Adaptive Drum Machine Microtiming with Transfer Learning and RNNs

Grigore Burloiu UNATC Bucharest

`rolypoly~`: adaptive microtiming

We introduce rolypoly~, a drum machine that is:
- lightweight - fast enough to run in real time;
- audio-interactive - not simply reactive to incoming [target] sound, but "listens" to the coupling of self + target;
- progressive & scalable - learns a piece from the ground up;
- score-agnostic - does not need to know the parts played by other musicians.

INPUT: each drum hit + location encoding +
+ drum-target onset diff for preceding hit
OUTPUT: drum microtiming offset for current hit

Every performance produces a sequence of "label" output values, to be trained on for the next iteration. Find out more in Fig. 2 (right).

research question

Can a machine learn to groove over a piece by adapting to its human partner(s) real-time input?

implementation: pyTorch + Max

Max does the audio analysis, sends onset diff values via OSC to Python, where a script parses the drumtrack MIDI file through the pyTorch deep learning model, and sends out timing values back to Max, which outputs the audio.

The current Max interface is shown in Fig. 1 below.

model architecture

We propose two models. The first is built on a 2-layer uni-directional LSTM network. The second is a simplification of the Seq2Seq architecture in Magenta's GrooVAE. Both models are pre-trained on Magenta's Groove MIDI Dataset of expressive drumming. Source code and trained models are available online.

Figure 1: The current configuration of the Max-based GUI. The controls are automated via OSC from Python. From left to right, top-down: reset button and playback toggle; next hit display (timing & notes); timing history, drum-target diff history, input audio selector, monitoring and level history.

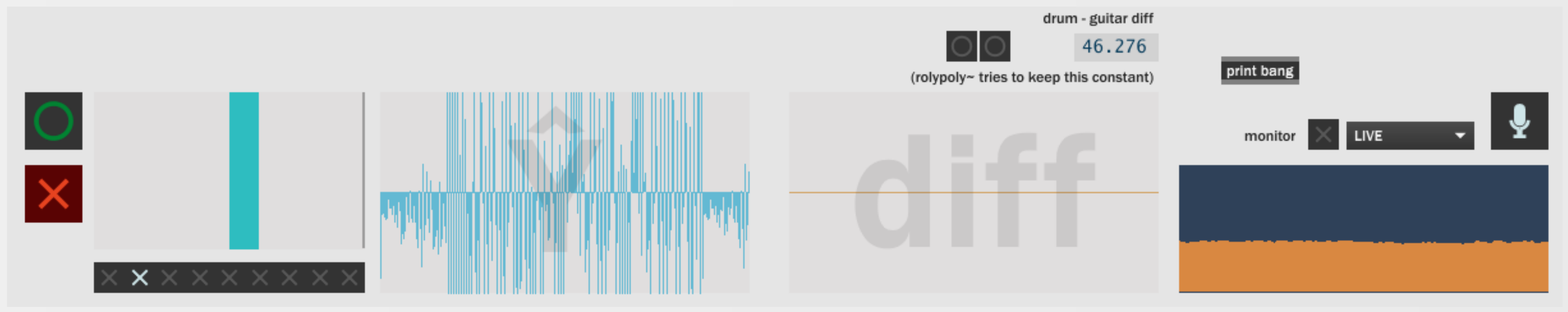
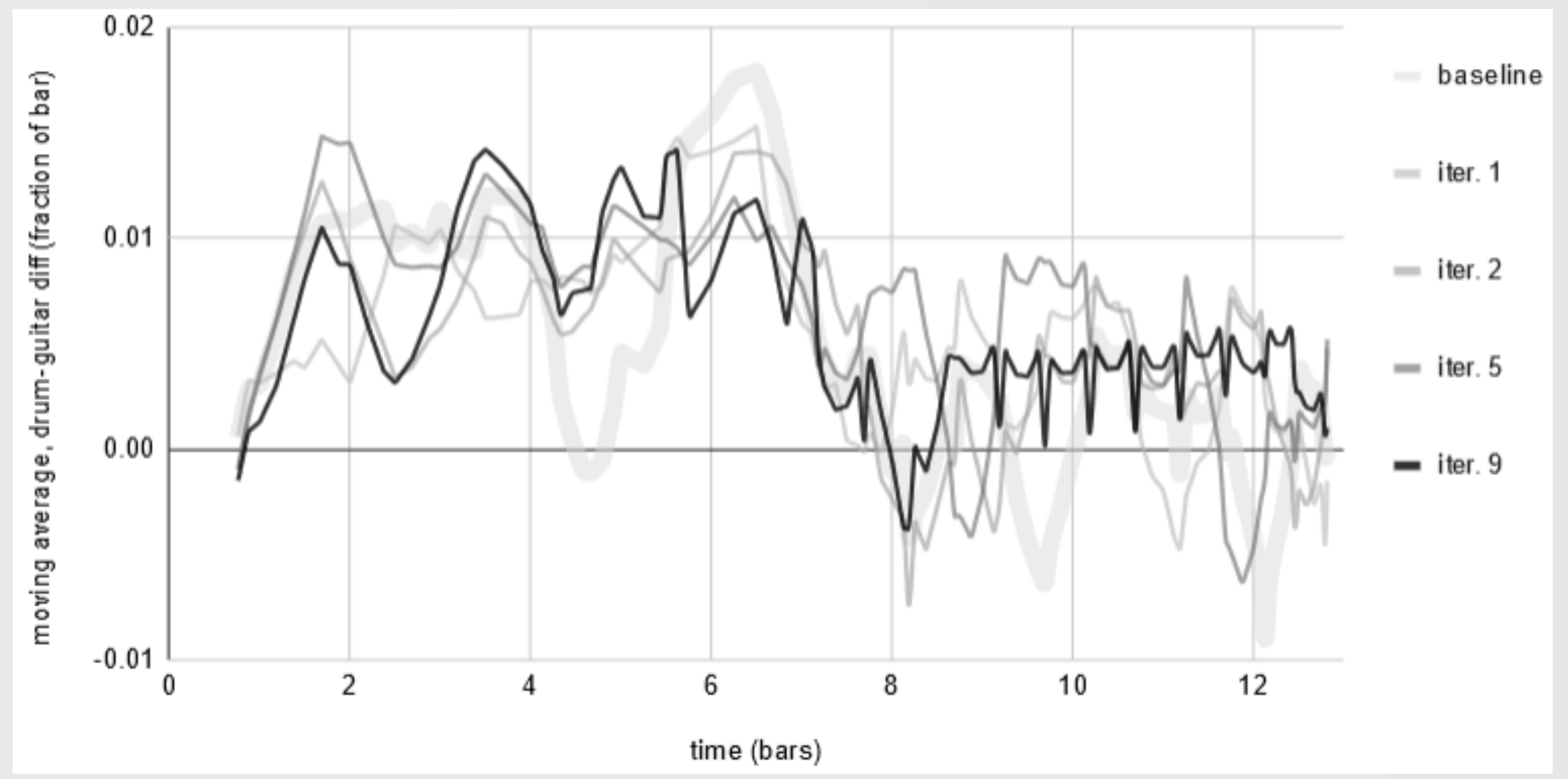


Figure 2: Moving average (period: 6 hits) of the timing difference between drum hit and guitar onset. The baseline (thick, light gray) is the basic LSTM model pretrained on GMD. Subsequent training runs for 5 epochs (Adam w/ l.r. 0.001) after each take.

This is the typical use case, where a song is performed multiple times and the basic LSTM model learns incrementally after each take. We plot here the evolution of drum-target offsets over several iterations. The agent is able to "tame" the variance of the guitar, and visibly adapts to structural patterns in the piece.



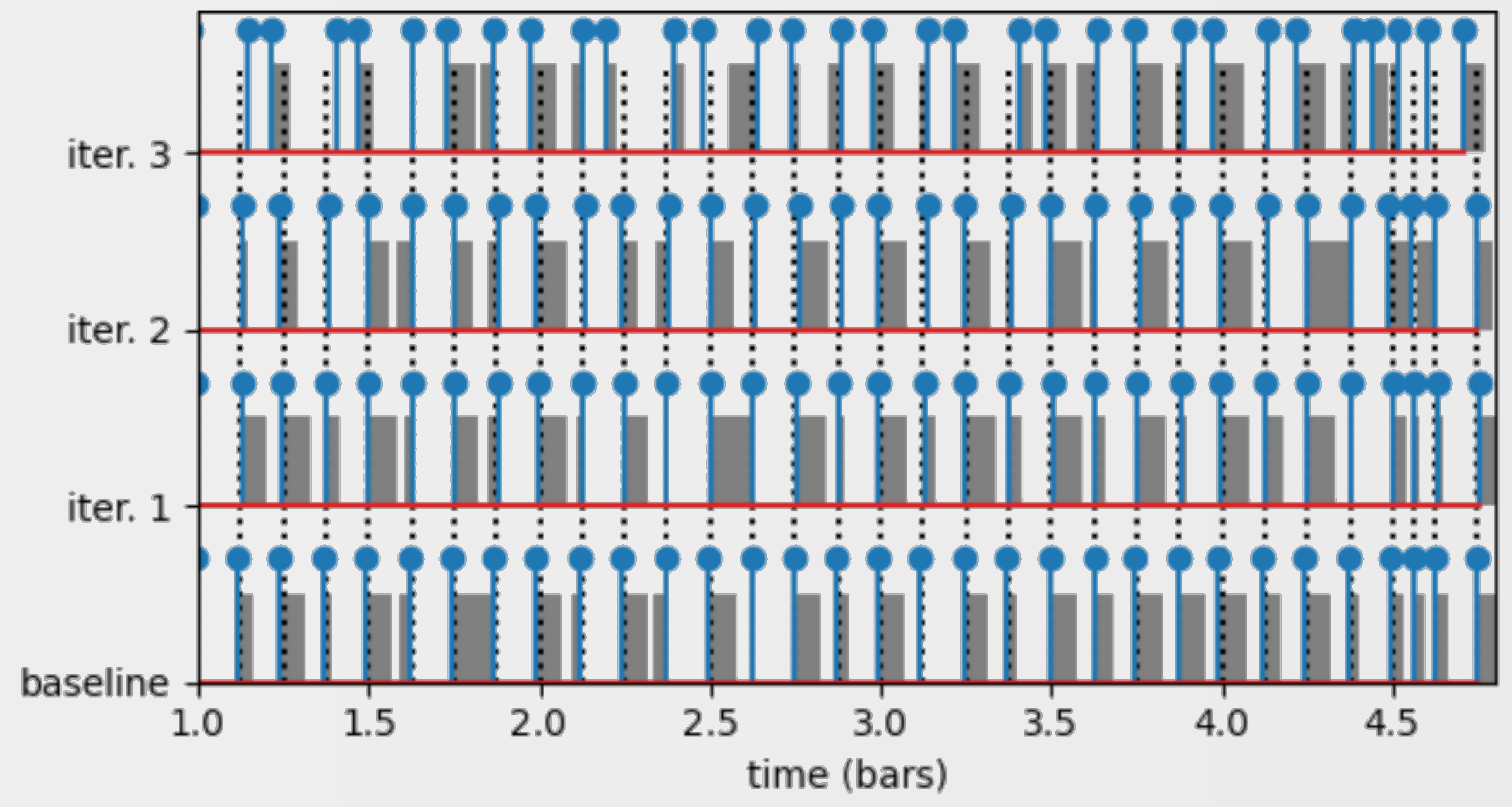


Figure 3: Rhythm morphing over three iterations.
Score: dotted lines.
Output: blue stems.
Drum-guitar diff: grey bars.
The transition occurs from straight (bottom) to swing (top).

The pattern resulting in iter. 3 shows drum hits grouped together two-by-two, with a reduced overall drum-guitar diff variance.
Note: this is not the typical intended use case of the system, but rather an experiment. We show that the system is able to drastically modify the feel of a beat, but the process would be more practical and aesthetically pleasing if the drum track was notated in a triplet feel to begin with.

upcoming work

Currently: modelling velocity, and porting the inference code from Python to a Max external. Next: the strategy, rather than to "solve" micro-timing, is to build a multi-timescale hierarchical system, with agents on different timescales (tempo, phrase, section, piece, beyond) informing each other and interacting.

open questions

We contributed a process for defining labels after each performance, which aims to minimise drum-target offset variance. Is this optimal / musically useful?
Possible research directions include:
- trying out different architectures e.g. Transformers
- exploring reinforcement learning-based solutions
- exploring one-shot learning over existing multitrack recordings
- adding audio representations via learning of latent feature space of descriptors
- replace onset detection with other just-in-time algorithms (e.g. audio alignment)
- multimodal representations to augment anticipation performance

experiment: swing induction

Fig. 3 shows the Seq2Seq model transitioning from a straight 4/4 beat to a "swing" shuffle, where offbeats are pushed slightly later. This is achieved simply by "swinging" the respective notes on guitar over three iterations.

`grigore.burloiu@unatc.ro`
`github.com/RVirmoors/rolypoly/`

ISMIR LBD Oct.2020