Supporting Interactive Machine Learning Approaches to Building Musical Instruments in the Browser

Louis McCallum
Goldsmiths, University of London
New Cross, London
l.mccallum@gold.ac.uk

Mick Grierson
Creative Computing Institute, UAL
London
m.grierson@arts.ac.uk

ABSTRACT

Interactive machine learning (IML) is an approach to building interactive systems, including DMIs, focusing on iterative end-user data provision and direct evaluation. This paper describes the implementation of a Javascript library, *Learner.js*, encapsulating many of the boilerplate needs of building IML systems for creative tasks with minimal code inclusion and low barrier to entry. Further, we present *MaxiInstruments*, a set of complimentary Audio Worklet-backed instruments to allow for in-browser creation of new musical systems able to run concurrently with various computationally expensive feature extractor and lightweight machine learning models without the interference often seen in interactive Web Audio applications.

Author Keywords

Interactive machine learning, browser-based tools

CCS Concepts

•**Applied computing** → **Sound and music computing;** Performing arts; •**Information systems** → *Music retrieval;*

1. INTRODUCTION

Interactive Machine Learning (IML) is a great approach for building interactive systems, including new music instruments and their mappings from input to sound [4]. Building browser based tools is useful because Javascript is a fast growing language with low barriers to entry. Additionally, browser based tools require little of the installation and dependencies that plague other data science endeavours. In the context of conducting limited time demos and workshops, fast proliferation and remixing of work, and uptake by non-technically savvy musicians, this is a massive advantage.

Building on an existing library for incorporating IML design into Javascript projects, we have sought to provide a scaffolding framework of additional libraries and example code which makes the production of web based musical instruments with mappings designed using IML as simple as possible for musicians. In doing so we address the non trivial challenges of connecting inputs from a variety of sources, running potentially computationally expensive feature extractors alongside lightweight machine learning models and generating audio output in real time.

NIME'20, July 21-25, 2020, Royal Birmingham Conservatoire, Birmingham City University, Birmingham, United Kingdom.

2. RELATED WORK

Using IML as a design tool for the development of mappings between inputs and outputs in interactive systems is supported by a number of softwares and libraries, including Wekinator [4], Teachable Machine [1], and Exemplar [6], with the first specifically oriented towards building musical and artistic systems. Such systems enable new controllers to be built quickly and by people without programming expertise by iteratively providing examples of behaviours and mappings, training and trying out models.

Browser based programming environments for teaching computer science concepts, including creative computing [9] are becoming increasingly prevalent and their advantages are clear [5]. As such, a number of libraries and frameworks have been developed to aid both music production and machine learning in the browser. For example, the Codecircle and MIMIC Project platforms have been used to teach a number of large scale MOOCs with a curriculum of interactive art and music [10] and the associated RAPID-MIX IML project ensured it cross-compiled a Javascript version of its library [2]. Further, the availability of much of Google Magenta's work as Javascript demos in Tensorflow.js can be seen as a key driver of interest from those outside of the data science community. Libraries adding extra functionality to the existing Web Audio API include Gibber[8], Tone.js [7] and maximilian.js [5].

3. THE TOOLKIT

With the aim of supporting musicians using machine learning to build new musical instruments in the browser, we present a toolkit consisting of minimal demonstration code for connecting inputs and extracting features, a library for easily incorporating the main functionality and graphical interfaces needed for including IML in a web project (*Learner.js*), with a simple synthesiser and sampler to provide musical output (*MaxiInstruments*). We develop with the constraint for all to run in realtime without audible interference. A typical user's workflow would be as follows

1. User picks an input (e.g. BodyPix skeleton tracking) and some instruments (e.g. 2 synthesisers) for a project.
2. User picks some parameters to be mapped (e.g. LFO rate, attack and pitch)
3. They record examples of inputs (e.g. poses), paired with examples of sounds they wish associate with the inputs
4. Train, then run the model, where new inputs (e.g. poses) make new sounds
5. Iterate

3.1 Machine Learning

A musical system using IML as a design tool will require the practitioner to use some form of sensor to get information from the world (e.g microphone, camera, IMU), and then extract some features from this signal. On the MIMIC Project online creative coding platform[1], we provide minimal example projects of how to incorporate a number of popular feature extractors with the library described below. These include musical audio extractors using the MMLL.js library [3], video feature extractors and sensors.

We have developed *Learner.js*, a lightweight Javascript library to encapsulate functionality for common tasks.

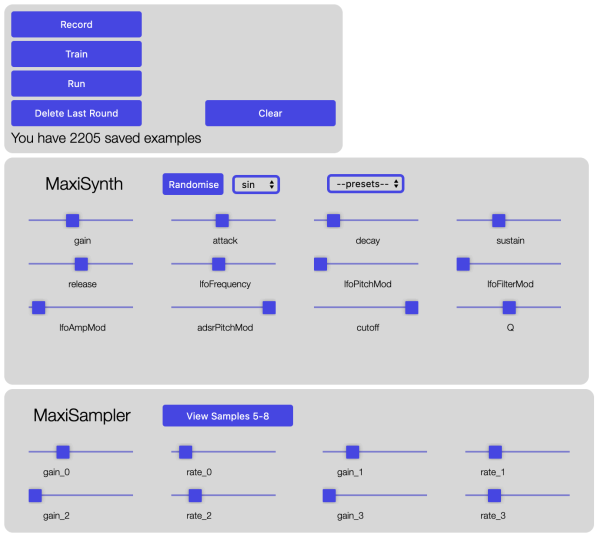
- *Recording and persisting a dataset of labelled examples*: A training set in supervised machine learning requires inputs paired with outputs. With one line of code, users can add in a hook that twins the inputs coming in with the current values of the mapped parameters of the instruments (described below). In recording mode, these are collected and persisted in the browser's IndexedDB so the data is not lost when the project is rerun. In running mode, the input is passed to the trained model to get new output values, which can be used to update the instruments parameters with the inclusion of another small API call.
- *Running machine learning models in a separate worker threads*: Users can choose either a classification or regression task. When new input is passed to a trained model, inference is executed on a separate worker thread, meaning any audio production or sequencer timing is not interrupted.
- *Adding in GUI for recording, training and running models*: With another one line code inclusion, users can add in a GUI for these common tasks (See Fig **??**. The methods are exposed so the library can also be run headlessly, or with the user's own GUI.

3.2 Audio

Sample level processing has been available in Web Audio since its inception, but this has been executed via the *ScriptProcessorNode* which runs on the main thread. This means it is easily interrupted by GUI interactions or other processes which limits its utility to run new musical instruments in real time. Audio Worklet's are a solution to this that allow for DSP and synthesis on a dedicated thread. We use the most recent version of maximilian.js and provide a polyphonic, additive synthesiser and a sampler running using Audio Worklets. Seen in Fig 1, each has a set of parameters that can be specified to be controlled using a given input, with a mapping designed by a user through iterative data provision, training and direct evaluation.

Running off a central clock on the Audio Worklet thread, each can be sequenced via MIDI interfaces, hand programmed scores, loaded MIDI files or directly with patterns generated by the Magenta library. Load testing has allowed for four separate polyphonic synthesisers and one sampler all running concurrently with the BodyPix multi-person body segmenter and a *Learner.js* regression model without audio or scheduling interference. Whilst making music in the browser has its limitations, the above configuration has a broad scope for creating expressive musical performances. Acknowledging that musicians may wish to generate audio outside of the browser whilst using *Learner.js* mappings, we also provide example code to send outputs via MIDI or OSC to external software.

[1]https://mimicproject.com



You have 2205 saved examples

Figure 1: GUIs for Synth, Sampler and IML

4. CONCLUSION

We have presented a toolkit, including two novel Javascript libraries, that allows for high quality, realtime music systems to be built rapidly in the browser using an interactive machine learning approach to mapping inputs to audio outputs. Encapsulating common tasks and addressing technical challenges involved with running realtime machine learning-powered musical systems in the browser, this frees musicians, students and educators to spend time focusing on creative challenges.

5. REFERENCES

[1] Teachable Machine. `https://teachablemachine.withgoogle.com/`.

[2] F. Bernardo, M. Grierson, and R. Fiebrink. User-Centred Design Actions for Lightweight Evaluation of an Interactive Machine Learning Toolkit. *Journal of Science and Technology of the Arts*, 10(2):2–25–38, July 2018.

[3] N. Collins and S. Knotts. A Javascript Musical Machine Listening Library. In *ICMC'19*, page 5.

[4] R. Fiebrink, P. R. Cook, and D. Trueman. Human model evaluation in interactive supervised learning. In *CHI'11*, pages 147–156, 2011.

[5] M. Grierson, M. Yee-King, L. McCallum, C. Kiefer, and M. Zbyszynski. Contemporary Machine Learning for Audio and Music Generation on the Web: Current Challenges and Potential Solutions. In *ICMC*, 2019.

[6] B. Hartmann, L. Abdulla, M. Mittal, and S. R. Klemmer. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *CHI'10*, pages 145–154, 2007.

[7] Y. Mann. Interactive Music with Tone.js. In *WAC'15*.

[8] C. Roberts and J. Kuchera-Morin. Gibber: Live Coding Audio in the Browser. In *ICMC'11*, page 6.

[9] M. J. Yee-King, M. Grierson, and M. D'Inverno. Evidencing the Value of Inquiry Based, Constructionist Learning for Student Coders. *International Journal of Engineering Pedagogy (iJEP)*, 7(3):109, Sept. 2017.

[10] M. Zbyszynski, M. Grierson, and M. Yee-King. Rapid Prototyping of New Instruments with CodeCircle. NIME, May 2017.