Machine Learning within Ableton Live

Giuseppe Torre[1]

Abstract. This paper presents and describes the features of a new M4L module for Ableton Live which enables basic gesture tracking and mapping. The module is based on the gesture-follower patch of the IRCAM's FTM library and it makes use of the native capabilities of the Live API. The mapping between the gestures and audible output is designed within the module and it enables the simultaneous control of macro features of the Ableton interface such as clip selection, volume and panning of the track/s.

1 INTRODUCTION

Machine Learning (ML) toolkits such as Wekinator [11], SARC Eyesweb Catalog[3], IRCAM MnM toolbox[4] and OpenCV [8] are between the most well-known toolkits available to artists and engineers [2]. Most of these tools have Open Sound Control (OSC) [9] capabilities which allow them to communicate with third-party software which create and manage the audio/visual elements of the performance. The process connecting gestures to audio-visual outputs is generally dealt with ad hoc algorithms to suits the performance or the performer's needs.

The module described in this paper, ML-AL, is based on the IRCAM's MnM toolbox and it exploits the M4L [6] capabilities to directly interface gesture recognition algorithms with the audio/visual elements within the software Ableton Live [5]. In that regard the module does not present a new algorithm or a novel approach in Machine Learning but rather implements existing technologies within one popular music software. However, the design of the module offers an approach to the use of gesture tracking algorithms for the discrete control of system's parameters rather than a continuous sound generating mechanism and/or continuos controllers.

2 ML-AL FEATURES

The ML-AL module has three sections each of which is dedicated to a specific task. These are visually separated by the blue vertical lines as depicted in Fig.1. The sections are:

1. Data-Input: it serves the purpose of retrieving data from a connected device via OSC.
2. Gesture Recognition: it performs gesture recognition analysis on a user-built gesture dictionary.
3. Mapping: it links the result of the performed gesture analysis to a series of Ableton Live native functions.

2.1 Data-Input

The data input section receives from any device that can transmit according to the OSC protocol. In this section the user can decide through which port number the communication needs to be established. The data needs to be formatted according to the following OSC address:

- $\backslash$*accxyz*: this is the data which will be read by the gesture recognition software for the creation and subsequent anaysis of the gestures. It needs to be a list of three numbers (e.g. the acceleration readings from a 3-axis accelerometer).
- $\backslash$*1*$\backslash$ *push2*: The sender device must have a push button which works as a gate. This push button serves the purpose of clearly marking beginning and end points of a gesture by letting the data through when depressed and stopping the data when released.

2.2 Gesture Recognition

The gesture recognition section is based on the gesture-follower example patch offered in the IRCAM's MnM toolkit. This toolkit was chosen because fully compatible with the M4L package available in Ableton Live. The patch proved to be a reliable and a ready-made software that could successfully implement gesture recognition. Furthermore, it is written in Max and this allows for easy manipulation and addition of user elements. It is important to notice that the patch is not making explicit use of the *time progression* features that it enables [1]. Rather, the patch is used as a gesture classifier.

The author acknowledges the limitations that such a system imposes. This choice was dictated by privileging artistic needs over technical possibilities. In particular, machine learning was thought, at least in this early version of the module, to be useful for the discrete control of macro elements of a live performance, such as the triggering of pre-recorded loops, rather then acting upon an eventual sound generating mechanisms or continuous controllers. This is a strategy that the authors has found productive in severals other works [10].

2.3 Mapping

The data used for the control of the Ableton interface is the *likeliest* number value retrieved by the gesture analysis routine performed by the gesture recognition section and made available for mapping each time the push button is released.

The ML-AL module can store up to eighty presets. These presets can store the start/stop mode of a clip, volume and panning of up to eight different audio or MIDI Ableton tracks. The *likeliest* number value is then used to recall one of these presets on-the-fly.

3 MODE OF USE

The presented module has been tested and interfaced with an iPad running a custom patch made with TouchOSC and complying with the specifications outlined in Section 2.1.

[1] University of Limerick, Ireland, email: giuseppe.torre@ul.ie

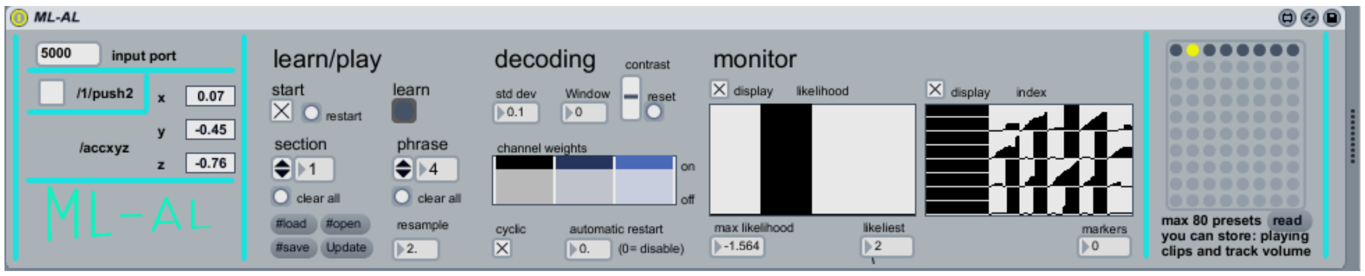

Figure 1. A snapshot of the ML-AL module for Ableton Live.

The mode of operation of the module consists of the following steps:

1. load the module onto a MIDI track.
2. input OSC port number over which to establish the connection (data-input section)
3. enable 'Learn' mode in the gesture recognition section
4. enable 'start' (toogle) and perform a gestures.
5. disable 'start' (toggle)
6. repeat steps 3 and 4 for as many gestures as you require. Make sure that to each gesture corresponds a different phrase number.
7. disable 'Learn' mode

Before using the module in performance mode, it is required to store some presets (preferably of equal quantity of the gestures performed during the training steps).

1. create up to eight audio or midi tracks in Ableton Live.
2. import or create as many clip as desired in each track
3. create a combination of playing clips, volume and panning settings and store these to a preset number (shift+click on one of the circles in the ML-AL module preset object).
4. repeat step 3

Now the module is ready to be used in performance mode. Enable start in the Gesture Recognition section and play a gesture. The gesture number performed, if successfully recognised, will recall the equivalent preset number.

3.1 Performance Scenario

The ML-AL module was preliminary tested by developing a short algorithmic process controlled by gestures performed using an iPad.. Four audio tracks were added to the Ableton project and each filled with two clips with distinct sound properties (high drones, low drones , granular, irregular percussive patterns). The following configuration was thought as 4-bit resolution system (four tracks by two clips and with a clip per track always playing) given therefore a total of sixteen possible combinations of playing clips stored as presets. The ML-AL module was trained with sixteen different gestures each of which mapped to a preset number. At performance time, the module response was fast and with a good rate of accuracy in recognising gestures. A drawback for the system is represented by the effort required by the user to learn sixteen different gestures. However, the quality of a live performance is not necessarily measured by the quantity of gestures performable or performed. The module capabilities, in conjunction with more traditional continuos controllers (faders) for the manipulation of audio effects, offers already a good sound palette for performance purposes.

4 CONCLUSION

This paper has presented and described the features of a new M4L module for Ableton Live which enables basic gesture recognition and mapping. The module, based on the IRCAM's MnM toolkit, offers simple and easy to use mapping of recognised gestures to basic control parameters available in Ableton Live such as clip selection, volume and panning settings for up to eight tracks. The module makes use of a highly efficient gesture recognition algorithm working on a non-continuos classification method. The mapping algorithm enables quick recall of Ableton presets so to, for example, control the macro elements of a performance such as the intervention of scenes and/or a selected combination of clips. The perceived latency during preliminary testing was found neglegible thus making the module suitable for live performances. ML-AL is freely available at [7].

REFERENCES

[1] F. Bevilacqua, B. Zamborlin, A. Sypniewski, N. Schnell, F. Guédy, and N. Rasamimanana, 'Continuous realtime gesture following and recognition', in *Gesture in Embodied Communication and Human-Computer Interaction*, eds., Stefan Kopp and Ipke Wachsmuth, volume 5934 of *Lecture Notes in Computer Science*, 73–84, Springer Berlin Heidelberg, (2010).

[2] B. Caramiaux and A. Tanaka, 'Machine Learning of Musical Gestures', *in Proceeding of New Interface for Musical Expression Conference, NIME'13*, (May 2013 KAIST, Daejeon, Korea,).

[3] Eyesweb Catalog. available at: http://www.somasa.qub.ac.uk/ngillian/sec.html [accessed 27th of January 2014].

[4] Ircam Ftm. available at: http://ftm.ircam.fr/ [accessed 27th of January 2014].

[5] Ableton Live. available at: https://www.ableton.com/ [accessed 27th of January 2014].

[6] Max for Live. available at: https://www.ableton.com/en/live/max-forlive/ [accessed 27th of January 2014].

[7] ML-AL. available at: http://muresearchlab.com/?/softwares/mlal/ [accessed 27th of January 2014].

[8] OpenCV. available at: http://opencv.org/ [accessed 27th of January 2014].

[9] OpenSoundControl. available at: http://opensoundcontrol.org/ [accessed 27th of January 2014].

[10] G. Torre, *The Design of a New Musical Glove: A Live Performance Approach*, (Ph.D. Thesis) University of Limerick, 2013.

[11] The Wekinator. available at: http://wekinator.cs.princeton.edu/ [accessed 27th of January 2014].