Text2midi: Generating Symbolic Music from Captions

Keshav Bhandari[1], **Abhinaba Roy**[2], **Kyra Wang**[2], **Geeta Puri**[2], **Simon Colton**[1], **Dorien Herremans**[2]

[1]Queen Mary University of London
[2]Singapore University of Technology and Design
k.bhandari@qmul.ac.uk, abhinaba_roy@sutd.edu.sg, kyra_wang@mymail.sutd.edu.sg

Abstract

This paper introduces `text2midi`, an end-to-end model to generate MIDI files from textual descriptions. Leveraging the growing popularity of multimodal generative approaches, `text2midi` capitalizes on the extensive availability of textual data and the success of large language models (LLMs). Our end-to-end system harnesses the power of LLMs to generate symbolic music in the form of MIDI files. Specifically, we utilize a pretrained LLM encoder to process captions, which then condition an autoregressive transformer decoder to produce MIDI sequences that accurately reflect the provided descriptions. This intuitive and user-friendly method significantly streamlines the music creation process by allowing users to generate music pieces using text prompts. We conduct comprehensive empirical evaluations, incorporating both automated and human studies, that show our model generates MIDI files of high quality that are indeed controllable by text captions that may include music theory terms such as chords, keys, and tempo. We release the code and music samples on our demo page for users to interact with `text2midi`.

Code — https://github.com/AMAAI-Lab/Text2midi

Introduction

The rapid advancements in large language models (LLMs) have revolutionized the way we interact with and leverage various forms of media, including text, images, and audio. Researchers have been able to build systems that demonstrate remarkable capabilities in both analyzing diverse input texts as well as generate highly precise text, image, video, as well as audio output (Touvron et al. 2023; Ramesh et al. 2022, 2021; Melechovsky et al. 2024) This has paved the way for a wide array of downstream applications such as question-answering systems (Touvron et al. 2023), image generation tools (Ramesh et al. 2021), and even music generation platforms (suno.com 2024). These breakthroughs have ushered in a new era of multimodal intelligence, where the integration of multiple data sources and modalities can be harnessed to enhance our understanding, creativity, and problem-solving abilities across a wide range of domains. While prior work has explored generating audio music from text, the potential of large language models has not yet been fully realized for symbolic music generation. In this paper, we introduce `text2midi`, a system that leverages the power of LLMs to generate symbolic music in the form of MIDI files from textual descriptions.

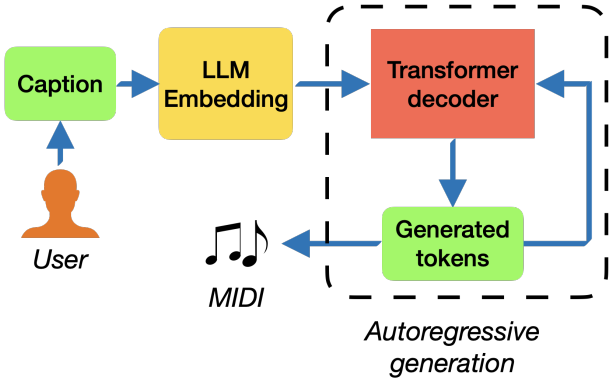

Figure 1: Overview of our approach. We use a pretrained LLM encoder to encode input captions. This is passed to the trained transformer decoder which generates encoded MIDI tokens autoregressively.

In Music Information Retrieval (MIR), MIDI is a crucial format that provides a symbolic representation of music (Schedl et al. 2014; Herremans, Chuan, and Chew 2017). Its symbolic nature has established it as a popular format for music creation, which is used by music producers and composers in popular Digital Audio Workstations (DAWs).

In recent times, there is a growing emphasis on the creation of music based on unstructured text instructions (Copet et al. 2023; Melechovsky et al. 2024; Huang et al. 2023). These models leverage large language models to transform textual descriptions of musical attributes into concrete audio compositions. This process requires a precise alignment between textual content and musical features to ensure that the resulting music accurately reflects the provided text guidelines. However, the majority of existing methodologies primarily focus on directly producing *audio* music. In contrast, symbolic music representation, particularly in MIDI format, has not received sufficient attention from researchers within the community. The lack of text-to-midi models can largely

be attributed to the lack of a large-scale dataset of MIDI files annotated with text captions. Recently, (Melechovsky, Roy, and Herremans 2024) expanded upon the Lakh MIDI dataset—which includes 168,407 MIDI files—by incorporating free-form text captions that provide musical insights; referred to as MidiCaps. To date, the only significant contribution comes from (Lu et al. 2023), which adopts an indirect methodology by deriving musical features from textual captions for use as conditions. This approach involves two distinct stages: initially focusing on understanding and extracting musical attribute values from plain text during a comprehension phase; subsequent to this extraction process, these attributes are fed into a generator network designed to produce symbolic music in a second pass. It should be noted that both training and inference phases associated with such a dual-stage process can prove time-consuming.

In this work, we propose `text2midi`, the first end-to-end model for generating high quality MIDI files directly from text (see Figure 1). Our goal is to create an intuitive and accessible solution that caters to both technical and non-technical musicians alike. We employ a transformer architecture based on an encoder-decoder framework capable of processing any unstructured text (caption) as its input. This process involves utilizing a pretrained large language model (LLM) encoder to embed the provided caption, which subsequently acts as conditional data for the decoder to generate MIDI sequences in an autoregressive manner. To maximize our architecture's potential, we implement semi-supervised pretraining using the SymphonyNet dataset (Liu et al. 2022) containing 46,359 MIDI files. During data preprocessing, we create pseudo captions that encapsulate the musical attributes present in the MIDI files and use the pseudo caption-MIDI pair for pretraining. Subsequently, we leverage Midicaps—a specialized dataset aimed at facilitating text-to-MIDI generation to fine-tune our model and enhance performance. To assess the effectiveness of our training methodology, we conduct subjective human evaluations to gauge the overall musical quality of the MIDI output. Additionally, we also perform objective assessments to verify alignment between musical attributes specified in input captions and those present in the resultant MIDI files. We expect that our efforts will assist both laymen, musicians, and music producers alike, as they can express their concepts through free-flowing text and utilize the generated MIDI files as a foundational element for their compositions, or just as they are. Furthermore, we are confident that our work will motivate researchers within the Music Information Retrieval (MIR) domain to explore MIDI generation tasks more thoroughly.

The main contributions of our work are as follows:

- We present **`text2midi`**, the first end-to-end model for the task of generating MIDI files from text captions. Our model leverages the power of pretrained large language models to encode the provided textual descriptions and then utilizes an autoregressive transformer decoder to generate the corresponding MIDI sequences.
- We utilize a semi-supervised pretraining approach using symphonynet (Liu et al. 2022) - a dataset with complex multi-track and multi-instrument MIDI files. We first extract relevant musical attributes from this dataset, such as instruments, tempo, and time signature, and generate pseudo captions from these attributes to create caption-MIDI pairs for pretraining our model.
- We are the first to train on the MidiCaps dataset, a curated large-scale collection of MIDI-caption pairs, for the task of text-to-MIDI generation. This dataset allows us to further fine-tune and evaluate our model's performance in generating MIDI compositions from textual descriptions.

Related Work

The first generative symbolic music model, developed by Funk (2018), composed the famous string quartet 'the Illiac Suite' using rules and Markov chains. The field of automatic composition has slowly grown since then, gaining new popularity with the introduction of various types of deep networks such as recurrent neural networks (RNNs), which researchers quickly discovered helped to maintain long-term structure in their models (Chuan and Herremans 2018). For instance, BachBot (Liang 2016) and DeepBach (Hadjeres, Pachet, and Nielsen 2017) are two of the popular models based on long-short term memory models (LSTMs) that generate music sequences in the style of Bach. With the introduction of the Transformer as well as steadily growing MIDI datasets, more powerful models such as Music Transformer (Huang et al. 2018) and the recent Museformer model (Yu et al. 2022) were developed. The latter leverages the fine-grained and coarse-grained attention in Transformers (Vaswani 2017) to generate fairly long and high quality music sequences. For a more complete overview of this evolution and currently remaining challenges, the reader is referred to Herremans, Chuan, and Chew (2017) and Ji, Yang, and Luo (2023).

To make generative music models really useful, they should be controllable by the user. This way, they provide a way to augment the human composer rather than just replace them. In the literature, we find models that allow the generated MIDI files to be controlled based on an input emotion (Makris, Agres, and Herremans 2021), chords (Zixun, Makris, and Herremans 2021), lyrics (Yu, Srivastava, and Canales 2021), tension (Herremans and Chew 2017; Guo et al. 2020), among others. Typically, these models work by feeding an input label with the desired condition, which are processed through cross attention in transformer networks, or other types of conditional neural networks.

In a recent evolution of the field of generative audio, a number of text-to-music models have been developed in the last two years (Melechovsky et al. 2024; Copet et al. 2023; Huang et al. 2023). These models generate audio directly based on a free-form text prompt. They typically leverage pretrained large language text encoders, such as FLAN-T5 (Chung et al. 2024), and feed the resulting embedded text representation as input to the audio decoder. Some models generate shorter fragments using diffusion (Melechovsky et al. 2024), whereas others allow longer fragments through an autoregressive transformer approach (Copet et al. 2023). In this paper, we follow a similar approach as the latter, except on symbolic music.

In the field of generative MIDI models, we have not yet seen such an evolution. The only text to MIDI model currently is MuseCoco (Lu et al. 2023). MuseCoco generates symbolic music from text descriptions by leveraging a two-stage framework that constitutes text-to-attribute understanding and attribute-to-music generation. It provides more control over music elements which has always been a pain point for previous models. The recently released chatGPT4o[1] is also able to generate MIDI files based on a text prompt, however, after a quick examinations, the files are mostly nonsensical or barely contain notes. A similar conclusion was drawn by Lu et al. (2023). Another attempt at developing symbolic music from text captions is due to (Zhang et al. 2020), who developed the BUTTER (Better Understanding of Text To Explanation and Recommendation) model. This model generates music in ABC format [2]based on a text caption. Like MuseCoco, the model also takes a two-stage approach and predicts key words such as music attributes from the text caption to generate folk music. Finally, Wu and Sun (2022) explores how pretrained language models can be used to generated ABC format music. However, the ABC format is limited in terms of representing polyphonic multitrack music.

In this work, we aim to take a holistic approach and train an end-to-end model directly on the recently released MidiCaps dataset (Melechovsky, Roy, and Herremans 2024). This dataset contains 168,385 MIDI files that have been annotated with rich text captions. In the next section we discuss out model architecture, which is followed by results of our experiments and discussion.

Method

Here, we first discuss the mathematical formulation for our problem followed by the transformer architecture used and the semi-supervised pretraining.

Mathematical Formulation

The input text T is first encoded using a pretrained Flan T5 encoder E, as $H_T = E(T)$, where H_T is an $n \times d$ dimensional vector, n is the token or text sequence length, and d is the T5 feature dimension. MIDI data M is tokenized using the REMI tokenizer to generate the sequence $S_M = M_{\text{tok}}(M)$. The decoder $D(H_T, S_M)$ utilizes the sequence S_M and H_T as hidden states to predict the next MIDI token at time step m:

$$P(S_{m+1} \mid H_T, S_m) = \text{softmax}(D(H_T, S_m))$$

where S_m represents MIDI tokens up to time step m. Finally, a REMI decoder M_{detok} converts the generated tokens back to MIDI:

$$M_{\text{gen}} = M_{\text{detok}}(D(E(T), S_M))$$

Architecture

Transformer Conditioning With Text As shown in Figure 2, the proposed `text2midi` model consists of an encoder-decoder transformer architecture. We use the pretrained FLAN T5 model (Chung et al. 2024) as our encoder to embed text captions before passing them to the Transformer (Vaswani 2017) decoder layers with cross attention. Flan T5 advances the T5 model (Raffel et al. 2020) through instruction fine-tuning on a diverse set of tasks. The instruction fine-tuning approach enables FLAN T5 to generalize better to unseen tasks, allowing it to perform tasks more effectively based on natural language instructions. As Flan T5 offers strong language understanding, evident from the zero shot and few shot learning benchmarks over regular T5, we postulate that its embeddings can effectively capture the semantic richness of text captions, translating textual descriptions into musical concepts accurately without re-training. Thus, we freeze the weights of the Flan T5 model during the training process.

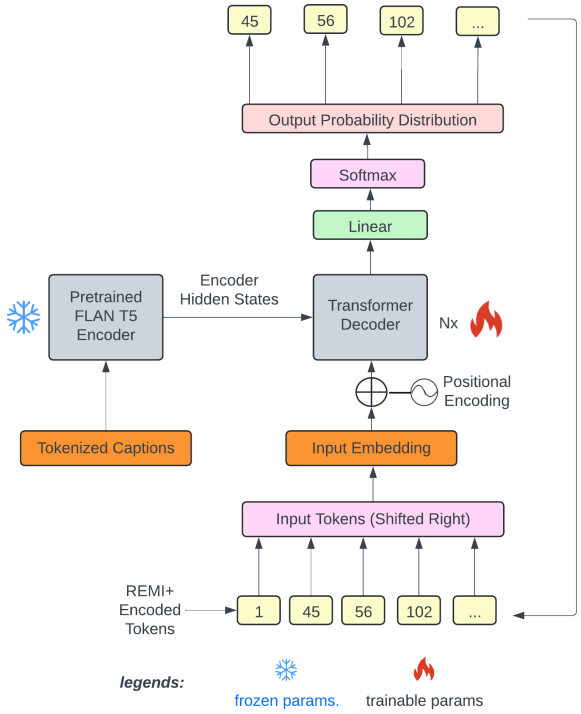

Figure 2: Overview of our model's architecture. The FLAN T5 encoder weights are frozen. The transformer decoder accepts the encoder's hidden states via cross attention before generating the REMI+ encoded tokens autoregressively.

MIDI Tokenizer We use the MidiTok library (Fradet et al. 2021) to encode multi-track multi-instrument MIDI files with the REMI+ tokenizer (von Rütte et al. 2023), an extension of the REMI encoding (Huang and Yang 2020). The REMI tokenizer represents musical notes in a tokenized format, using tokens for tempo, pitch, velocity, duration, bar, onset, and positions within bars. This tokenizer is extended to REMI+ by adding MIDI program and time signature tokens, allowing it to handle multi-track and multi-instrument compositions with varying time signatures.

[1]chatgpt.com

[2]https://abcnotation.com/

Transformer Decoder Given that we have represented the MIDI sequence as tokens, we can use a transformer decoder to generate these tokens autoregressively, much like a language generation task. We implemented the flash attention (Dao et al. 2022) algorithm in the Transformer Decoder, over the vanilla attention mechanism from the original study (Vaswani 2017) so as to speed up the computation as well as reduce the memory usage of the model, particularly for long sequence tasks such as this one.

We utilize vanilla positional embeddings to encode the order of tokens in the sequence. After the final decoder layer, a linear projection layer is employed to map the decoder outputs to the vocabulary size. During training, we optimize the model using the categorical cross-entropy loss, defined as:

$$\mathcal{L}_{\text{CCE}}(y, \hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{1}$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability for class i.

Semi-Supervised Pretraining

The proposed `text2midi` model aims to generate music sequentially (from the beginning) based on textual prompts, which necessitates using the first n encoded tokens per MIDI file rather than randomly cropping from the middle. This approach, however, limits our training data as we cannot utilize information beyond the model's context window. To address this limitation, we implement a two-stage training process. First, we pretrain on a larger dataset using placeholder text captions. For this pretraining dataset, which lacks predefined captions, we extract objective attributes from the MIDI files using the Music21 library(Cuthbert and Ariza 2010). These attributes are: time signature, key signature, beats per minute (BPM), and instruments. We construct 10 different placeholder sentences incorporating these attributes, which are then processed by the Flan T5 encoder. An example of a placeholder caption is *"Played at **114** beats per minute in **4/4** time signature and the key of **G# minor**, classical piece with the following instruments: **clarinet, English horn, flute, horn, piccolo, trombone, and trumpet**."*, where the text in bold are the placeholders.

Following the pretraining phase, the model is fine-tuned on a dataset with more detailed captions that include both the objective attributes used during pretraining and subjective elements such as style and mood. This allows the model to learn sophisticated text-to-music mappings. During finetuning, a sentence omission technique is applied at each iteration, where 20%-50% of sentences are randomly omitted with a 50% probability. This technique regularizes the model by exposing it to a more varied and unpredictable training set, preventing overfitting and enhancing its generalization to unseen data. Additionally, it serves as implicit data augmentation by creating a larger and more diverse set of training examples, helping the model learn to fill in gaps and generate coherent outputs from incomplete inputs. Finally, it prepares the model for real-world scenarios where users may provide minimal or incomplete captions, enhancing its practical effectiveness.

Experimental Setup

Evaluating generative models is a challenging task (Agres, Forth, and Wiggins 2016). In order to establish the effectiveness of `text2midi`, we conduct both an objective evaluation of the results as well as a subjective evaluation in the form of a listening test. This allows us to thoroughly test both the resulting musical quality as well as how much the generated MIDI adheres to the input caption. In this section, we first detail the datasets and model configurationused to train the model in the experiments, followed by experimental configuration and evaluation metrics. Finally, we discuss results and findings of our experiments.

Training Datasets

We work with two datasets during our training process: SymphonyNet for semi-supervised pretraining and MidiCaps for finetuning towards MIDI generation from captions.

MidiCaps is a dataset of 168,401 unique MIDI files with text captions (Melechovsky, Roy, and Herremans 2024). The MIDI files were originally provided in the Lakh MIDI dataset (Raffel 2016), released under the CC-BY 4.0 license. Each MIDI file is paired with a music feature-rich caption. An example caption in the dataset: "A melodic and happy pop song with a Christmas vibe, featuring piano, clean electric guitar, acoustic guitar, and overdriven guitar. The song is in the key of A major with a 4/4 time signature and a moderate tempo. The chord progression revolves around D, E6, D, and E, creating a motivational and loving atmosphere throughout the piece." We use the provided training set (90% of the data) to train the model in our experiments.

SymphonyNet (Liu et al. 2022) is a comprehensive dataset of symphonic music. It comprises 46,359 multi-instrument, multi-track MIDI files, predominantly symphonic, with an average duration of 4.26 minutes per file. We selected SymphonyNet for semi-supervised pretraining due to its high-quality, multi-track data, which aligns well with our model's requirements. The SymphonyNet MIDI files were augmented with pseudo-captions as described in the methods section.

Model Configuration

We use an encoder-decoder transformer architecture for pretraining on SymphonyNet and finetuning on MidiCaps. As mentioned earlier, our encoder is a pretrained FLAN T5 model (Chung et al. 2024). Our transformer decoder consists of 18 layers and 8 attention heads, with 159M trainable parameters. Our architecture consists of 272M parameters in total. We observe that the MIDI tokenizer contains on average 4-5 minutes length of data per 2,000 tokens. Since the average length of tracks in SymphonyNet as well as MidiCaps is 4-5 minutes, we configured the input to the transformer decoder to be 2,048 MIDI tokens. Any smaller sequences in dataset are padded. For pretraining, we train for 100 epochs, with a batch size of 4 and gradient accumulation set to 4. For finetuning on MidiCaps, we trained for 30 epochs. For both runs, we use the Adam optimizer (Kingma and Ba 2014) coupled with a cosine learning rate schedule

with a warm-up of 20,000 steps. For pretraining, our base learning rate is $1e^{-4}$ whereas for finetuning, we use a reduced base learning rate of $1e^{-6}$. Our models are trained on 6 NVIDIA L40S 48 GB GPUs.

Test Datasets and Baselines

In order to maintain fairness with subjective results (based only on five generated examples), we consider 100 (5%) randomly selected samples from the MidiCaps test set (Melechovsky, Roy, and Herremans 2024). We compare the MIDI generated by `text2midi` to the ground truth MIDI file from the MidiCaps dataset, as well as with MIDI generated by the MuseCoco model (xlarge) (Lu et al. 2023)[3].

Objective Evaluation Metrics

In the objective evaluation of our model, we address three questions - how much long-term structure and patterns the music contains, how similar the input text and generated music are and how close some of the important musical features are to that of the ground truth. To answer these, we use three types of evaluation metrics as discussed below:

Compression ratio uses the COSIATEC algorithm (Meredith 2013) as used by (Chuan and Herremans 2018) to measure the long-term structure and repeating patterns in music. We measure the compression ratio MIDI files to quantify amount of long-term structure in them.

CLAP or Contrastive Language-Audio Pretraining (Wu* et al. 2023) enables the extraction of joint latent representations of audio and text samples. We use an improved CLAP checkpoint (LAION CLAP[4]), specifically trained on music and speech. We extract the latent representations of an input text (caption) and the synthesized audio of the MIDI file. We then use the cosine similarity between these latent representations as a measure of similarity between text (caption) and MIDI (synthesized audio). This enables us to evaluate how closely MIDI files fit their text caption.

Feature wise comparison consists of four features extracted from generated MIDI files and their related ground truth MIDI files as used by Melechovsky et al. (2024):

- Tempo Bin (TB): the ratio of files of which the extracted tempo (in terms of beats per minute (bpm)), falls within the tempo bin extracted from the ground truth MIDI file, with bin borders defined at 40, 60, 70, 90, 110, 140, 160, 210 bpm. The tempo is extracted with music21 (Cuthbert and Ariza 2010).
- Tempo Bin with Tolerance (TBT): this metric allows for slightly more tolerance than the previous one. It is calculated as the ratio of MIDI files of which the predicted bpm falls into the ground truth tempo bin or a neighboring one, compared to all files.
- Correct Key (CK): the ratio of MIDI files of which the extracted key (using music21 (Cuthbert and Ariza 2010)) matches the extracted of the ground truth key MIDI file.
- Correct Key with Duplicates (CKD): this metric allows for slightly more tolerance than the previous one. It is calculated as the ratio of generated MIDI files for which the extracted key matches the key of the extracted key of the ground truth MIDI key or an equivalent key (i.e., a major key and its relative minor).

Listening Test

For the subjective assessment of our model, we conducted a listening study utilizing PsyToolkit (Stoet 2010). Participants were invited to listen to 15 rendered MIDI files: 5 randomly selected captions from the MidiCaps test dataset with their accompanying ground truth MIDI, 5 generated by `text2midi` using the text captions of the above 5 MIDI files from MidiCaps as input, and 5 generated by MuseCoco using the same text captions. We did not cherry pick the generated samples. These 15 samples were randomly ordered for each participant. Listeners evaluated these tracks across six criteria:

- Musical quality of the music
- Overall match between the music and the caption
- The genre of the music matches the caption
- The mood of the music matches the caption
- The key of the music matches the caption
- The chords in the caption are prominent in the music
- The tempo of the music matches the caption

Results and Discussion

We assess various aspects with a particular focus on how the generated music reflects the captions, both in an objective experiment as well as a listening study. For all our evaluation tasks, we generate MIDI with 2,000 tokens. All our generated MIDI files typically last between 30-40 seconds when played back. For a fair comparison, while comparing with ground truth (gt) data, we only consider the first 40 seconds of the latter.

Objective Results

Table 2 shows the results of the objective evaluation. Evaluation metrics are computed on MIDI files generated (from the same caption) by both `text2midi`, MuseCoco (Lu et al. 2023), and the original MIDI files (ground truth) from the data set (MidiCaps).

Generating music with long-term structure is a known challenge (Bhandari and Colton 2024). We use the compression ratio, as done by Chuan and Herremans (2018), to evaluate the presence of repeated patterns in the MIDI files. `text2midi` achieves an average compression ratio of 2.31, significantly higher than MuseCoco's 2.12 ($p < 0.0001$). This result demonstrates that `text2midi` outperforms MuseCoco in creating long-term structure and repeating patterns.

To evaluate the match between text caption and MIDI, we calculate the CLAP score. In Table 1, we see that `text2midi`'s CLAP score (0.22) beats that of MuseCoco (0.21), which is quite encouraging. However, both models

[3]For detailed and latest results, readers are invited to visit our demo page https://github.com/AMAAI-Lab/Text2midi

[4]https://huggingface.co/laion/larger_clap_music_and_speech

Generated by:	MidiCaps	`text2midi`	MuseCoco
Question	Avg. rating (1-7)		
Musical Quality	5.79	4.62	4.40
Overall matching	5.42	4.67	4.07
Genre matching	5.54	4.98	4.40
Mood matching	5.70	5.00	4.32
Key matching	4.61	3.64	3.36
Chord matching	3.20	2.50	2.00
Tempo matching	5.89	5.42	4.94

Table 1: Results of the listening study. Each question is rated on a Likert scale from 1 (very bad) to 7 (very good). The table shows the average ratings per question for each set of generated music.

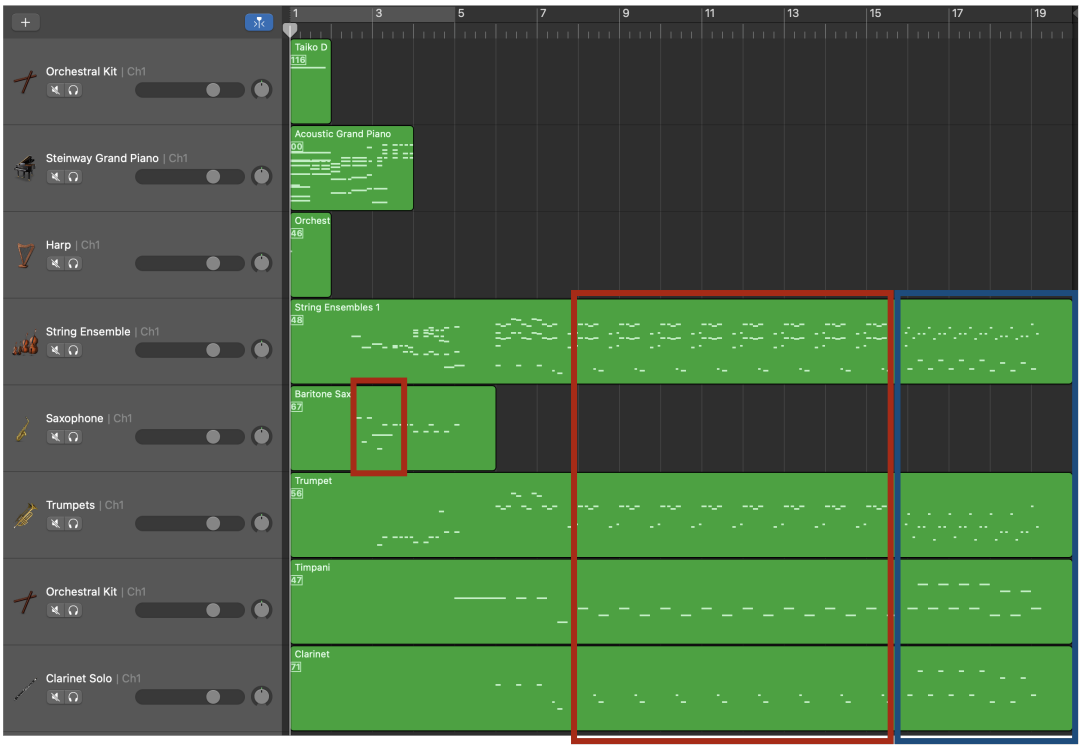

Figure 3: A piano roll representation of a generated MIDI using the following caption "A haunting electronic ambient piece that evokes a sense of darkness and space, perfect for a film soundtrack. The string ensemble, trumpet, piano, timpani, and synth pad weave together to create a meditative atmosphere. Set in F minor with a 4/4 time signature, the song progresses at an Andante tempo, with the chords F, Fdim, and F/C recurring throughout."

fall below the ground truth CLAP score (0.26), showing room for further improvement. We observed that, for the specific examples where the similarity between the text and ground truth MIDI is high (i.e. high CLAP score), the CLAP score for generated MIDI is also high. We thus posit that our model is able to adhere to the prompts and achieve better text-music alignment compared to our baseline.

While the CLAP score provides a general indication of how well the text and MIDI align, we further analyze specific aspects in which the generated MIDI resembled the ground truth MIDI. It is important to note that as these metrics compare the generated MIDI with the ground truth MIDI, they cannot be computed for MidiCaps, which serves as the ground truth here. We show that in terms of both tempo and key attributes, `text2midi` outperforms the MuseCoco model. The tempo bin (TB) metric reaches 39.70 compared to 21.71 for MuseCoco, and the TBT reaches 65.80 compared to 54.63 for MuseCoco. These ratios indicate that about 66% of the tempo instructions are very close to the desired tempo. The CKD ratios for `text2midi` are also higher than MuseCoco (35.60% vs 14.59%), indicating better key matching. We note that both CK and CKD are statistically significant ($p < 0.0001$). In terms of inference speed, we notice that `text2midi` is much faster compared

	text2midi	MidiCaps	MuseCoco	P-Val
CR ↑	2.31	3.43	2.12	< 0.0001
CLAP ↑	0.22	0.26	0.21	0.0102
TB (%) ↑	39.70	-	21.71	0.1102
TBT (%) ↑	65.80	-	54.63	0.2051
CK (%) ↑	33.60	-	13.70	< 0.0001
CKD (%) ↑	35.60	-	14.59	< 0.0001

Table 2: Summary of our objective evaluations. Numbers are average results for all captions from the MidiCaps test set. CR: Compression ratio; CLAP: CLAP score; TB: Tempo Bin; TBT: Tempo Bin with Tolerance; CK: Correct Key; CKD: Correct Key with Duplicates; ↑: higher score better.

to MuseCoco as a result of unifying the text-to-attribute and attribute-to-music models into a single holistic approach. Based on our observation, text2midi takes around 55 seconds to generate a 40-second long MIDI file, compared to 120 seconds for MuseCoco (inference performed on GPU in both cases). This comprehensive evaluation demonstrates that text2midi consistently surpasses MuseCoco on all objective metrics, despite training on a data set of only one fifth of the size used to train MuseCoco.

Subjective Results

A total of 11 participants participated in the listening test, of these, 3 individuals reported being able to recognize chords and keys with absolute pitch capabilities, while 6 individuals reported receiving more than one year of musical training experience.

The results of our subjective test are shown in Table 1. In general, all ratings for the ground truth MidiCaps examples gravitate towards 'good' (score 5) and that of text2midi and MuseCoco generated examples, towards 'neutral'(4) to 'good' (5). On questions where the MidiCaps files receive lower scores, the generated examples also tend to score lower. The similar lower scores indicate a close alignment between the trained model and the original MidiCaps data. On average, although for all evaluation criteria MidiCaps examples fare better, they don't outperform text2midi generated examples by a large margin. text2midi generated examples also scored better in all questions compared to MuseCoco generated examples. This shows the effectiveness of text2midi's ability to better follow the musicality of the MidiCaps ground truth. We have kept the listening test open for participation. Readers are requested to visit our GitHub page[5] for updated results.

Discussion

The text2midi model exhibits strong capabilities in generating music with long-term structure and repeating patterns that make compositional sense, while still being able to break out of those repetitions, and to then make use of the same motifs from earlier in the composition. Figure 3 shows the piano roll representation of a generated piece. An example of a repeated motif is shown in the red boxes, which first occurs at 3 seconds in, being used again with variations at 14 seconds, before switching to a different repeating pattern with a similar theme at 33 seconds (highlighted in blue). Such translated motifs would be recognized as patterns with COSIATEC (Meredith 2013) and hence leads to a higher compression ratio. The example also nicely shown that appropriate instrument tracks are generated to match the example (classical piece). The notes also appear 'long' on the piano roll, indicating a match with the desired 'slow and emotional' character. text2midi still struggles with instrumentation, which is apparent in this example: where a 'classical' piece is asked for, the instruments definitely match this (e.g. trombone, french horn), however, the additional instruction of 'church organ' is ignored. This could be due to the fact that the captions describe the entire song, whereas we only used part of the song for training, hence some instruments may have not started played. In future work, it would be interesting to explicitly focus on encoding instrumentation.

[5]https://github.com/AMAAI-Lab/Text2midi

Conclusion

In this paper, we introduce text2midi, the first end-to-end model for generating MIDI files directly from textual descriptions. The text2midi model leverages the power of pretrained large language models (LLMs) and combines it with an autoregressive transformer decoder to generate expressive MIDI files. The model is trained using semi-supervised learning on large-scale datasets including MidiCaps and SymphonyNet. In a detailed analysis, our analytical experiment shows that text2midi is able to generate MIDI pieces that have a long-term structure with repeating patterns, and show that the requested features in the input text caption are indeed adhered to in the generated MIDI.

This work not only advances the state-of-the-art in text-to-MIDI generation but also opens up new avenues for intuitive music composition, by both making it accessible to a broader audience through the use of simple text prompts, as well providing a user-friendly tool for expert composers and producers to generate musical ideas. In future, we believe this model may be further trained to increase its output quality as well as be expanded to include more complex tasks including midi editing guided by text prompts. Another current limitation is the data quality, with higher quality creative commons MIDI files, model quality would further increase.

Acknowledgments

This work was supported by the UKRI and EPSRC under grant EP/S022694/1 and SUTD's Kickstart Initiative under grant number SKI 2021_04_06. We owe much appreciation to our reviewers for their insightful critiques which greatly strengthened the work.

References

Agres, K.; Forth, J.; and Wiggins, G. A. 2016. Evaluation of musical creativity and musical metacreation systems. *Computers in Entertainment (CIE)*, 14(3): 1–33.

Bhandari, K.; and Colton, S. 2024. Motifs, Phrases, and Beyond: The Modelling of Structure in Symbolic Music Gener-

ation. In *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*, 33–51. Springer.

Chuan, C.-H.; and Herremans, D. 2018. Modeling temporal tonal relations in polyphonic music through deep networks with a novel image-based representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, Y.; Wang, X.; Dehghani, M.; Brahma, S.; et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70): 1–53.

Copet, J.; Kreuk, F.; Gat, I.; Remez, T.; Kant, D.; Synnaeve, G.; Adi, Y.; and Défossez, A. 2023. Simple and Controllable Music Generation. *arXiv preprint arXiv:2306.05284*.

Cuthbert, M. S.; and Ariza, C. 2010. music21: A toolkit for computer-aided musicology and symbolic music data.

Dao, T.; Fu, D. Y.; Ermon, S.; Rudra, A.; and Ré, C. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Fradet, N.; Briot, J.-P.; Chhel, F.; El Fallah Seghrouchni, A.; and Gutowski, N. 2021. MidiTok: A Python package for MIDI file tokenization. In *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*.

Funk, T. 2018. A Musical Suite Composed by an Electronic Brain: Reexamining the Illiac Suite and the Legacy of Lejaren A. Hiller Jr. *Leonardo Music Journal*, 28: 19–24.

Guo, R.; Simpson, I.; Magnusson, T.; Kiefer, C.; and Herremans, D. 2020. A variational autoencoder for music generation controlled by tonal tension. *arXiv preprint arXiv:2010.06230*.

Hadjeres, G.; Pachet, F.; and Nielsen, F. 2017. DeepBach: a Steerable Model for Bach Chorales Generation. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 1362–1371. PMLR.

Herremans, D.; and Chew, E. 2017. MorpheuS: generating structured music with constrained patterns and tension. *IEEE Transactions on Affective Computing*, 10(4): 510–523.

Herremans, D.; Chuan, C.-H.; and Chew, E. 2017. A functional taxonomy of music generation systems. *ACM Computing Surveys (CSUR)*, 50(5): 1–30.

Huang, C.-Z. A.; Vaswani, A.; Uszkoreit, J.; Shazeer, N.; Simon, I.; Hawthorne, C.; Dai, A. M.; Hoffman, M. D.; Dinculescu, M.; and Eck, D. 2018. Music Transformer. arXiv:1809.04281.

Huang, Q.; Park, D. S.; Wang, T.; Denk, T. I.; Ly, A.; Chen, N.; Zhang, Z.; Zhang, Z.; Yu, J.; Frank, C.; et al. 2023. Noise2music: Text-conditioned music generation with diffusion models. *arXiv preprint arXiv:2302.03917*.

Huang, Y.-S.; and Yang, Y.-H. 2020. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM international conference on multimedia*, 1180–1188.

Ji, S.; Yang, X.; and Luo, J. 2023. A survey on deep learning for symbolic music generation: Representations, algorithms, evaluations, and challenges. *ACM Computing Surveys*, 56(1): 1–39.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Liang, F. 2016. Bachbot: Automatic composition in the style of bach chorales. *University of Cambridge*, 8(19-48): 3–1.

Liu, J.; Dong, Y.; Cheng, Z.; Zhang, X.; Li, X.; Yu, F.; and Sun, M. 2022. Symphony generation with permutation invariant language model. *arXiv preprint arXiv:2205.05448*.

Lu, P.; Xu, X.; Kang, C.; Yu, B.; Xing, C.; Tan, X.; and Bian, J. 2023. MuseCoco: Generating Symbolic Music from Text. arXiv:2306.00110.

Makris, D.; Agres, K. R.; and Herremans, D. 2021. Generating lead sheets with affect: A novel conditional seq2seq framework. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Melechovsky, J.; Guo, Z.; Ghosal, D.; Majumder, N.; Herremans, D.; and Poria, S. 2024. Mustango: Toward controllable text-to-music generation. *Proc. of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Melechovsky, J.; Roy, A.; and Herremans, D. 2024. MidiCaps–A large-scale MIDI dataset with text captions. *arXiv preprint arXiv:2406.02255*.

Meredith, D. 2013. COSIATEC and SIATECCompress: Pattern discovery by geometric compression. In *International society for music information retrieval conference*. International Society for Music Information Retrieval.

Raffel, C. 2016. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Ramesh, A.; Dhariwal, P.; Nichol, A.; Chu, C.; and Chen, M. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2): 3.

Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; and Sutskever, I. 2021. Zero-shot text-to-image generation. In *International conference on machine learning*, 8821–8831. Pmlr.

Schedl, M.; Gómez, E.; Urbano, J.; et al. 2014. Music information retrieval: Recent developments and applications. *Foundations and Trends® in Information Retrieval*, 8(2-3): 127–261.

Stoet, G. 2010. PsyToolkit: A software package for programming psychological experiments using Linux. *Behavior research methods*, 42: 1096–1104.

suno.com. 2024. Suno AI.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.;

Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

von Rütte, D.; Biggio, L.; Kilcher, Y.; and Hofmann, T. 2023. FIGARO: Controllable music generation using learned and expert features. In *The Eleventh International Conference on Learning Representations*.

Wu, S.; and Sun, M. 2022. Exploring the efficacy of pretrained checkpoints in text-to-music generation task. *arXiv preprint arXiv:2211.11216*.

Wu*, Y.; Chen*, K.; Zhang*, T.; Hui*, Y.; Berg-Kirkpatrick, T.; and Dubnov, S. 2023. Large-scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*.

Yu, B.; Lu, P.; Wang, R.; Hu, W.; Tan, X.; Ye, W.; Zhang, S.; Qin, T.; and Liu, T.-Y. 2022. Museformer: Transformer with Fine- and Coarse-Grained Attention for Music Generation. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 1376–1388. Curran Associates, Inc.

Yu, Y.; Srivastava, A.; and Canales, S. 2021. Conditional LSTM-GAN for melody generation from lyrics. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 17(1): 1–20.

Zhang, Y.; Wang, Z.; Wang, D.; and Xia, G. 2020. 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)*, 54–58.

Zixun, G.; Makris, D.; and Herremans, D. 2021. Hierarchical recurrent neural networks for conditional melody generation with long-term structure. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.