Interactive Musical Co-Creation in the Style of Pop Piano

Max Graf – 190120296

School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK

max.graf@qmul.ac.uk

Abstract. Computer-generated music can provide interesting insights into the structure of music and serve as inspiration for novice and professional composers alike. I investigate the use of the *Transformer-XL* neural network architecture for interactive co-creation of symbolic music in the style of pop piano. I present a modular system consisting of two software components: backend (music generation engine) and frontend (user interaction). I evaluate the neural network architecture and discuss the overall system with regard to higher-level issues in the field of computational creativity. Based on musical prompts, the system can be used to iteratively generate musical pieces of several bars length. However, it does not generalise well to new data, hindering interaction with complex user prompts.

1 Introduction

This project tackles the problem of interactive musical co-creation with artificial intelligence (AI). The topic is situated at the intersection of computing, human-computer interaction (HCI), AI and music. I investigate the use of a deep learning model, more specifically a transformer model, for generating music under the supervision of a human user. The goal of the system is not to simply generate a sequence of music of a certain length, but rather to incrementally compose a piece of music *in conjunction with the user*. This transforms the linear process of machine-composition into a cycle of composition-evaluation-feedback-composition. At any given stage in the cycle, the next musical segment depends on the feedback previously provided by the user.

The modalities of generating music are manifold, and training generative systems (or any complex deep learning model) is an expensive process, both in terms of time and resources. To minimise these expenses during the development-testing-evaluation cycle, I decided to focus on symbolic music in this project. The framework I present here is designed for music in the popular Musical Instrument Digital Interface[1] (MIDI) format. MIDI is a widely adopted standard for digital symbolic music, and as such it serves as a reasonable baseline for rapid development.

The inspiration for this research is twofold. Firstly, I am personally interested in symbolic music generation, since it is one of the core topics of my PhD research. Secondly, I discovered François Pachet's *Continuator* system [1] several months ago. This sparked my interest in implementing a co-creation system based on MIDI music myself.

Approach

My approach is based on techniques originally designed for problems in the field of natural language processing (NLP). I employ a transformer architecture [2] to model the structure and relationships of music. The style of music I selected for this project is pop piano music; an attractive candidate for an initial study, since musical pieces written in it tend to be relatively simple in terms of their rhythmic and harmonic complexity.
The system operates by generating an initial set of musical bars (sets of MIDI notes) from a prompt (a small set of MIDI data). This prompt can be user-defined (via a MIDI input device, such as a keyboard), or random. The bars are generated independently of each other. The resulting notes are visualised and presented to the user via a frontend in a web browser. The user can then inspect the generated MIDI data and play it back to evaluate it.
In the next step, the user provides feedback to the system by selecting one of three generated bars. The selected bar is then used to *condition* the generator and the cycle starts anew. In each step of the iteration the total length of the presented bars increases by 1, i.e. generated bars are attached to the prompt. This machine-composition-human-feedback cycle continues until the user is satisfied with the result.

[1] https://en.wikipedia.org/wiki/MIDI, accessed on 2 May 2021

Findings

In general, the system produces coherent and stylistically intact pieces of music. This is highly dependent on the initial prompt, however. This means that the system does not generalise well to inputs with musical structures that significantly differ from the training data, a shortcoming that becomes most striking when using custom MIDI data as a prompt. Several experiments were conducted in an approach to overcome this problem (cf. section 4).

When working with random prompts, the model is able to compose interesting pieces of pop piano music after several iterations of the cycle. User-generated prompts appear to be harder for the system continue, depending on the complexity of the input. Overall, the results are limited to the style of the underlying training data.

Structure

The remaining parts of this report cover relevant background literature (section 2). Section 3 contains a detailed explanation of the dataset used to train the model. Section 4 describes the model's structure in detail. Section 5 covers the experiments undertaken with the model, including the training and evaluation procedures. The results of the experiments are shown and discussed in section 6. Section 7 addresses higher-level issues in the field of computational creativity with regard to the system. Finally, section 8 summarises the findings and provides an outlook, explaining how the system could be improved in the future.

2 Background

The approach to modelling music taken in this project is based on sequence-to-sequence (seq2seq) modelling. Seq2seq models have been implemented using a variety of network topologies, including Long short-term memory (LSTM) networks [3], convolutional neural networks [4], as well as multitask learning [5]. Figure 1 shows a (very) high-level view of a seq2seq model. The fundamental working process shared by most of these models is given by:

1. Create a vocabulary based on the data → *tokenise* the data
2. Encode a tokenised input sequence by some means
3. Store the encoded data in *hidden states*

4. Decode the hidden states
5. Evaluate the model's performance in terms of the model's capability to correctly predict the next token

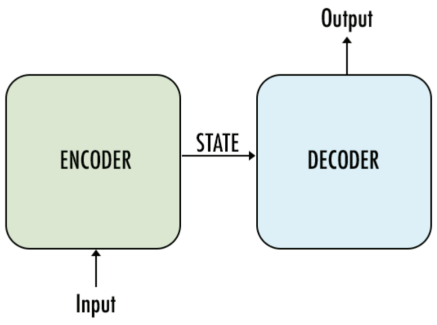


Figure 1: High-level view of a seq2seq model, taken from [6]

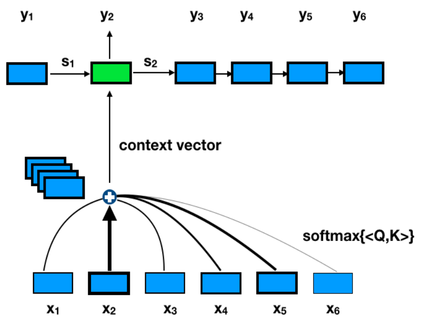


Figure 2: Conceptual view of the attention mechanism, taken from [7]

In 2017, Vaswani et al. presented a novel neural network architecture, the *transformer* [2]. In their work, they showed that an architecture that does not rely on recurrence or convolution can achieve similar or better scores on a number of sequence transduction tasks. Their model is based solely on attention and self-attention mechanisms, which significantly accelerates the training process compared to recurrent topologies. Figure 2 shows a conceptual view of the attention mechanism. Attention can be generally described as "[…] *a way to non-uniformly weight the contributions of input feature vectors so as to optimize the process of learning some target*" [7]. This means that instead of treating each token of an input sequence as equally important, a second factor is introduced (often called a *context vector*) to assign a weight to past (and/or future) tokens in the sequence. This gives a model a notion of context without the need of explicit recurrent units. This context is static, however. Two years later, Dai et al. presented one solution to this fixed-context problem [8]. Their model, *Transformer-XL*, allows for the learning of longer-term dependencies beyond a fixed length in the context vector.

In 2020, Huang and Yang applied the transformer-xl architecture to the problem of pop piano composition, showing that it is able to learn abstractions of symbolic musical data autonomously [9]. In their work they also presented a novel symbolic music representation (revamped MIDI-derived events or REMI) based on MIDI, which aids the model in learning the metrical structure, allowing it to generate MIDI data in *bars*. Their contribution, in particular the REMI data representation serves as the foundation for the work presented here.

Higher-level issues in computational creativity

Both the input data and the structure of the model in the current implementation heavily influence its outputs. For the users of the system, these two factors are implicitly framed, as there is currently no explanation of exactly *how* the model arrives at its outputs. My idea for mitigating this was to explicitly frame the outputs that the model produces by introducing the feedback cycle described above. Instead of simply generating *n* bars of music and presenting the result to the user, the model exposes its outputs continuously, creating a dialogue with the user, rather than a presentation. Finally, the creative context the system operates in is explicitly framed via the user interface; the visualisations of notes using piano rolls for each suggestion act as signifiers for the model's creative capacities.

3 Dataset

The dataset used in this project was taken from [9]. The data consists of a total of 874 single-instrument pop piano compositions in the MIDI format (774 in the training dataset and 100 in the test set). Since training the model on the full dataset proved to be computationally expensive, a subset of 600 songs was selected. During the data preparation step these 600 songs were converted into the REMI format using the code[2] supplied by the authors. This process consists of two steps:

1. Extract high-level musical events, such as
    a. Notes
    b. Chords
    c. Start/end of musical bars
    d. Tempo changes

[2] https://github.com/YatingMusic/remi, accessed on 2 May 2021

2. Tokenise these events into sequences that can be passed to the transformer model

The resulting data was stored in segments of sequences, resulting in approximately 800 segments.
As stated above, the musical complexity of the data is relatively low. To visualise that, table 1 shows a piano roll representation of the MIDI data in the file *000.midi* from the training dataset. The zoomed in view of the piece (figure on the right) shows that both rhythm and harmony are mostly kept simple, with basic chords and very few instances of syncopation.

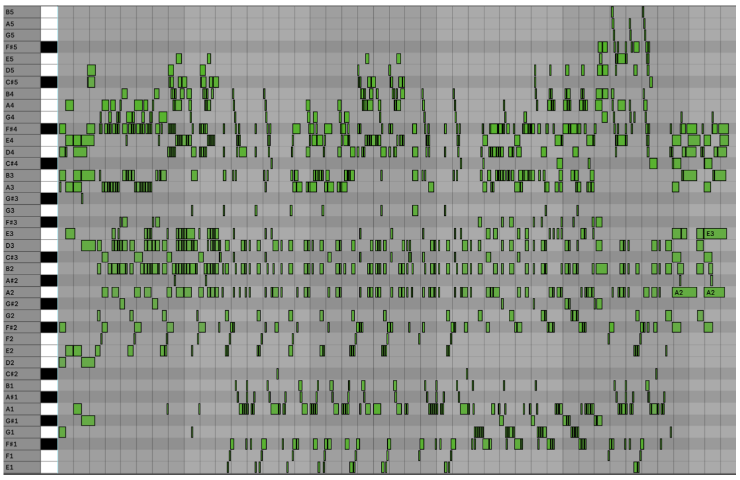
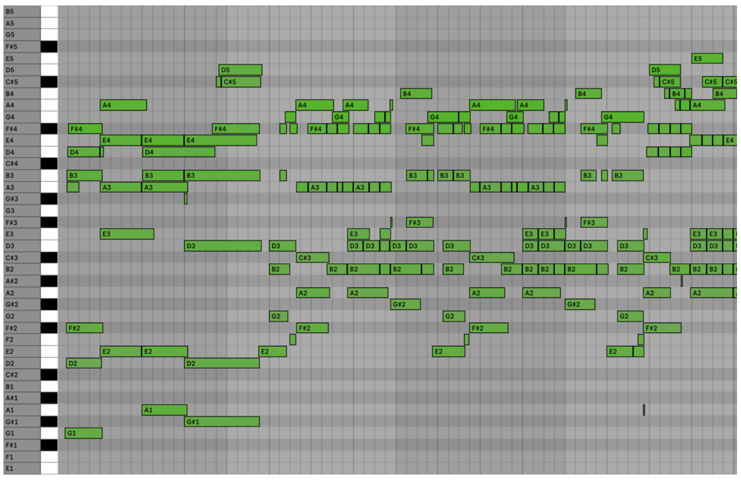



Table 1: Piano roll view of file 000.midi from the training dataset
Left: Whole song, right: zoomed in to the introduction part of the song

4 Implemented techniques

Transformer-XL model

The transformer model used in this project is based on the implementation described in [9]. While the authors implemented the model in the TensorFlow[3] machine learning framework, it is re-implemented here

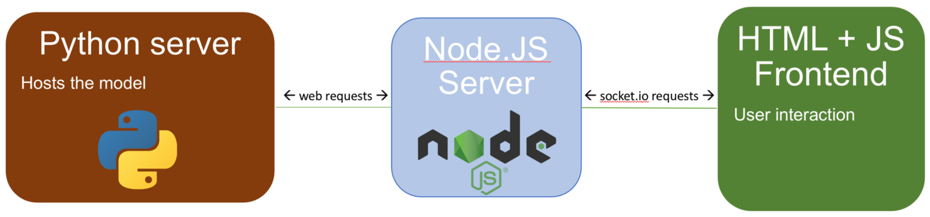
[3] https://www.tensorflow.org/, accessed on 2 May 2021

using the PyTorch[4] framework in combination with the *Transformers* software library[5]. The model uses 12 self-attention layers and 12 attention heads per attention layer in the encoder, with a memory length of 512. The length of the context vectors of the attention mechanism can be varied for each token. The label-smoothing loss function (a variation of the negative log-likelihood) is used to avoid overfitting and overconfidence. Table 2 shows a list of the most important model parameters. Dropout layers and weight-tying (tying the weights of the encoder and decoder) were employed for regularisation.

Attribute	**Value**
Number of tokens	308
Number of self-attention layers (both encoder and decoder)	12
Number of attention heads	8
Dropout probability	0.1
Memory length	512 tokens
Length of embeddings	512
Inner dimensionality of feed-forward layer	2048
Use the same attention length for all tokens?	No

Table 2: The most important model parameters

The model is integrated into a simple local Python web server (here called the *backend*). The backend communicates with the frontend via a second local web server written in Node.JS[6]. Figure 3 outlines the high-level structure of the system



[4] https://pytorch.org/, accessed on 2 May 2021
[5] https://huggingface.co/transformers/, accessed on 2 May 2021
[6] https://nodejs.org/en/, accessed on 2 May 2021

Figure 3: High-level structure of the system

To enable user interaction with the model a GUI (here called the *frontend*) was created using HTML and JavaScript. The frontend allows users to access the model's core functionality and constitutes the evaluation and feedback portions of the *composition-evaluation-feedback-composition* cycle described in section 1.

Figure 4 shows a screenshot of the user interface, after an initial set of three bars have been generated from scratch. The GUI has three major components:
The ***Generate button*** generates a new set of bars *from scratch*, i.e. using a random prompt. This will delete any previously composed MIDI data from the frontend – acting as a "reset" function.
The middle portion is comprised of ***three containers***. Those containers will be filled with generated MIDI data. The MIDI data in the containers is visualised using a simple, non-interactive piano roll representation and can be played back by clicking the play button of each player (or pressing 1, 2 or 3 on the computer keyboard). Selecting one container with the mouse (or through keys 1, 2, 3) will *highlight* it, resulting in a green border around the container (see third container in figure 4). This means that the selected section of MIDI will be used to condition the model for the next bar of MIDI that is to be generated. Once a container has been highlighted, ***pressing the "enter" key on the keyboard*** will instruct the model in the backend to compose a new bar based on the MIDI data contained in the highlighted container. This constitutes providing feedback to the model. Upon pressing the "enter" key, a few seconds (depending on the underlying computer system) will pass while the model generates three new pieces based on the data it has been conditioned with. Then the user will be presented with three new pieces of MIDI to play back and evaluate. Figure 5 shows the containers after conditioning the model with the MIDI data from container 3 in figure 4. Note that the MIDI sequence from figure 4 constitutes the first bar of all the newly generated segments in figure 5.

The final component is the circular red ***record button*** at the bottom of the GUI. By clicking it, users can record their own MIDI data into the browser using the Web MIDI API[7]. This requires a MIDI device

[7] https://www.w3.org/TR/webmidi/, accessed on 2 May 2021

connected to the computer the frontend is running on, as well as a compatible browser (Google Chrome is recommended). There is no limit to the length of user-recorded MIDI, but in the experiments shorter segments tended to produce more interesting results.

Through these controls, users can interactively co-create pop-piano music. Once they are satisfied with the resulting MIDI output, they can access the resulted MIDI files via a directory of the Node.JS server.

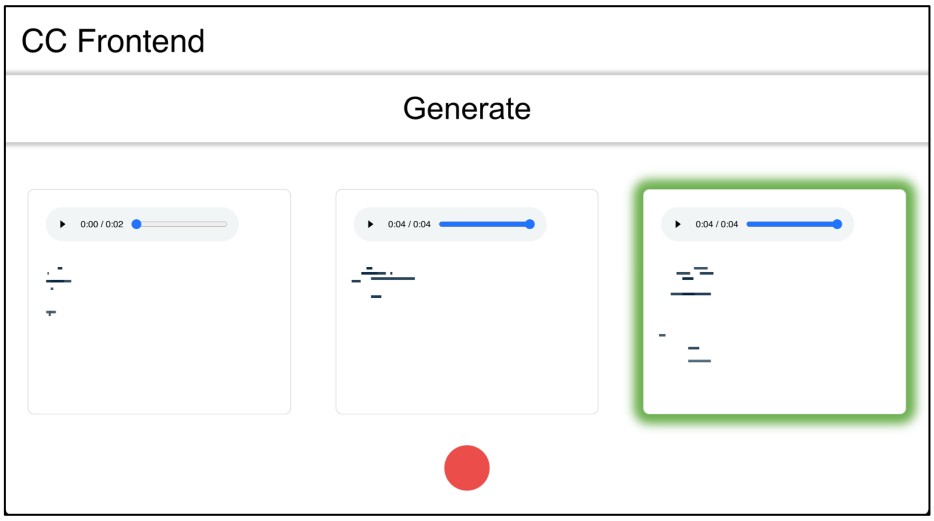


Figure 4: GUI running in a web browser

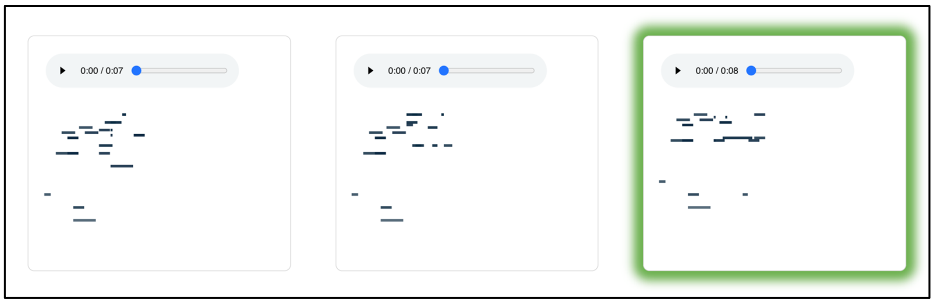


Figure 5: The containers after running one iteration with conditioned input

5 Experimental Study

To evaluate the model's performance several configurations with regard to the network topology and the employed training data were created. Data used during the experiments was dynamically split up into *train* and *validation* datasets, using 10% of the overall data for the validation set. The original baseline configuration was taken from [9], and subsequently adapted in various ways to create the other configurations. Parameters and hyperparameters of the configurations are given in table 3. Each configuration was run until the validation loss reached a certain lower threshold or stopped early if the validation loss did not decrease.

In total, six experiments were run on a high-end NVIDIA 2080Ti GPU using the school-internal compute servers. The goal of the experiments was to find a configuration that would minimise the low loss value of the validation dataset. This would indicate that the model had learned to generalise from the observations in the training dataset and increase its likelihood of performing well on a set of completely unknown input data (such as user recorded MIDI input).

Configuration	Num. train segments	Num. epochs	Learning rate	Num. attention heads	Num. attention layers	Dimension of inner ff layer	Dropout probability
C1 (b)	400	200	0.00005	8	12	2048	0.1
C2	100	200	0.0001	8	12	2048	0.1
C3	400	200	0.0001	6	8	1024	0.3
C4	400	200	0.00005	6	8	1024	0.3
C5*	200	200	0.00005	8	12	2048	0.1
C6*	800	50	0.00005	8	12	2048	0.1

Table 3: Training configurations
*** … encoder and decoder biases tied in addition to weights**

In addition to evaluating the model in different configurations, the musical outputs it produced were evaluated for one specific configuration (C1). This configuration was chosen because it was trained on a large amount of data. The musical outputs were evaluated in terms of their rhythmic and harmonic coherence.

6 Results

The results of the four experiments are given in table 4, showing average training losses (blue) and validation losses (orange) by epoch for each experiment. Due to unsatisfactory validation loss values during training, all six conditions were stopped early. In all experiments the model started overfitting to the data from epoch 5 onwards.

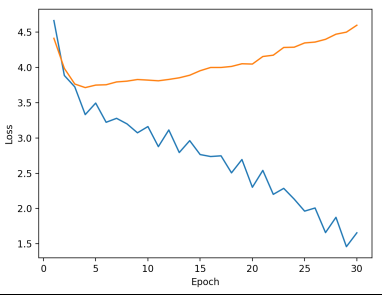
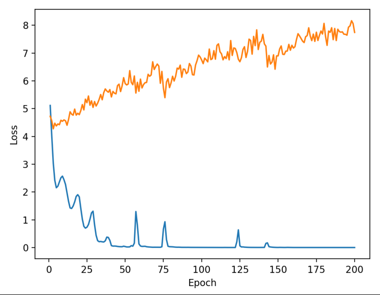
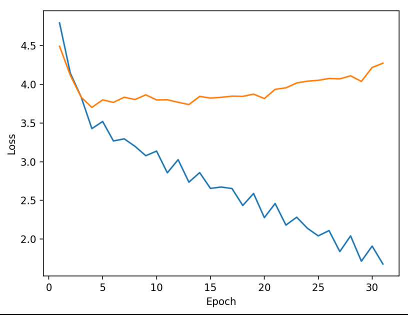
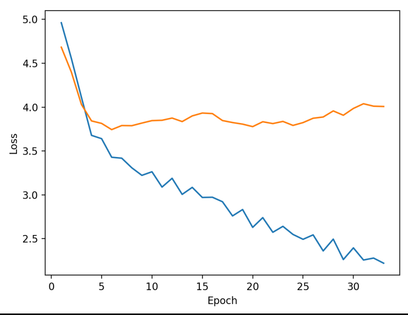
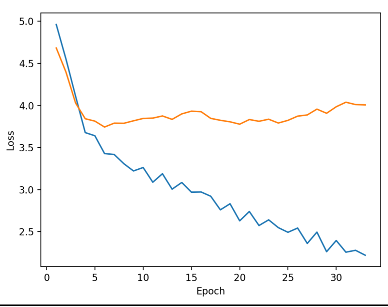
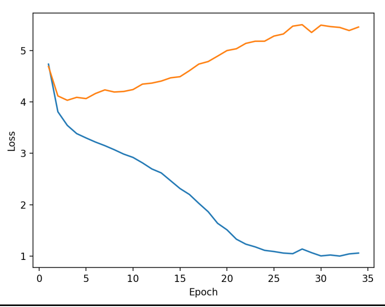
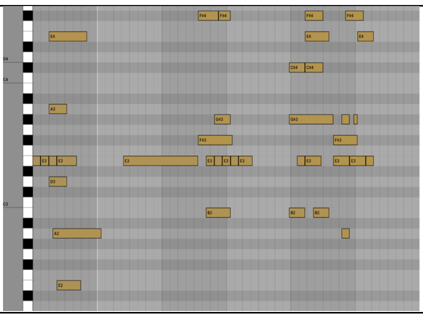
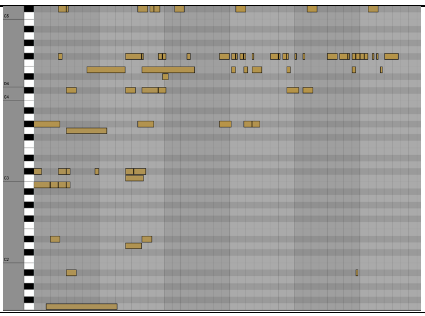
	
C1 (baseline)	**C2**
	
C3	**C4**
	
C5	**C6**

Table 4: Average training losses (blue) and validation losses (orange) per epoch for the configurations

In total, six recordings were made using the MIDI outputs of the model (located in the "*outputs*" folder in the submission bundle). Three of them were generated from scratch (***FS*** condition), i.e. using a random prompt initially. The remaining three were generated using a user-defined MIDI prompt (***UG*** condition). The resulting MIDI piano rolls are shown in table 5.

The results demonstrate that the model has learnt a notion of continuity but lacks understanding of rhythmic context. All example outputs contain noticeable gaps between each bar of generated MIDI notes. This suggests that the tokens responsible for note positions and bar start/end indicators have not been sufficiently learnt. However, the pitches of produced notes in all outputs proved to be coherent, which indicates that the model has learnt pitch-relationships between notes to a degree where it is capable of relating them both intra-bar and over several bars. This is most apparent in the FS conditions.

The UG conditions particularly show the effects of the overfitted model. In both the UG1 and UG2 condition the model does not relate its generated sequence to the user prompt. In UG3, a single-note prompt was used instead of a chord/sequence. This resulted in a more coherent continuation.

	
From scratch (FS) 1 (4)	**FS2 (4)**

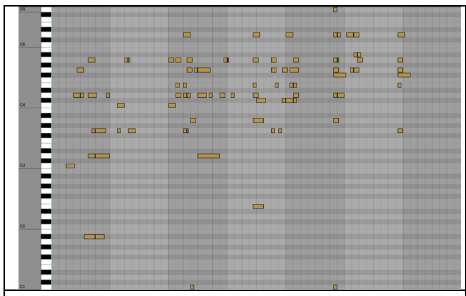
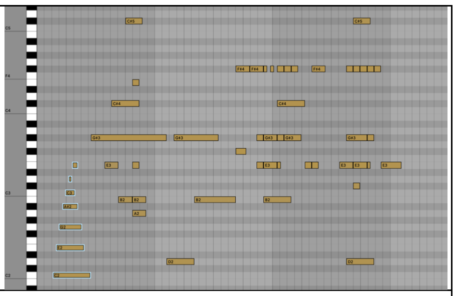
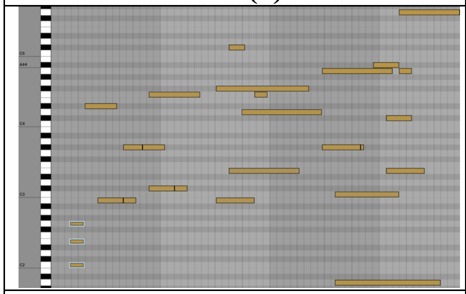
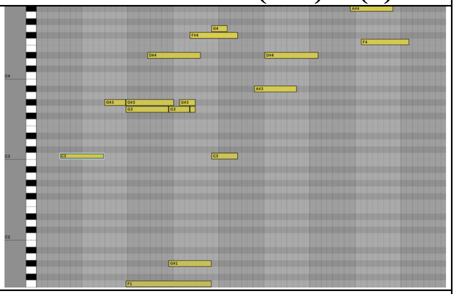
	
FS3 (4)	**User Generated (UG) 1 (4)***
	
UG2 (4)*	**UG3 (2)***

Table 5: Screenshots of the produced MIDI sequences, number of feedback-compose cycles given in parentheses
*** MIDI notes from user prompt are outlined in blue**

7 Discussion

I employed the FACE evaluation framework proposed by Colton et al. [10] to evaluate the system in terms of its creative capabilities. Currently, the system only operates in the first set of the framework (superscript *g*), since it does not produce methods for generating artefacts but only artefacts on their own.
Firstly, the system can be classified as a weak E^g (generating expressions of a concept) system – weak, because the expressions (MIDI data) can vary, however the concept (pop piano music) is static. In other words, in the current implementation the concept of pop piano music is intrinsic to the system – something that could be remedied with larger amounts of data.
Secondly, the system also falls into the C^g category (generating concepts), since the resulting MIDI files can be used as a template for further work. The justification for this claim is found in a human

encounter – during showcasing my work to a colleague, I was asked to send them the generated MIDI outputs so that they could use them in their own compositions.
Based on these observations I classify the system as $\langle C^g, E^g \rangle$.

The second lens through which I analysed the system is the idea of the Meta Mountain. In most aspects, the system is situated in level 1 – a generative system that produces interesting outputs. However, during experimenting with the trained model, I discovered one aspect that I linked to level 4 (persuasive systems): The model's tendency to confidently generate (mildly) dissonant music. It should be noted here that the dataset used to train the model contains mostly simple, consonant pieces of piano. During my experimentation with the model I realised that I had inadvertently primed my expectations of the outputs through the consonant songs contained in the training data – expecting the model to generate equally harmonious outputs. As I was generating more and more outputs however, I realised that the dissonant outputs were *systematically* dissonant to a certain degree. I interpreted this apparent shortcoming as an interesting feature instead, from which I deduct that models such as this one possess at least some capacity for persuading users of their results.

8 Conclusions and Future Work

I have presented a system for interactive human-machine composition. The system is able to produce harmoniously coherent pieces of symbolic pop piano music from random prompts, and to a lesser extent from user prompts. I link the inability of the model to produce satisfying continuations from user prompts to the problem of overfitting to the data during the training process. However, even in the current configuration, the system is capable of creating interesting pieces through the composition-evaluation-feedback-composition cycle in conjunction with a human user.
Future work could address the issue of overfitting, for example by gathering a larger dataset or further tuning the model parameters. At a later stage, the backend could be connected to digital audio workstations or digital notation systems to inject the composed music directly into the working environment of musicians. Combining this integration with an

extension of the system to a fully fulfil the E^g FACE classification would result in an exciting new method for stylistically diverse co-creation between musicians and machines.

References

[1] F. Pachet, 'The Continuator: Musical Interaction With Style', *Journal of New Music Research*, vol. 32, pp. 333–341, Aug. 2010, doi: 10.1076/jnmr.32.3.333.16861.

[2] A. Vaswani *et al.*, 'Attention Is All You Need', *arXiv:1706.03762 [cs]*, Dec. 2017, Accessed: May 03, 2021. [Online]. Available: http://arxiv.org/abs/1706.03762.

[3] I. Sutskever, O. Vinyals, and Q. V. Le, 'Sequence to Sequence Learning with Neural Networks', *arXiv:1409.3215 [cs]*, Dec. 2014, Accessed: May 03, 2021. [Online]. Available: http://arxiv.org/abs/1409.3215.

[4] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, 'Convolutional Sequence to Sequence Learning', *arXiv:1705.03122 [cs]*, Jul. 2017, Accessed: May 03, 2021. [Online]. Available: http://arxiv.org/abs/1705.03122.

[5] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, 'Multi-task Sequence to Sequence Learning', *arXiv:1511.06114 [cs, stat]*, Mar. 2016, Accessed: May 03, 2021. [Online]. Available: http://arxiv.org/abs/1511.06114.

[6] M. Chablani, 'Sequence to sequence model: Introduction and concepts', *Medium*, Jun. 23, 2017. https://towardsdatascience.com/sequence-to-sequence-model-introduction-and-concepts-44d9b41cd42d (accessed May 03, 2021).

[7] D. S. Odaibo, 'Attention Mechanisms in Deep Learning — Not So Special', *Medium*, Apr. 09, 2020. https://medium.com/retina-ai-health-inc/attention-mechanisms-in-deep-learning-not-so-special-26de2a824f45 (accessed May 03, 2021).

[8] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, 'Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context', *arXiv:1901.02860 [cs, stat]*, Jun. 2019, Accessed: Apr. 27, 2021. [Online]. Available: http://arxiv.org/abs/1901.02860.

[9] Y.-S. Huang and Y.-H. Yang, 'Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions', *arXiv:2002.00212 [cs, eess, stat]*, Aug. 2020, Accessed: Mar. 17, 2021. [Online]. Available: http://arxiv.org/abs/2002.00212.

[10] S. Colton, J. Charnley, and A. Pease, 'Computational Creativity Theory: The FACE and IDEA Descriptive Models', 2011.