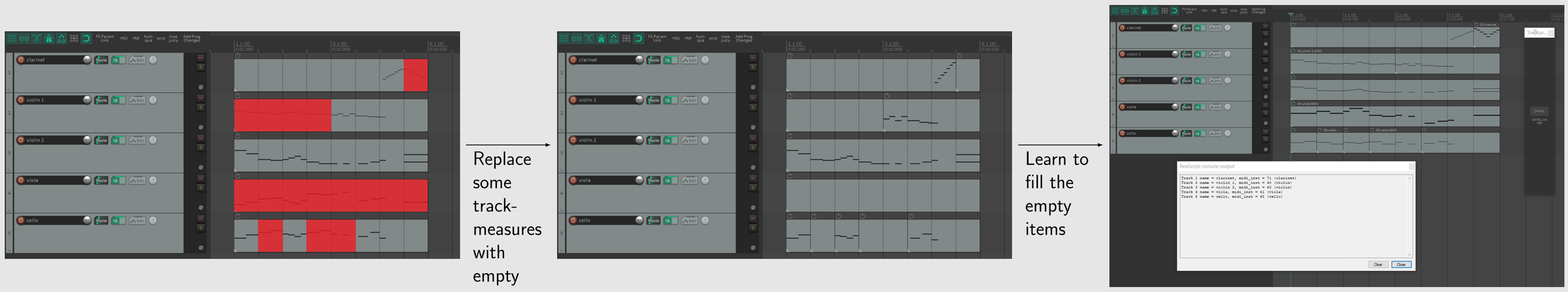
Composer's Assistant: An Interactive Transformer for Multi-Track MIDI Infilling

Martin E. Malandro [1]

[1]Sam Houston State University

Sam Houston State University
MEMBER THE TEXAS STATE UNIVERSITY SYSTEM

Problem: Multi-track MIDI infilling



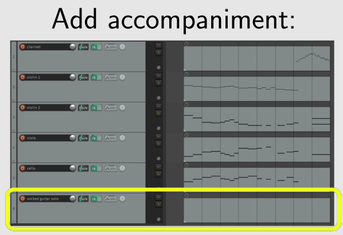
Motivation and Approach

Motivation: Create a tool that composers can (and want to) use
Approach:

- T5 [4]-like encoder-decoder transformer model with DAW integration
- Custom token-based language that allows for straightforward masking of track-measures
- Novel data pipeline:
  - Code to dedupe a dataset of MIDI files. (ex: 42% of Lakh MIDI dataset files in the "f" folder also exist in other folders.) Deduping code is based on musical content—specifically, coarsely quantized note onset chromagrams, transposed to all 12 keys. This helps catch duplicate pieces of music that differ in grid resolution and/or key.
  - Code for identifying and removing "free-flowing performance" MIDI files (which do not respect the underlying grid) from the dataset. This helps prevent the model from learning to produce mis-quantized outputs.
    To perform this filtering, given a MIDI file M, we quantize the note onsets in M to a resolution of 12 ticks per quarter note, and we form a length-12 vector v_M whose ith entry ($i \in \{0, \dots, 11\}$) is the number of note onsets in M occurring i ticks after a grid quarter note. The idea is that if the note onsets in M have nothing to do with the grid, then v_M will point in a similar direction to the uniform vector $v_1 = (1, \dots, 1) \in \mathbb{R}^{12}$. We therefore compute the cosine of the angle θ_M between v_M and v_1 and we declare a threshold T such that when $\cos(\theta_M) > T$ we remove the file M from our dataset. Hand exploration indicated that $T = 0.8$ was a reasonable threshold.
  - Code for identifying and removing "delay" tracks within MIDI files:
    Given tracks T_1, T_2 within a MIDI file, we define an "overlap measure" $O(T_1, T_2)$ that measures the percentage of note intervals in the larger of the two tracks accounted for by the note intervals in the smaller. We use a threshold of 0.9 for asserting near-overlap between two tracks. As we go through the tracks in a MIDI file in order, a later track T is thrown out if there exists an earlier track T_0 using the same instrument such that some shift T_s of T of no more than a half note has $O(T_0, T_s) \geq 0.9$.
- Copyright free/permissively-licensed training set: See acknowledgments at our github.
- Quantization that is sufficiently fine for score generation: Our grid accommodates 32nd notes and 16th note triplets.
- Code supports finetuning by end users: A video card with 6 GB RAM can train on inputs and outputs of length ≤ 1024, and a video card with 12 GB RAM can train on inputs and outputs of length ≤ 1650. Several composers have personalized CA for themselves this way.

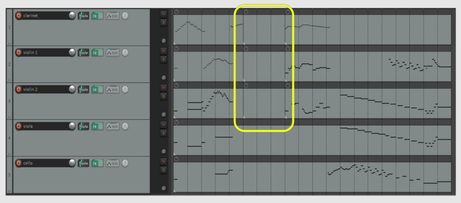
Many Useful Applications

Add accompaniment:

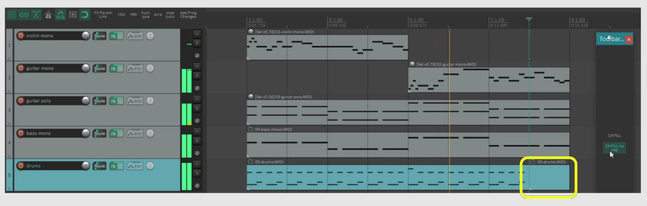


Continue a song (MuseNet [5]-style):

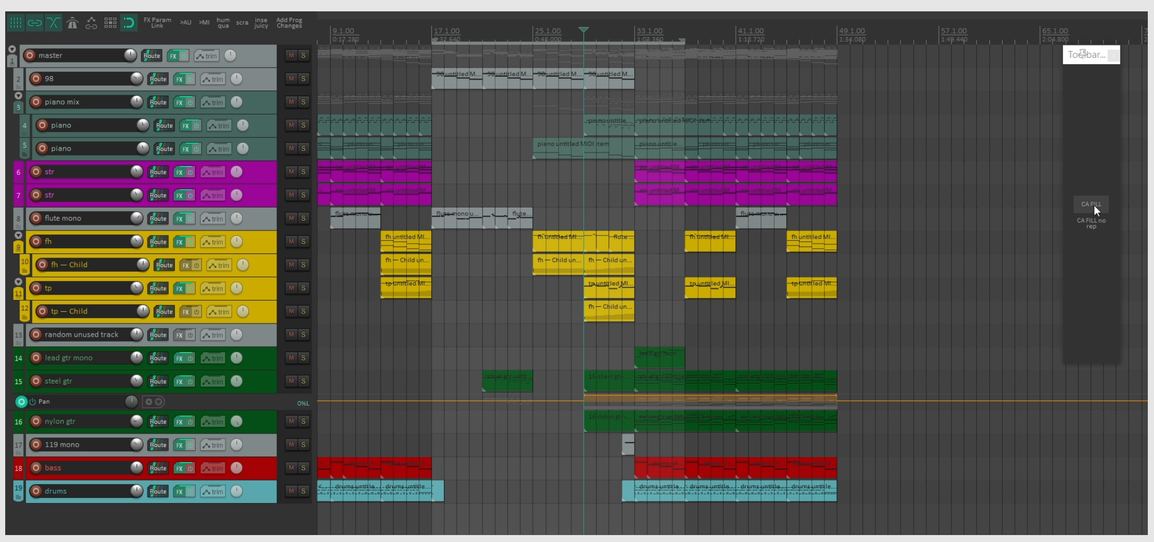
Connect two parts of a song:



Add a drum fill:



Large Project? No Problem.



Feature Comparison with Work by Other Authors

	Composer's Assistant	MMM [2, 3] (Colab)	MusIAC [1] (Colab)
# bars	any	4 or 8	16
# tracks	any	12 or 6	3
Time signatures allowed	1/16 thru 8/4	4/4	4/4, 3/4, 2/4, 6/8
Quantization	Mixed grid; supports 32nd notes and 16th note triplets	32nd note triplets (which also supports 16th notes and 16th note triplets, but not 32nd notes)	16th notes
Training Set	Copyright-free / permissively licensed	Lakh	Lakh
User controls	Mono/poly switches	Note density	5 (3 track-level, 2 bar-level)

Model Evaluation

Task	CA	CA (no mono/poly)	MMM-8	MMM-4
Note F_1 results. Higher is better.				
8-bar random infill	**0.5414** $\pm$ $(0.1887)^a$	0.5315 $\pm$ $(0.1904)^b$	0.4153 $\pm$ $(0.1819)^c$	0.4025 $\pm$ $(0.165)^d$
16-bar random infill *	**0.5771** $\pm$ $(0.1661)^a$	0.5705 $\pm$ $(0.1669)^b$	0.4133 $\pm$ $(0.1534)^c$	0.4059 $\pm$ $(0.139)^d$
8-bar track infill	**0.179** $\pm$ $(0.1902)^a$	0.1634 $\pm$ $(0.18)^b$	0.1063 $\pm$ $(0.1573)^d$	0.1427 $\pm$ $(0.164)^c$
16-bar track infill	**0.1773** $\pm$ $(0.1752)^a$	0.1609 $\pm$ $(0.165)^b$	0.1107 $\pm$ $(0.1383)^d$	0.1467 $\pm$ $(0.148)^c$
8-bar last-bar fill	0.5019 $\pm$ $(0.2719)^a$	**0.5063** $\pm$ $(0.2751)^a$	0.4329 $\pm$ $(0.2445)^b$	0.3756 $\pm$ $(0.228)^c$
16-bar last-bar fill *	**0.5415** $\pm$ $(0.2853)^a$	0.539 $\pm$ $(0.2823)^a$	0.4338 $\pm$ $(0.2468)^b$	0.3818 $\pm$ $(0.229)^c$
Pitch class histogram entropy difference results. Lower is better.				
8-bar random infill	**0.2845** $\pm$ $(0.1627)^a$	0.2948 $\pm$ $(0.1597)^b$	0.3045 $\pm$ $(0.1561)^c$	0.3049 $\pm$ $(0.149)^c$
16-bar random infill	**0.2691** $\pm$ $(0.1325)^a$	0.2797 $\pm$ $(0.1326)^b$	0.3093 $\pm$ $(0.124)^c$	0.3063 $\pm$ $(0.113)^c$
8-bar track infill	0.3933 $\pm$ $(0.3032)^c$	0.42 $\pm$ $(0.3134)^d$	**0.2864** $\pm$ $(0.2966)^a$	0.3021 $\pm$ $(0.251)^b$
16-bar track infill	0.3842 $\pm$ $(0.2654)^c$	0.3995 $\pm$ $(0.2763)^c$	**0.284** $\pm$ $(0.2348)^a$	0.3036 $\pm$ $(0.207)^b$
8-bar last-bar fill	**0.3018** $\pm$ $(0.2661)^a$	0.3072 $\pm$ $(0.2692)^a$	0.3213 $\pm$ $(0.2602)^b$	0.3439 $\pm$ $(0.277)^c$
16-bar last-bar fill *	**0.2851** $\pm$ $(0.2652)^a$	0.2925 $\pm$ $(0.2672)^a$	0.3209 $\pm$ $(0.2619)^b$	0.3454 $\pm$ $(0.274)^c$
Groove similarity results. Higher is better.				
8-bar random infill	**0.9534** $\pm$ $(0.0298)^a$	0.9519 $\pm$ $(0.0306)^b$	0.9333 $\pm$ $(0.0369)^c$	0.9314 $\pm$ $(0.036)^d$
16-bar random infill *	**0.956** $\pm$ $(0.027)^a$	0.9552 $\pm$ $(0.0275)^b$	0.9323 $\pm$ $(0.0337)^c$	0.9317 $\pm$ $(0.03)^c$
8-bar track infill	**0.9115** $\pm$ $(0.0592)^a$	0.9069 $\pm$ $(0.0617)^b$	0.8921 $\pm$ $(0.0695)^d$	0.8987 $\pm$ $(0.062)^c$
16-bar track infill	**0.9113** $\pm$ $(0.0547)^a$	0.9082 $\pm$ $(0.0553)^b$	0.8946 $\pm$ $(0.0561)^d$	0.9011 $\pm$ $(0.053)^c$
8-bar last-bar fill	0.9517 $\pm$ $(0.0414)^a$	**0.9524** $\pm$ $(0.0411)^a$	0.9381 $\pm$ $(0.045)^b$	0.9334 $\pm$ $(0.045)^c$
16-bar last-bar fill *	**0.9544** $\pm$ $(0.0481)^a$	0.9542 $\pm$ $(0.0424)^a$	0.938 $\pm$ $(0.051)^b$	0.9339 $\pm$ $(0.047)^c$

Table 1. Objective infilling summary statistics. All cells are of the form mean $\pm$ (std dev)^s, where s is a letter. Different letters within a row indicate significant location differences ($p < 0.01$) in the samples for that row according to a Wilcoxon signed rank test with Holm-Bonferroni correction. Asterisks ($*$) indicate a significant performance difference ($p < 0.01$) between a 16-bar task and the 8-bar task in the previous row for our model according to a Wilcoxon rank sum test.

	Real Music	Our Model	MMM
1st place	**66**	32	27
Avg rank	**1.664**	2.032	2.304

p-values	Our Model	MMM
MMM	0.0239	-
Real Music	0.0034	$2.3 \cdot 10^{-5}$

Table 2. Subjective results from our listening test.

Selected References

R. Guo et al. "MusIAC: An Extensible Generative Framework for Music Infilling Applications with Multi-level Control". In: *Artificial Intelligence in Music, Sound, Art and Design. EvoMUSART 2022. Lecture Notes in Computer Science*. Vol. 13221. Springer, Cham, 2022, pp. 341–356.

J. Ens and P. Pasquier. "MMM : Exploring Conditional Multi-Track Music Generation with the Transformer". In: *arXiv preprint arXiv: 2008.06048* (2020).

J. Ens and P. Pasquier. "Flexible Generation with the Multi-Track Music Machine". In: *Extended Abstracts for the Late-Breaking Demo Session of the 21st Int. Society for Music Information Retrieval Conf.* Montréal, Canada, 2020.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67.

C. Payne. *MuseNet*. `openai.com/blog/musenet`. Apr. 25, 2019.