `command` contains pairs of metadata abbreviations (see Table 1) and their corresponding values; `music` is MIDI music data tokenized into text-based music using REMI [9] from MuseCoCo [11].

The final stage focuses on post-processing the music generated in the second stage. Although the generative model has been carefully trained, there is always a possibility that the output data may not be convertible into the desired MIDI format. Therefore, we propose validation and correction algorithms to ensure that the results of the model can be converted into the correct data format.

Table 1. Metadata corresponding to the data fields in the command

Symbol	Metadata
I1s2	Instrument
I4	Main Instrument
R3	Rhythm Intensity
B1s1	Bar
TS1s1	Time Signature
K1	Key
T1s1	Tempo
P4	Pitch Range
TM1	Time

3.2 Text-to-attribute model

We use BERT [4], an encoder-type model for text feature extraction, along with the architecture of MusicBERT [18], to build a model that predicts and classifies musical attributes from descriptive text (see Figure 2). We developed a customized variant for the task of classifying different musical attributes. These attributes include both qualitative aspects, such as the presence of instruments, and quantitative aspects, such as the duration and tempo of the music. The model adds [CLS_i] tokens, which are the information of the labels mentioned in Table 1. After passing through BERT to extract features, the output is logits, which are then passed through a Softmax layer to classify the labels.

3.3 Attribute-to-MIDI model

We use the GPT-2 model [14], which is optimized by applying the LoRA technique [7] to reduce the number of parameters needed for training. The goal is to speed up the training process while maintaining performance. The music dataset has its own unique vocabulary and data type, including attributes such as instrument, pitch, time signature, bar, tempo, etc., which are precisely encoded for the model to understand and learn (see Figure 3). In addition, we rely on the REMI [9] tokenizer to tokenize the data. The model is then fine-tuned on the dataset.

3.4 Post-Processing

By treating MIDI data as a form of time-series data, we examine each position for the times at which MIDI events occur, such as tempo changes, time signature changes, or the appearance of new notes. We check whether each event follows the predefined rules. These rules are mapped from the REMI-based rule set of MuseCoCo (see MuseCoCo's text-based music attribute arrangement rules in Table 2). After examining all the events, it is possible that there are false events.

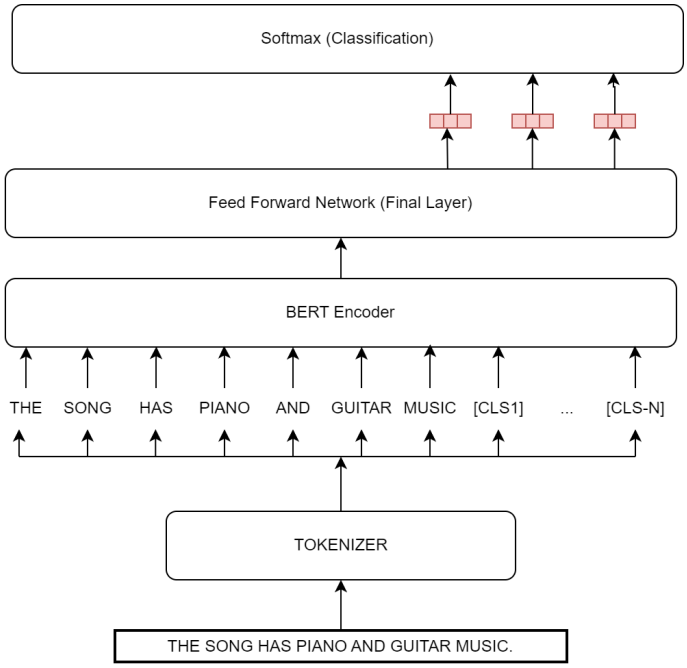

Fig. 2. The text2att model architecture using BERT

Fig. 3. The music generation model in "command - music" format

If these are ignored, the coherence of the generated music may be disturbed. To solve this problem, we perform two main steps (see the illustration in Figure 4):

1. **Restoring structure:** For structurally broken events, we replace them with the structure of the closest similar and correct event. If the entire dataset does not contain a correct event, a default structure corresponding to the broken event, as specified in our data specifications, is applied.
2. **Restoring values:** The values of the structural components in the false event are filled into the new structure to maintain the most accurate information possible. If a structural component is missing a value, we will select a value to fill in according to the following priority order: the value of the clos-

est corresponding component in the entire music piece; a predefined default value[4].

Table 2. Text-based music attribute arrangement rules of MuseCoCo[11]

Abbreviation	Attribute	Next Abbreviation
s	Time Signature	b, o.
o	Position	t.
t	Tempo	i.
i	Instrument	p.
p	Pitch	d.
d	Duration	v.
v	Velocity	i, b, p, o.
b	Bar	s.

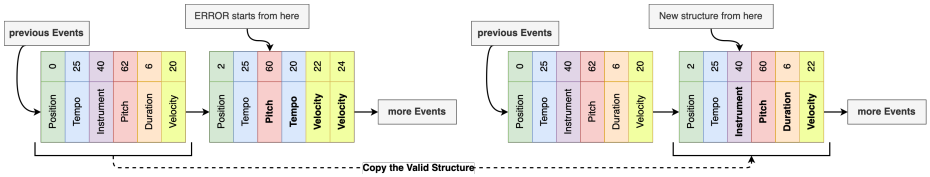

Fig. 4. Illustration of the post-processing process

This algorithm allows us to minimize errors and ensure that the final music generated is always as valid and consistent as possible.

4 Experiments

4.1 Dataset

Text2att dataset. The goal of the data preparation process is music description templates in English and Vietnamese. To meet the demand for English training data, we reuse 4,815 templates created by the authors of the MuseCoCo paper. In addition, we used prompt engineering with ChatGPT [12] to enrich the data,

[4] For pitch, the replacement value can be chosen by: Determining the key signature of the non-false note set, then filling in the pitch value by averaging the pitch values of the notes before and after the false note (rounded to the nearest pitch value in the key).

bringing the total number of English templates to 14,900. We propose a translation technique to obtain 14,900 corresponding Vietnamese templates. The list of labels in the template sentences and their corresponding values are described in Table 3.

Table 3. Names and corresponding values of labels in each template sentence from the MuseCoCo paper[11] (a more compact version based on the labels we use)

Label Name	**Value**
Instrument	28 instruments: piano, keyboard, percussion, organ, guitar, bass, violin, viola, cello, harp, strings, voice, trumpet, trombone, tuba, horn, brass, sax, oboe, bassoon, clarinet, piccolo, flute, pipe, synthesizer, ethnic instrument, sound effect, drum. Each instrument: 0: Played, 1: Not played, 2: NA
Pitch	Range: 0-11: octaves, 12: NA.
Rhythm Danceability	0: danceable, 1: not danceable, 2: NA.
Bar	0: 1-4 bars, 1: 5-8 bars, 2: 9-12 bars, 3: 13-16 bars, 4: NA.
Time Signature	0: 4/4, 1: 2/4, 2: 3/4, 3: 1/4, 4: 6/8, 5: 3/8, 6: other time signatures, 7: NA.
Key	0: major, 1: minor, 2: NA.
Tempo	0: slow (<=76 BPM), 1: medium (76-120 BPM), 2: fast (>=120 BPM), 3: NA.

Att2midi dataset. The output of the music data preparation process is the "command - music" pairs as described in Section 3.1. In addition to reusing 300 publicly available samples from MuseCoCo, we collected data from Hooktheory [2] — a website specializing in providing e-books, articles, statistics, educational software on music theory, as well as musical notation and harmony information for over 40,000 songs worldwide. After the data collection process, we convert the collected data into 29,000 music segments in MIDI file format. These MIDI files are converted into "command - music" format through the following three steps:

1. First, the midi_data_extractor library from MuseCoCo is used to extract metadata information such as tempo, musical speed, and instruments from a MIDI file. This metadata is then mapped to a given music dictionary to determine details such as the duration of the music piece, tempo, instruments played, and the others (see metadata fields in Table 1).

2. The command part is created by assembling this metadata into prompt sentences appropriate for the piece of music. This serves as the command for music generation.
3. The MIDI file is then converted to a text format using REMI tokenization method of MuseCoCo. This represents the music part in the target data format.

4.2 Training configuration

Text2att model. The hyperparameters used in the BERT model training process for this stage are as follows. The number of epochs is 100, which determines the number of times the model is trained on the entire dataset. The batch size is 32, which is the size of each batch of data fed into the model during each training step. The optimizer used is AdamW, which updates the parameters of the model during training. The learning rate is 2×10^{-5}, which determines the extent to which determines how much the weights of the model are adjusted after each training step. The maximum sequence length is 128, and longer sequences will be truncated. The number of warm-up steps is 2,000, which is the number of initial steps during which the learning rate gradually increases to its maximum value. The dropout rate is 0.1, which is used during training to prevent overfitting. The number of gradient accumulation steps is 2, which is the number of gradient accumulation steps before updating the model weights. The weight decay coefficient is 0.01, which helps adjust the rate of weight decay to avoid overfitting.

Att2midi model. In the process of training the GPT-2 model, we perform detailed steps from data preparation, tokenization, model configuration, application of the LoRA technique, to training and evaluation of the results. The data was tokenized with parameters such as maximum length and padding, and then divided into training, validation, and test sets in an 80:10:10 ratio. The GPT-2 model was configured with 20 transformer layers, 16 heads in multi-head attention, an embedding size of 1,024, a vocabulary size of 1,253, and a maximum positional embedding size of 2,048. The LoRA technique was used to reduce the number of parameters to be trained, thus speeding up the training process. Specific parameters for LoRA include $r = 16$, $\alpha = 12$, $dropout = 0.1$, and target modules such as c_proj, c_attn, wte, lm_head. The training and evaluation process was run with the following parameters 10 epochs, batch size 8, AdamW optimizer, learning rate 2×10^{-4}, maximum sequence length 2,048, warm-up steps 2,000, dropout rate 0.1, gradient accumulation steps 2, and weight decay 0.01. The model was trained using the Trainer class of transformers with a total number of parameters of **203 million**. The number of trainable parameters was approximately **3.5 million**.

4.3 Evaluation setting and baseline

Evaluation dataset. We performed individual evaluations for each model. To evaluate the models, we created standard test sets of 1,000 samples for the text2att model and 600 samples for the att2music model. For the text2att model, we evaluated by matching labels from the test set with those predicted by the model, similar to classification models. For the att2midi model, we had to go through an additional step of extracting metadata from the generated music to match the command labels from the test set.

Evaluation metric. We specifically evaluated each type of label as a classification model using the formula:

$$\text{accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}}$$

To evaluate the models objectively, we used a measure called Average Samplewise Accuracy (ASA) [11], which is calculated by determining the proportion of attributes correctly predicted in each sample and then averaging the prediction accuracy over the entire test set.

In addition, several other evaluation criteria were used, including:

1. Average pitch count [16]: This measure is used to determine the distribution of pitches among the 128 pitches supported by the MIDI file. The formula is:

$$\text{average\_pitch\_count} = \frac{\text{number of occurrences of the considered pitch}}{\text{total number of notes in the MIDI file}}$$

2. Average pitch class count [16]: This measure is used to determine the distribution of pitches among the 12 basic musical notes. The formula is:

$$\text{average\_pitch\_class\_count} = \frac{\text{number of occurrences}}{\text{total number of notes in the MIDI file}},$$

   where the number of occurrences is calculated based on the number of notes whose pitches are congruent to the considered pitch modulo 12.
3. Subjective evaluation: Assessment based on listening and perception.

Baseline. In this paper, we compared our method with the MuseCoCo work [11]. To ensure fairness, we reconstructed the MuseCoCo model using the original source code and configured the system with similar parameter sizes, training server settings, training data, output data formats, etc. to those of our proposed models. The comparison results were compiled after training and prediction with the best checkpoint.

4.4 Results

When evaluating the **text2att** model, for music attributes related to instrument types, we focused only on the common instruments in the training data. The results were measured and showed very high accuracy (see Table 4).

Specifically, instruments such as accordion, brass, celesta, choir, guitar, harmonica, organ, piano, synth, viola, violin, and voice, all achieved high accuracy which is equal or greater than 0.90 in both English (ENG) and Vietnamese (VIE). This indicates that our feature extraction model performs well and is consistent across these instrument types, regardless of the evaluation language.

Table 4. Accuracy of each instrument for the Instrument attribute

Instrument	Accuracy (ENG)	Accuracy (VIE)
accordion	0.94	0.93
brass	0.98	0.96
celesta	0.91	0.92
choir	0.95	0.97
guitar	0.99	0.93
harmonica	0.97	0.94
organ	0.90	0.91
piano	0.96	0.95
synth	0.92	0.94
viola	0.91	0.90
violin	0.93	0.92
voice	0.95	0.96

For other music attributes, the results indicate very high accuracy, with attributes such as **Time Signature** and **Pitch Range** achieving the highest accuracies of 0.94 (English - ENG) and 0.94 (Vietnamese - VIE), respectively. This shows that the model has a strong capability to classify rhythmic and pitch range factors effectively. Attributes such as **Rhythm Danceability** and **Tempo** have slightly lower accuracies, ranging from 0.85 to 0.93, but still show high and reliable accuracy (see Table 5).

Table 5. Accuracy of other attributes

Attribute	Accuracy (ENG)	Accuracy (VIE)
Rhythm Danceability	0.85	0.87
Bar	0.91	0.89
Time Signature	0.94	0.92
Key	0.88	0.90
Tempo	0.93	0.85
Pitch Range	0.90	0.94

Comparison with MuseCoCo. The accuracy of the music attributes between our proposed model and the reconstructed MuseCoCo model, as shown in Table 6, demonstrates that the LoRA GPT-2 model outperforms MuseCoCo in most metrics. Specifically, in the Rhythm Danceability metric, LoRA GPT-2 achieves very high accuracies of 0.96. However, for the Time Signature metric, MuseCoCo shows a higher result with a value of 0.72 compared to 0.60. Overall, our model performs 9.84% better performance on the ASA metric than the reconstructed MuseCoCo model.

Table 6. Comparison of results between the two models

Metric	LoRA GPT-2	Reconstructed MuseCoCo
ASA	**0.67**	0.61
Instrument	0.68	0.60
Pitch Range	0.68	0.52
Rhythm Danceability	0.96	0.89
Bar	0.51	0.42
Time Signature	0.60	0.72
Key	0.57	0.51
Tempo	0.69	0.61

Evaluation based on musical note pitch. The visualization results of pitch metrics are shown in Figures 5, 6, 7, and 8. The values of pitch class and pitch are not continuous because the prompts we used focus on two main keys: C major and A minor. Pitch refers to all MIDI notes that appear in the music pieces. Pitch class refers to the basic pitches present in the music pieces (pitch class = pitch mod 12). When we calculated the average number of notes used in each music segment, the LoRA GPT-2 model yielded approximately 303 notes, while the reconstructed MuseCoCo model yielded approximately 139 notes (about 0.5 times the result of the other model). This characteristic is also clearly reflected in the four figures we just mentioned. The pitch class distribution charts do not show significant differences between the two models. However, the pitch distribution charts indicate that our model has a smoother distribution, which is closer to a normal distribution compared to the other model. In contrast, the distribution of the reconstructed MuseCoCo model shows outliers at pitches 88 and 43. The average count of pitch values tends to be higher for higher pitch values compared to lower pitch values. This is because: 1) medium-high pitch values are often where chords (multiple notes at the same time) are concentrated, 2) higher and medium-high pitch values are where melodies (usually one note at a time) are concentrated, and 3) lower pitch values are where bass notes (typically one note at a time) are concentrated.

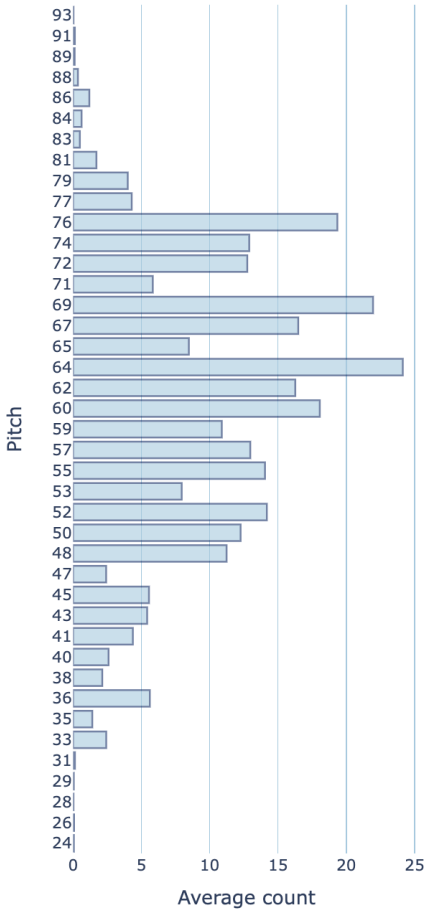

Fig. 5. Average occurrence of musical pitch values (LoRA GPT-2)

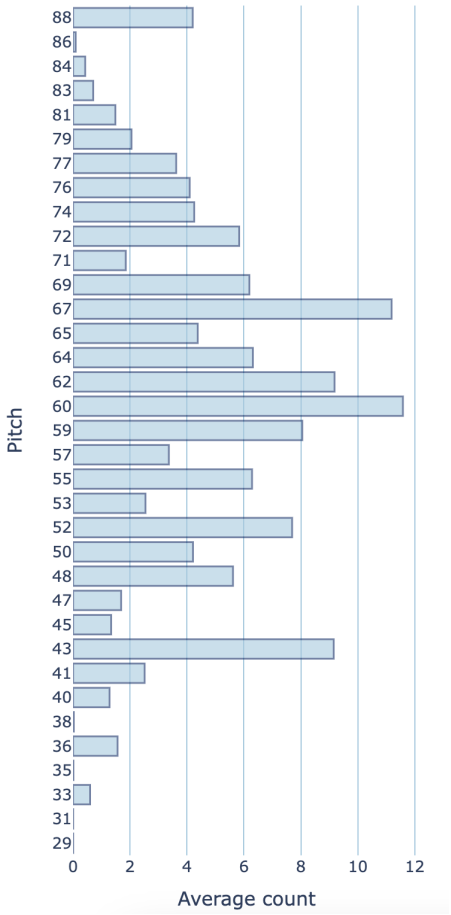

Fig. 6. Average occurrence of musical pitch values (Reconstructed MuseCoCo)

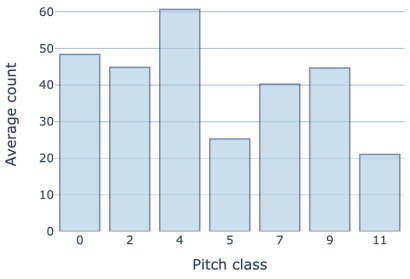

Fig. 7. Average occurrence of musical pitch classes (LoRA GPT-2)

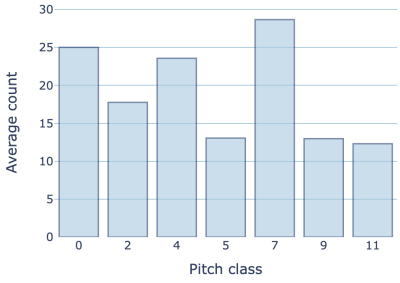

Fig. 8. Average occurrence of musical pitch classes (Reconstructed MuseCoCo)

5 Conclusion

In this paper, we study the model for generating MIDI from text descriptions in multiple languages. Instead of using a large one-stage model, we experimented with an architecture of two compact models and adjusted various parameters to evaluate their music generation capabilities. The results show that decoder models such as GPT-2 are well suited for symbolic music generation, while encoder models such as BERT are well suited for classification and feature extraction. We also applied the LoRA technique during training to optimize resource usage. In addition, we propose a method to verify the correctness of the generated music and to interpolate based on music theory principles to ensure that the music is complete and applicable in practice. While the proposed methods yield satisfactory results, further research is needed to adapt the parameters to more diverse and comprehensive datasets. The goal is to develop a model that can support a wide range of musical genres and styles. Furthermore, these experiments can be extended by incorporating additional parameters into the models. In conclusion, this study illustrates the potential of compact models and the LoRA technique in symbolic music generation, thereby providing a foundation for future research and applications.

References

1. Agostinelli, A., Denk, T.I., Borsos, Z., Engel, J., Verzetti, M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., Sharifi, M., Zeghidour, N., Frank, C.: Musiclm: Generating music from text (2023)
2. Anderson, C., Carlton, D., Miyakawa, R., Schwachhofer, D.: Hooktheory, https://www.hooktheory.com
3. Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., Défossez, A.: Simple and controllable music generation. In: Thirty-seventh Conference on Neural Information Processing Systems (2023)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019), https://arxiv.org/abs/1810.04805
5. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks (2014), https://arxiv.org/abs/1406.2661
6. Hayes, T., Zhang, S., Yin, X., Pang, G., Sheng, S., Yang, H., Ge, S., Hu, Q., Parikh, D.: Mugen: A playground for video-audio-text multimodal understanding and generation (2022)
7. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models (2021), https://arxiv.org/abs/2106.09685
8. Huang, Q., Jansen, A., Lee, J., Ganti, R., Li, J.Y., Ellis, D.P.W.: Mulan: A joint embedding of music audio and natural language (2022), https://arxiv.org/abs/2208.12415
9. Huang, Y.S., Yang, Y.H.: Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions (2020)

10. Lidy, T., Rauber, A.: Music Information Retrieval, pp. 448–456. IGI Global (1 2009). https://doi.org/10.4018/978-1-59904-879-6.ch046, http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-59904-879-6.ch046
11. Lu, P., Xu, X., Kang, C., Yu, B., Xing, C., Tan, X., Bian, J.: Musecoco: Generating symbolic music from text (2023), https://arxiv.org/abs/2306.00110
12. OpenAI: Chatgpt, https://chatgpt.com
13. Payne, Christine: Musenet (2019), https://openai.com/blog/musenet/
14. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2023), https://arxiv.org/abs/1706.03762
16. Warnerfjord, M.: Evaluating chatgpt's ability to compose music using the midi file format (2023)
17. Wu, S., Sun, M.: Exploring the efficacy of pre-trained checkpoints in text-to-music generation task. arXiv preprint arXiv:2211.11216 (2022)
18. Zeng, M., Tan, X., Wang, R., Ju, Z., Qin, T., Liu, T.Y.: Musicbert: Symbolic music understanding with large-scale pre-training (2021), https://arxiv.org/abs/2106.05630
19. Zhang, Y., Wang, Z., Wang, D., Xia, G.: Butter: A representation learning framework for bi-directional music-sentence retrieval and generation. In: Proceedings of the 1st workshop on nlp for music and audio (nlp4musa). pp. 54–58 (2020)