Dicy2 for Ableton Live

interacting with generative audio agents

Version document 0.1 (draft)

Dicy2 by Jérôme Nika, Augustin Muller, Joakim Borg, Matthew Ostrowski
Design and implementation Max for Live plugin: Manuel Poletti
ANR-DYCI2; ANR-MERCI; ERC REACH dir. by G. Assayag; Ircam UPI-CompAI
Redaction document: Jérôme Nika

December 8, 2022

Contents

1 Dicy2 Max for Live devices

1.1 Introducing Dicy2

Dicy2 for Live is an Ableton Live plugin using machine-learning for the interactive generation of sequences in musical relation with the real-time analysis of an incoming audio stream. It can be integrated into musical situations ranging from the production of structured material within a compositional process to the design of autonomous agents for improvised interaction. It is available as a plugin for Ableton Live and a library for Max.

To discuss **Dicy2 for Live** features, use the Forum discussion groups at https://discussion.forum.ircam.fr/c/dicy2-for-live/.

1.2 Getting ready

1.2.1 Requirements

- Mac OS >= High Sierra
- Ableton Live >= 11. To use this plug-in developed with Max For Live, you must have Ableton Live Suite or Ableton Live Standard and the Max For Live extension.

1.2.2 Installation

Drag the Ableton Live demo session embedding the device somewhere in your Ableton Live path (e.g. in "Music/Ableton/User Library").

Warning In order to use Dicy2 for Live, you need three items that must be in the same sub-folder:

- the device `dicy2.agent.amxd`
- the device `dicy2.server.amxd`
- the application `dicy2.server.app`

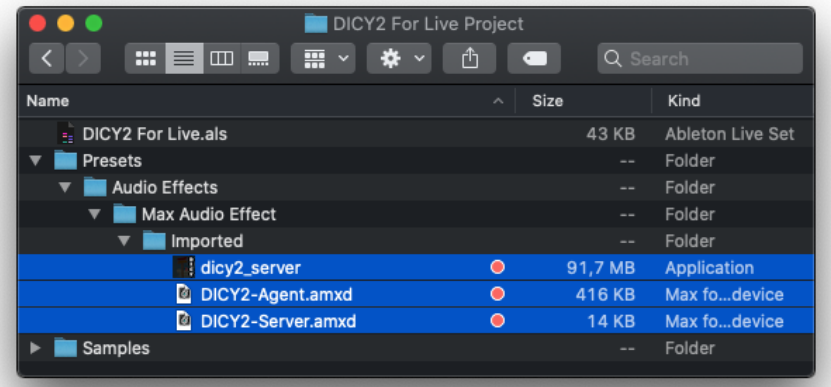

Figure 1: The 3 elements you need to use Dicy2 for Live.

Warning The first time you use Dicy2 for Live, you will probably get a pop-up message asking you to allow `dicy2.server.app` to run. Click OK.

1.2.3 Play

A demo Ableton Live session is provided with Dicy2 for Live: by soloing a track and starting the playback of the sound file at the top of the stack that serves as guiding input, you will hear 4 different behaviors of Dicy2 among those you can compose yourself.

The Dicy2 agent has a behavior defined by all its parameters and will generate sound by drawing audio segments from its "memory": an audio file that has been analyzed according to your parameters.

In these examples we use other files to guide the generation (e.g. a saxophone file guiding a guitar, a double bass, a voice...), but of course we invite you to use a live audio input as guiding input !

Video tutorials are available on Ircam's Youtube channel.

2 A Live session using Dicy2 for Live

Dyci2 is a plugin for generative real-time interaction and composition. The basis material, called the Memory, is used to train the `Dicy2.agent` device, which uses machine learning techniques to create an internal map of temporal relationships within the Memory. An audio guiding input send queries to the agent, and Dicy2 returns segments of musical material in response.

Dicy2's generative core runs in a background application. The `Dicy2.server` device communicates with this application from your Ableton Live Session. So, in order to use Dicy2 for Live, you need to have one **- and only one -** `Dyci2.server` device in your session that will allow you to use as many `Dyci2.agent` devices as you want.

Figure 2: You should have one and only Dicy2.server device placed on any track (for example the master).

This `Dyci2.server` device can be put on any track, for example the master, and a green light turns on when it is connected and the `Dyci2.agent` devices can be used.

3 Tuning your agent with the Analysis tab

With the `Dyci2.agent` device, an audio Memory is built automatically, and generation from this Memory is guided by scenarios derived from an analysis of live audio input. The parameters of the various objects involved in the chain are used to compose the "behavior" of a Dicy2 agent.

3.1 The analysis tab

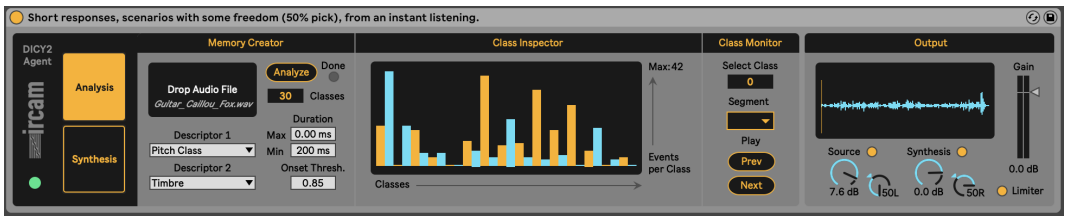

Figure 3: Dicy2 agent: the Analysis tab.

The first tab of the `Dyci2.agent` device allows to give parameters to build a Memory automatically and specify the musical dimensions on which the Memory and the guiding audio input will be connected.

3.2 Memory creator

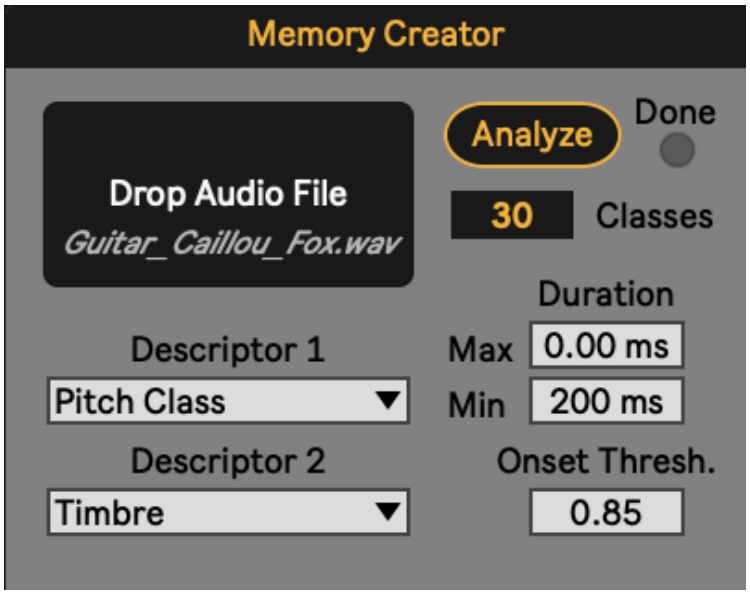

Figure 4: Dicy2 agent: create a memory from a file.

Audio file The first step is to choose which audio file we want to use to build a Memory in which we will navigate guided by the audio guiding input.

Descriptors Then, we can choose one or two musical dimensions - or descriptors - that will be used to analyze the memory and to search later for correspondences with the audio guiding input.

Segmentation Before being analyzed using the selected descriptors, the Memory is automatically segmented. The settings in the right column are default settings that will work in most cases, but expert users can modify them, especially the minimum and maximum duration of an audio segment (NB: 0 means "no maximum duration"). For the value of `Onset Thresh.` we invite expert users to refer to the help of the `pipo` library developed by the ISMM team at Ircam, and in particular `onseg`.

Classes Once the segmentation module has sliced the soundfiles into segments, and assigned each segment a set of descriptor values, this module looks at the descriptor values for each segment, and groups segments with similar descriptors together, assigning each class a numerical label. These are the labels the Dicy2 model will use when receiving and responding to queries. The number of classes you choose to use is very important: the greater the number of classes, the more detailed the analysis will be, but each class will have fewer members (see next section).

Memory Model This analysis of the memory in different classes is then used to build a temporal model using machine-learning techniques, and this will be used to find correspondences with the guiding audio input (tab Synthesis presented in the next section).

Using pattern recognition algorithms, it looks for a 'best match' between the input and the material stored in the Memory, taking into account, musical similarities, the temporal structure of the Memory, and the behavior you composed for your agent.

3.3 Inspecting and monitoring the memory

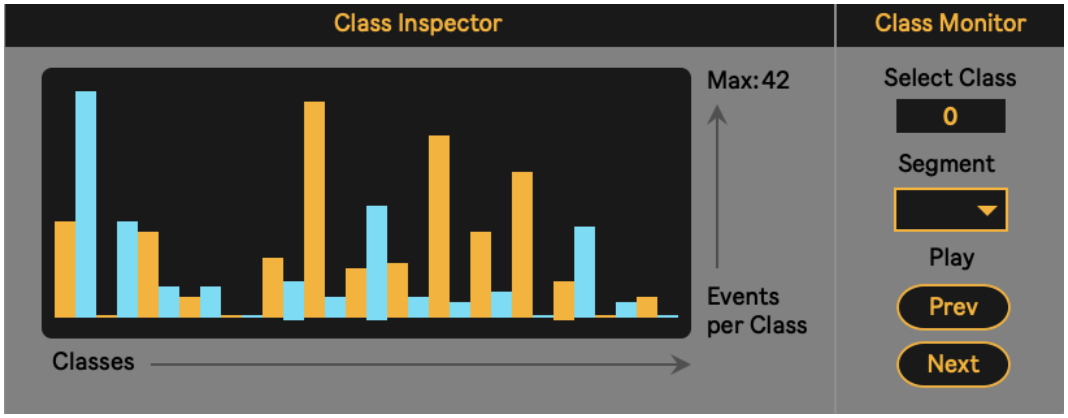

Figure 5: Dicy2 agent: inspect your memory for fine tuning.

The histogram visualization in the "Class Inspector" section allows you to see approximately the distribution of the audio segments making up the memory in the different classes. This visualization as well as the possibility of listening to how the segments are distributed in classes will perhaps allow you to fine-tune your instrument.

Indeed, in the "Class Monitor" section, you can select a class and click several times on "Next" to listen to the population of your different classes.

Hint

- If some classes contain events that are too varied with respect to the chosen analysis descriptors, increase the number of classes, which will increase the precision of the analysis.
- If all the members of a class sound too much alike, decrease the the number of classes to achieve more variety within a class.

4 Define the behavior of your agent with the Synthesis tab

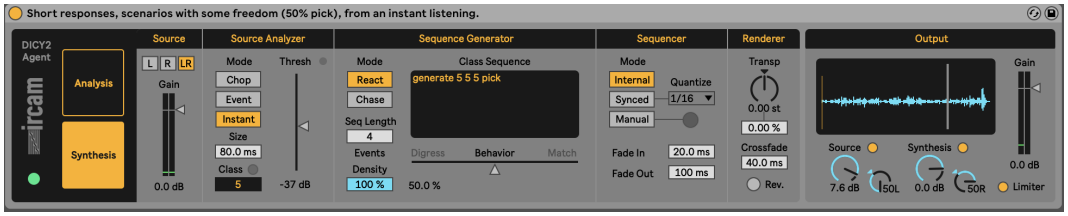

Figure 6: Dicy2 agent: the Synthesis tab.

Once your memory is ready, the second tab of the device allows you to compose the behavior of your `Dyci2.agent` device, and in particular the relation to be established between the guiding audio input and the outputs generated by the agent.

4.1 Analysing the guiding input to react

The function of the "Source Analyzer" is to analyze the guiding audio input and identify on the fly which class of memory it is closest to at that moment ("Class" int box). This identification can then be used to generate scenarios that are passed to the agent, in order to create a musical relationship between an audio stream and the content in the Memory.

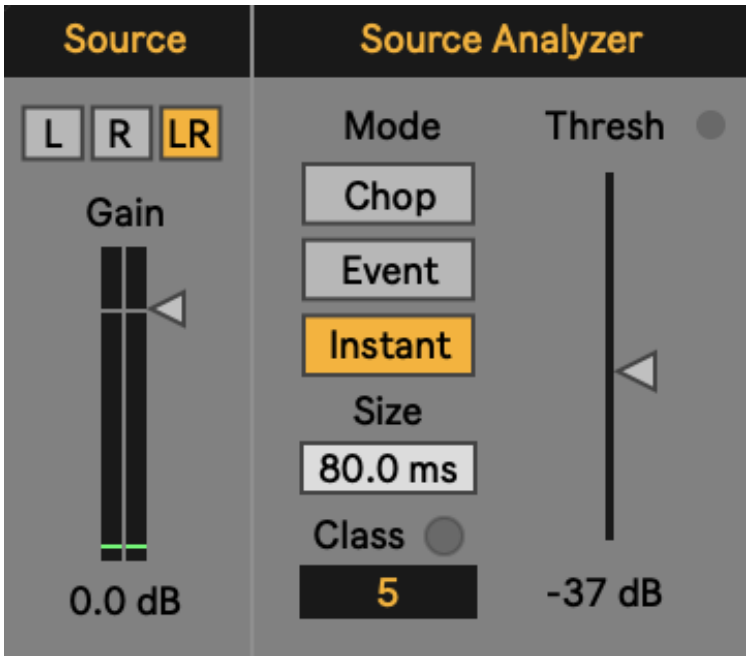

Figure 7: Dicy2 agent: Analysing the guiding input to react.

Threshold The first thing to do is to set the threshold to the right: set a threshold level allowing only events you want analyzed to pass through.

Mode The analysis timing determines the rate and method by which the incoming audio is analysed. There are three possible modes:

- `Chop`: the incoming audio is divided into equal segments, regardless of audio content. The segment size can be set with the `size` parameter. *When you want a continuous analysis or when the source is very legato*.
- `Event mode`: incoming audio is segmented using loudness onsets, and the class is output when the segment is completed.textitOne class per event.
- `Instant mode`: the analyzer looks at the incoming audio stream and analyzes a short time window to estimate the likely timing and class of the next event. The size of this analysis is set by the `size` parameter.*We do not wait for the end of the event to send the class. This will make the agent's reaction seem much more instantaneous*.

4.2 Sequence Generator: Compose the musical reaction

Once the guiding audio input has been analyzed to assign it a class, this class is used to prepare a "generation scenario" to send to the agent, and this is where the most important part of the agent's behavioral composition lies.

From event to sequence The principle of Dicy2 is that the analysis of one segment of the guiding audio input will result in the generation of an entire temporal sequence. We will see below the different ways of constructing these sequences that tell the agent what to generate each time it receives something.

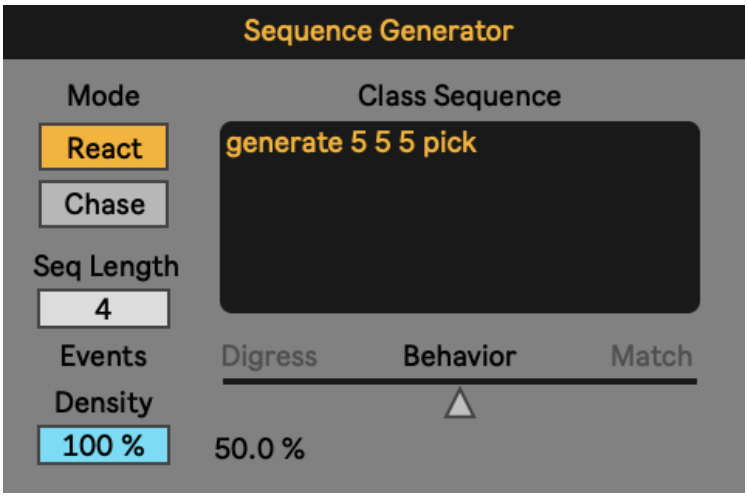

Figure 8: Dicy2 agent: Compose the musical reaction.

Length `Length` (in number of events) of the Class Sequence constituting the order of generation for the agent, and thus of the musical sequence that will be played in response to each new incoming class.

Density Chance to output a sequence for each new incoming class.

Mode Every incoming class is used as a startpoint to generate a new sequence. If the Agent is presented a new Class Sequence before the previous one is finished, the remaining events are discarded.

- `React`: Match the incoming analysed class, then go on matching or let digress for more fluidity / autonomy (depending on the `Behavior` value).

- `Chase`: Presents the last classes received from the analyzer. The number of classes is determined by the `Length` parameter. The generation query can be exactly this Class sequence or allow more more fluidity / autonomy (depending on the `Behavior` value).

Behavior The `Behavior` parameter defines the percentage of "pick" introduced in the Class Sequence. "pick" asks the agent to choose an optimal segment. We leave it to you to experiment to better understand, but you will feel that in `React` mode for example, a `Behavior` introducing a lot of "pick" will define an agent that digresses a lot from the guiding audio stream (`Digress`), while a value that introduces none will define an agent that tries to stick as much as possible to the input, at the risk of being repetitive and "patchwork" (`Match`).

Class sequence Based on this sequence defined by all the parameters descrived above, the Agent will generate optimized lists of events from its Memory model.

4.3 Sequencing and rendering the generated sequences.

Once a sequence is generated, it is then a matter of deciding how it will be played.

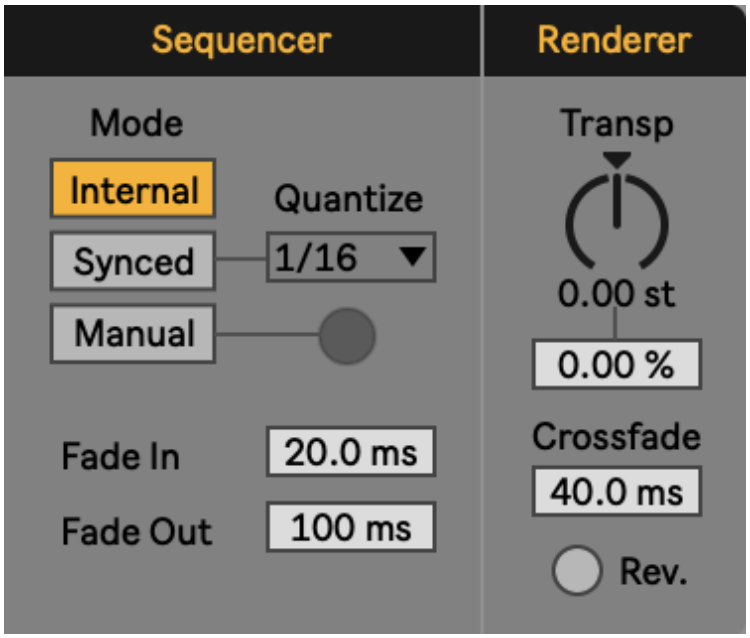

Figure 9: Dicy2 agent: Sequencing and rendering.

Sequencer Mode

- `Internal`: the generated sequence is played continuously.
- `Synced`: the playing of the next grain is synchronized/quantized to the tempo of the session.
- `Manual`: allows the manual triggering of the reading of the grains constituting the generated sequence.

Sequencer Fade In and Fade Out concern the entry and leaving of the generated sequences.

Renderer Crossfade concerns the crossfade between two grains in a generated sequence.

Rev plays each grain of the generated sequence in reverse.

4.4 Settings and visualization of the output.

At the end of the chain, what the `Dyci2.agent` device will play is a mix between the `Source` (audio guiding input) and the `Synthesis`: a sequence

of grains in the memory to play in the order prescribed by the generation.

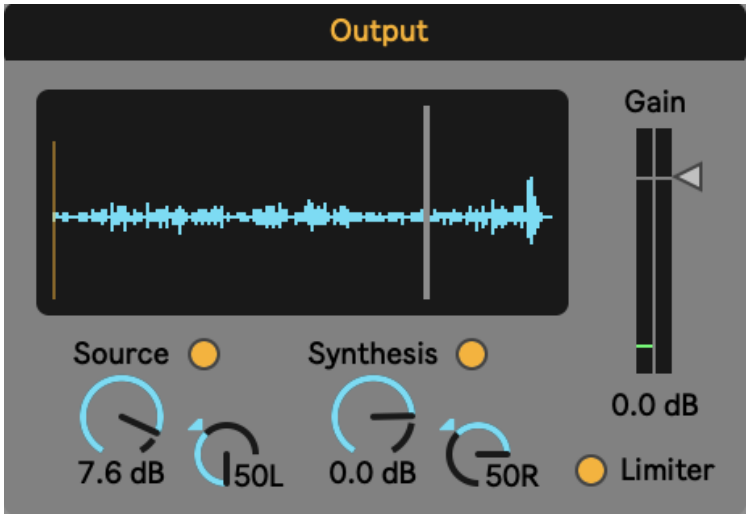

Figure 10: Dicy2 agent: Settings and visualization of the output.

This last panel allows to have a visualization of the output showing which part of the memory is played in real time, and to mix the `Source` and the `Synthesis`.

5 More about Dicy2

5.1 Some references

If using Dicy2, please quote: **Nika, J., Déguernel, K., Chemla, A., Vincent, E., & Assayag, G. (2017, October). Dyci2 agents: merging the" free"," reactive", and" scenario-based" music generation paradigms. In International Computer Music Conference** (1). Additional article references are given in the "References" section (2; 3; 4; 5; 6).

- Video presentation about Dicy2 in French
- Video presentation about Dicy2 in English
- Some videos of collaborations with musicians using Dicy2 or its previous versions

5.2 Authors

Dicy2 is a library for Max and a plugin for Max for Live of the Ircam Musical Representations team, designed and developed by Jérôme Nika, Augustin Muller (Max library), Joakim Borg (Python generative engine / Gig RepMus API), and Matthew Ostrowski (tutorial patchers and videos, abstractions) in the framework of the projects ANR-DYCI2, ANR-MERCI, ERC-REACH directed by Gérard Assayag, and the UPI-CompAI Ircam project.

The audio use cases have been designed and developed with Diemo Schwarz and Riccardo Borghesi, and use the MuBu(7) and CatArt(8) environments of the ISMM team of Ircam. Max4Live plugin by Manuel Poletti. Contributions / thanks: Serge Lemouton, Jean Bresson, Thibaut Carpentier, Georges Bloch, Mikhaïl Malt, Axel Chemla–Romeu-Santos, Vincent Cusson, Tommy Davis, Dionysios Papanicolaou, Greg Beller, Markus Noisternig.

5.3 Artistic collaborations

5.3.1 An instrument designed through artistic productions

Dicy2 integrates scientific and musical research results accumulated through productions and experiments with Rémi Fox, Steve Lehman, the Orchestre National de Jazz, Alexandros Markeas, Pascal Dusapin, Le Fresnoy - Studio National des Arts Contemporains, Vir Andres Hera, Gaëtan Robillard, Benoît Delbecq, Jozef Dumoulin, Ashley Slater, Hervé Sellin, Rodolphe Burger, Marta Gentilucci... After having evolved research prototypes crystallizing the contributions of these various projects for several years, a collaborative work carried out during the year 2022 has led to the finalization of a release of Dicy2 as a plugin for Ableton Live and a library for Max.

5.3.2 Example and tutorial files

This distribution includes agents and sound files from past productions with our friends and collaborating musicians and composers who helped bring Dicy2 to life (courtesy of the artists). Please do not use these agents and files in any context other than these tutorials to respect their work and generosity.

List of files

- – *Doublebass_Perrot_Fox.wav*,
  – *Guitar_Caillou_Fox.wav*,
  – and *Voice_Daumergue_Fox.wav*

were respectively recorded by Alex Perrot, Thomas Caillou, and Manu Daumergue during Rémi Fox's residency at Ircam for the concerts and first album of "C'est pour ça".

- *Balafon_Lehman_ExMachina.wav* was recorded by Steve Lehman for "Ex Machina" with Orchestre National de Jazz.
- *Fox_Sax_1/2/3.aif* come from a performance by "C'est pour ça" at Ircam.

5.4 More

- Please write to `jerome.nika@ircam.fr` and `augustin.muller@ircam.fr` for any question, or to share with us your projects using Dicy2!
- Interested developers can check out the generative core of Dicy2, implemented as a Python library.

References

[1] J. Nika, K. Déguernel, A. Chemla, E. Vincent, G. Assayag, *et al.*, "Dyci2 agents: merging the" free"," reactive", and" scenario-based" music generation paradigms," in *International computer music conference*, 2017.

[2] J. Nika, M. Chemillier, and G. Assayag, "Improtek: introducing scenarios into human-computer music improvisation," *Computers in Entertainment (CIE)*, vol. 14, no. 2, pp. 1–27, 2017.

[3] J. Nika, D. Bouche, J. Bresson, M. Chemillier, and G. Assayag, "Guided improvisation as dynamic calls to an offline model," in *Sound and Music Computing (SMC)*, 2015.

[4] G. Assayag, G. Bloch, M. Chemillier, A. Cont, and S. Dubnov, "Omax brothers: a dynamic yopology of agents for improvization learning," in *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pp. 125–132, 2006.

[5] J. Nika and J. Bresson, "Composing structured music generation processes with creative agents," in *2nd Joint Conference on AI Music Creativity (AIMC 2021)*, p. 12, 2021.

[6] T. Carsault, J. Nika, P. Esling, and G. Assayag, "Combining real-time extraction and prediction of musical chord progressions for creative applications," *Electronics*, vol. 10, no. 21, p. 2634, 2021.

[7] N. Schnell, A. Röbel, D. Schwarz, G. Peeters, R. Borghesi, *et al.*, "Mubu and friends–assembling tools for content based real-time interactive audio processing in max/msp," in *ICMC*, 2009.

[8] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-time corpus-based concatenative synthesis with catart," in *9th International Conference on Digital Audio Effects (DAFx)*, pp. 279–282, 2006.