Write once run anywhere revisited: machine learning and audio tools in the browser with C++ and emscripten

Michael Zbyszyński
EAVI Group
Computing, Goldsmiths
London
m.zbyszynski@gold.ac.uk

Mick Grierson
EAVI Group
Computing, Goldsmiths
London

Leon Fedden
Computing, Goldsmiths
London

Matthew Yee-King
Computing, Goldsmiths
London

ABSTRACT

A methodology for deploying interactive machine learning and audio tools written in C++ across a wide variety of platforms, including web browsers, is described. The work flow involves development of the code base in C++, making use of all the facilities available to C++ programmers, then transpiling to asm.js bytecode, using Emscripten to allow use of the libraries in web browsers. Audio capabilities are provided via the C++ Maximilian library that is transpiled and connected to the Web Audio API, via the ScriptProcessorNode. Machine learning is provided via the RapidLib library which implements neural networks, k-NN and Dynamic Time Warping for regression and classification tasks. An online, browser-based IDE is the final part of the system, making the toolkit available for education and rapid prototyping purposes, without requiring software other than a web browser. Two example use cases are described: rapid prototyping of novel, electronic instruments and education. Finally, an evaluation of the performance of the libraries is presented, showing that they perform acceptably well in the web browser, compared to the native counterparts but there is room for improvement here. The system is being used by thousands of students in our on-campus and online courses.

1. INTRODUCTION

The last few years have seen a step change in the level of activity around machine learning, Most recently, there has been increased interest in the application of machine learning in the context of computational creativity, for example the generation of sound and graphics using neural networks. The techniques being developed have amazing potential to transform the creative process, especially if they can be placed into the hands of artists. The problem is that the techniques and software used in machine learning are currently, somewhat impenetrable to non-specialists. The software is complicated to install, the models are difficult to design and train, and there is therefore a steep learning curve to climb.

The aim of the work presented in this paper is to address this problem, by providing a lower barrier of entry to machine learning techniques (and their creative applications, especially sound), aligned with a rapid prototyping philosophy. The target audience includes creatives wishing to experiment with machine learning, and students, who we wish to expose to machine learning as early as possible in their studies.

The challenge here is to provide powerful, reliable libraries, with high quality code bases, which at the same time can be easily deployed and used by people with limited experience of machine learning and audiovisual techniques, and which provide real-time, interactive performance. In addition, we are keen to provide a pathway through to more advanced, performant use of the libraries such as one might expect from native applications written in C++.

In this paper, we describe an approach to solving this problem in the specific context of machine learning combined with audio, involving three key elements:

1. Machine learning and audio libraries implemented in C++
2. Transpiled, asm.js versions of the libraries, accessible in web browsers
3. A browser-based IDE which makes it possible to access, run and adapt examples and tutorials with minimal setup time

In the following sections, we describe the components of this system and how they interoperate, and we present an initial performance evaluation.

1.1 Related work

The RapidLib library described later draws extensively on the Wekinator interactive machine learning library, which was originally implemented in Java [4]. Interactive machine learning has long been used for creative purposes, for example the use of interactive genetic algorithms to design sound synthesis algorithms [5]. Wekinator makes this much easier and faster to achieve.

There are several machine learning libraries available in JavaScript, via the node package manager (NPM) ecosystem. An example is Specht's Thoughtful library[1], which provides Feed Forward Neural Networks, Naive Bayes Classification, K-Nearest Neighbours and K-Means Clustering. Such libraries are not typically designed for beginners, but they show it is possible to do this work in JavaScript.

The Web Audio API provides some audio analysis capabilities such as FFT, but more advanced feature extraction requires other libraries such as Meyda [1]. JavaScript libraries are less mature

[1] https://github.com/T-Specht/thoughtful

than other languages - Moffat et al. selected the Meyda library as the only JS library in a recent survey of toolboxes in a range of languages [3]. The library we describe combines sound synthesis with audio analysis, with an aim to provide this functionality with the minimal amount of starter code.

Emscripten is a tool that is used to convert C++ code into fast JavaScript code (asm.js), to run this code in the web browser, and to expose hooks into the code to normal JavaScript code [7]. Emscripten has been used before, to allow the development of sound synthesis algorithms in non-JavaScript environments, then their conversion to Web Audio API. For example, Faust [6] and PureData via Roth's Heavy compiler[2].

2. IMPLEMENTATION

In this section, we briefly describe how we have implemented the three key components of our rapid prototyping toolchain: the machine learning library, the audio library, and the Codecircle IDE. Figure 1 shows an overview of how the machine learning and audio libraries interact with each other and the rest of the web browser.

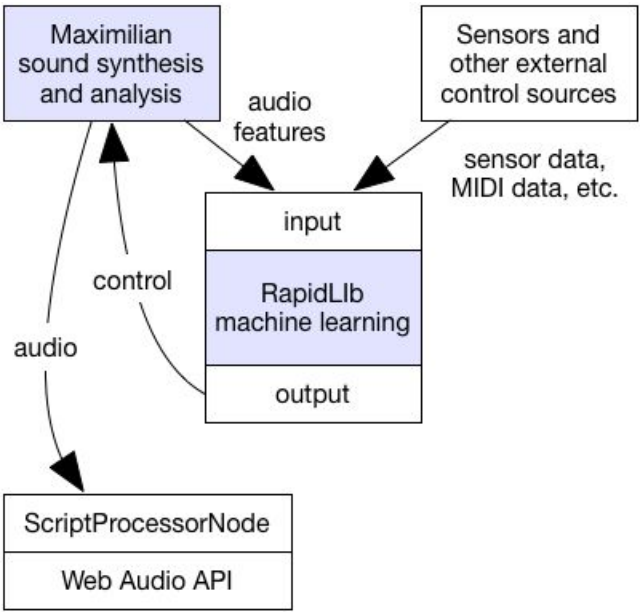

Figure 1. Overview of the library components of the system

2.1 RapidLib: a machine learning library

RapidLib is a machine learning library that was developed as part of the *Real-time Adaptive Prototyping for Industrial Design of Multimodal Interactive and eXpressive technologies* (RAPID-MIX) project, an Innovation Action funded by the European Commission under the Horizon 2020 program. It is a lightweight set of libraries, written in C++, that implements Interactive Machine Learning (IML) in the style of Fiebrink's [4] Wekinator[3]. Like Maximilian (below), this library was written for easy transition between C++ and JavaScript API's. The RapidLib C++ and JS API's share a JSON import/export format with a custom version of Wekinator, so users can move easily between them.

IML allows developers and end-users to quickly customize interactions by demonstration, encoding intuitive knowledge about performance gestures into trained machine learning models. IML is characterized by smaller data sets (relative to classical machine learning) and rapid iteration between training and testing.

At the time of writing, the core RapidLib library provides the following features:

- Feedforward, multilayer percetptron neural networks with backpropagation for regression
- K-Nearest Neighbours for classification
- Dynamic Time Warping for time series classification
- Basic signal processing for sensor conditioning
- Easy access to interactive machine learning

Using RapidLib in Codecircle, performers can create a set of gestures associated with desired outcomes and immediately (a few seconds) experiment with an interactive space based on those gestures. Undesirable results can be refined or discarded as the design process continues.

The following JavaScript code fragment shows how to create a regression model using RapidLib, train the model and query the model with a new input:

```
//Access the global instance of the library
var rapidMix= window.RapidLib();
//Create a regression (MLP) object
var myRegression= new rapidMix.Regression();
//Specify training data (normally we'd train
from sensor input)
var trainingSet= [
{ input: [0,0], output: [0]},
{ input: [0,1], output: [1]},];
// train it
myRegression.train(trainingSet)
// run the model with mouse position input
regressionOutput= myRegression.run([mouseX,
mouseY]);
```

2.2 Maximilian: an audio library.

Maximilian is a cross platform audio library written in C++ [Grierson, 2010]. It was originally developed with the aim of providing artists, with limited experience of audio programming a means to utilise high performance, advanced synthesis and analysis techniques in C++ projects. It was also developed for the purposes of teaching these techniques, to artists and others who were learning C++. The core library provides the following features:

- sample playback, recording and looping
- a selection of oscillators and filters
- enveloping
- multichannel mixing for 1, 2, 4 and 8 channel setups
- effects including delay, distortion, chorus, flanging
- granular synthesis, including time and pitch stretching
- realtime music information retrieval functions: spectrum analysis, spectral features, octave analysis, Bark scale analysis, and MFCCs

Maximilian has been adapted such that it can be transpiled to

[2] https://enzienaudio.com/

[3] http://www.wekinator.org/

asm.js, then accessed in the web browser. Importantly, the JavaScript code that users write looks almost the same as the C++ code, meaning users can transition to the native C++ mode when necessary.

The following JavaScript code extract illustrates the use of Maximilian to create a basic synthesis patch:

```
var maxiAudio = new maxiLib.maxiAudio();
var myWave = new maxiLib.maxiOsc();
maxiAudio.init();
maxiAudio.play = function() {
  this.output = myWave.sinewave(440);
};
```

2.3 Codecircle: a browser based IDE

Codecircle is a browser based, integrated development environment, that we created initially for use in our teaching of creative computing. It has been iteratively developed to have the features that we have deemed necessary for this task, as opposed to being developed in a top down way to offer a more typical, IDE feature set. Codecircle has been used by thousands of students, from our on campus and online, MOOC courses. Figure 2 shows a screenshot of the Codecircle user interface. The key features are as follows:

- Code is edited and executed in the browser
- Programmers can use JavaScript, HTML and CSS.
- It is very simple to fork and edit programs (e.g. example code provided during a programming lab).
- It allows live coding where code is re-interpreted as you type.
- It allows assets such as audio files to be added to projects
- It generates high resolution, timestamped code editing logs for the purposes of learning analytics.
- It uses jshint to highlight basic coding errors.
- Audiovisual and machine learning libraries are integrated, allowing scaffolded use of real-time graphics and sound.

More information about Codecircle, including a comparison to other browser programming environments, is provided in [8]. Details of how we are using Codecircle for teaching and research are provided in [9].

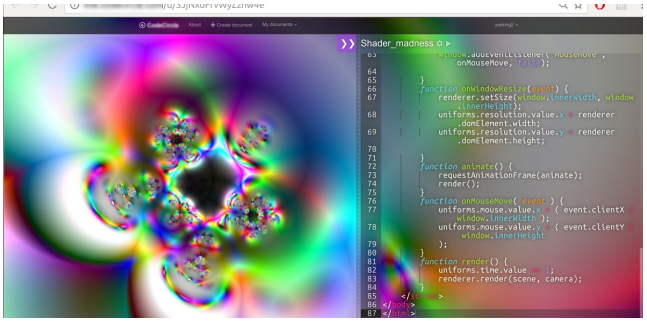

Figure 2. Screenshot of the codecircle platform, showing the program running on the left and the code being edited on the right

2.4 Integration with external hardware and sensors

This section is a brief survey of methods to bring sensor and media data into the browser environment. Characteristic of web development, there are many methods and implementations are sometimes inconsistent across browsers and platforms. The sections below describe methods we have been able to use. Some modes of interaction, such as stylus and multitouch, are not available at this time.

2.4.1 Mouse & keyboard

Keyboard and mouse gestures can be used to control complex musical material. IML can allow users to quickly map two-dimensional control spaces to high-dimensional control spaces [10]; these mappings can be generated and refined without any programming by the end-user.

2.4.2 Gamepad API

The Web Gamepad API[4] is still in the draft stage. The current API supports buttons (which can have an analog value) and axes (normalized from -1.0 to 1.0) for up to four controllers. There is also a draft for extensions to this API that includes haptic feedback and gamepad position[5].

The authors were able to use this API in Chrome and Firefox on OSX and Windows, using a Saitek Xbox-type controller, a GameTrak, and a Logitech Flight controller. We view this API as very promising, and are planning future prototypes to explore further.

2.4.3 Sensors on mobile devices

Mobile operating systems (both iOS and Android) provide specific API's for web apps directly to access on-board sensors, including accelerometers, gyroscopes, magnetometers, and GPS. These data can be forwarded to a central server, as in the CoSiMa project [11], or processed directly on the device.

2.4.4 MIDI

The Web MIDI API (https://webaudio.github.io/web-midi-api/) is also in draft form, and is currently only implemented in Chrome and Opera[6]. While this might be a serious limitation for a commercial product, it does not seem unreasonable to ask developers of experimental instruments to download a common browser. In December 2016, Chrome had more than 50% share of desktop browsing[7], and even more users have Chrome on their computer.

2.4.5 WebSockets

The most secure and rich method for two-way data exchange between browser and server is currently via the WebSocket protocol. For communication between sensors and browsers running on the same computer, WebSockets require that a server be running locally. Although this is somewhat inconvenient, it is not an insurmountable obstacle.

Many manufacturers provide WebSocket software that facilitates communication with their devices. For example, Myo's daemon[8]

[4] https://www.w3.org/TR/gamepad/

[5] https://w3c.github.io/gamepad/extensions.html

[6] http://caniuse.com/#feat=midi

[7] http://gs.statcounter.com/#all-browser-ww-monthly-201612-201612-bar

[8] https://github.com/logotype/myodaemon

and BITalino's python server[9].

WebSockets can also be a general transport for Open Sound Control. [12] We have developed a stand-alone NodeJS server[10] that passes OSC packets to and from the browser, and a special build of Codecircle that understands OSC. Users of OSC-enabled software can send OSC to the server over UDP.

2.4.6 WebRTC

WebRTC is a protocol for real-time communication between browsers, and allows for peer-to-peer exchange of audio, video, and control data. Incoming audio buffers can be passed to MaxiLib as arrays of 32-bit floats for processing and/or feature extraction, while video frames can be passed to RapidLib via a canvas.

3. EXAMPLE USE CASES

3.1 Rapid NIME prototyping

When designing new interfaces for musical expression (NIMEs), it is usually desirable to refine a design by iterating over a number of prototypes. As a design iterates, it should become more aligned with the physical affordances of the performer as well as the aesthetic needs of the performance. Web browsers have been identified as appealing hosts for new expressive interfaces. Roberts, Wakefield and Wright [13] demonstrated the potential for browsers, and specifically the Web Audio API to allow both synthesizers and interfaces to be programmed in JavaScript. Wyse and Subramanian [14] examine computer music in browsers more generally, noting the potential offered by browsers' intrinsic connection to networks, ease of access, and portability. New standards from W3C and extensions to JavaScript have made browser technology more "viable" for musical work. Our tool chain builds on this, adding easy access to machine learning, and more sound synthesis and analysis capabilities. Also we can easily provide working examples for different types of input sensors and controllers. The Codecircle platform adds more here, providing real-time coding, and easy sharing of programs.

3.2 Education

The tool chain can be used for teaching in a variety of ways. During a lab class, the tutor might provide example code in the Codecircle platform, which students can quickly evaluate in class, download, fork, customise, etc. Students might share code with each other online, for feedback or even peer assessment, knowing that it will run as soon as their peer hits the URI. The key features that make it very useful in the educational use case are the instant, no setup access to sound synthesis and machine learning, real-time coding capabilities, and the easy sharing and forking of code.

4. EVALUATION

We have conducted a preliminary performance evaluation, wherein we compared the performance of Maximilian and RapidLib running on the same machine in native, compiled mode and in transpiled mode in the Chrome browser. For Maximilian, we calculated how many times we would be able to call the various sound synthesis functions before the audio thread would run late. This was achieved by timing thousands of calls to the various functions and averaging the time required, then combining this with the sample rate. We verified these theoretical calculations about the performance, by attempting to run that many oscillators, filters etc. in real time and listening for audio dropouts. For RapidLib, we simply measured the time taken to train a neural net on a single example, i.e. feeding the input, output pair in and back propagating the error, and then the time taken to compute the output based on an unseen input.

Table 1. Performance comparison of native libraries and their transpiled JavaScript counterparts

	C++ Maximilian/ RapidLib Linux	Transpiled Maximilian/ RapidLib in Chrome, Linux
maxiOsc.sinewave(): how many calls?	315	29
maxiOsc.sinebuf(): how many calls?	617	26
maxiFFT.process(): how many calls?	5	2
maxiSVF.play() : how many calls?	668	18
NN train ms per example	8.762	24.445
NN test ms per example	0.015	0.058

The results of the tests are shown in Table 1. We found that the performance of the audio library in the browser was an order of magnitude slower than its native counterpart, and the machine learning library was approximately 3 times slower on training and 4 times slower on testing. Further testing revealed that we should be able to improve the performance of the audio library in the browser significantly with some optimisation, as a basic ScriptProcessorNode calling Math.sin seemed to run a lot faster than one calling maxiOsc.sinwave via the transpiled library. This work is ongoing. and the code for the tests is available as Github gists [11].

5. CONCLUSION

We have described a novel tool chain which can be used to rapidly prototype interactive machine learning and audio programs, in the web browser. The technical approach is to

[9] https://github.com/BITalinoWorld/python-serverbit

[10] http://gitlab.doc.gold.ac.uk/rapid-mix/rapid_web_service

[11] https://gist.github.com/fedden/33abe0700b9df4efcce96828061b4a98
https://gist.github.com/fedden/01fd102662b382982a2c7bb29d74f11d
https://gist.github.com/fedden/280f7199818b6cd91f123194d5f7184b
https://gist.github.com/fedden/54ff0f8e5122368eaf7801abaff9b9f9

develop machine learning and audio libraries in C++, then to use Emscripten to transpile them into asm.js code that can be understood by web browsers. The RapidLib library provides interactive machine learning functionality, such as neural networks, k-NN, and DTW based on a novel C++ implementation of a subset of the Wekinator functionality. The Maximilian library provides extensive, real-time sound synthesis and analysis capabilities, and has been designed to be accessible to artists, and useful for teaching purposes. It is transpiled to asm.js then fed to the Web Audio API using a ScriptProcessorNode. The Codecircle platform is a browser based IDE, developed for educational purposes, that makes it easy to rapidly prototype digital musical instruments with machine learning functionality. The libraries perform acceptably well in the browser, but are significantly slower than their native counterparts. There is plenty of scope for adding more functionality to the libraries and improving their performance in the future.

6. ACKNOWLEDGMENTS

This work was partially funded under the HEFCE Catalyst Programme, project code PK31.

The research leading to these results has also received funding from the European Research Council under the European Union's Horizon2020 programme, H2020-ICT-2014-1 Project ID 644862

7. REFERENCES

[1] Rawlinson, Hugh, Nevo Segal, and Jakub Fiala. 2015 "Meyda: an audio feature extraction library for the web audio api." The 1st Web Audio Conference (WAC). Paris, Fr. 2015.

[2] Grierson, M. 2010. “Maximilian: A cross platform c++ audio synthesis library for artists learning to program”. In Proceedings of the International Computer Music Conference (Jan. 2010), New York.

[3] Moffat, David, David Ronan, and Joshua D. Reiss. 2015, "An evaluation of audio feature extraction toolboxes." Proceedings of the 18th International Conference on Digital Audio Effects (DAFx-15), Trondheim, Norway.

[4] Fiebrink, Rebecca, and Perry R. Cook, 2010. "The Wekinator: a system for real-time, interactive machine learning in music." Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010), Utrecht.

[5] Woolf, Sam, and Matthew Yee-King, 2002, "Virtual and physical interfaces for collaborative evolution of sound." Contemporary Music Review 22.3 (2003): 31-41.

[6] Borins, Myles. 2014, "From faust to web audio: Compiling faust to javascript using emscripten." Linux Audio Conference, Karlsruhe, Germany. 2014.

[7] Zakai, Alon. 2011, "Emscripten: an LLVM-to-JavaScript compiler." Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion. ACM, 2011.

[8] Fiala, Jakub, Matthew Yee-King, and Mick Grierson. 2016, "Collaborative coding interfaces on the Web." Proceedings of the 2016 International Conference on Live Interfaces.

[9] Yee-King, Matthew, Mick Grierson, and Mark d’Inverno. 2017 "STEAM WORKS: Student coders experiment more and experimenters gain higher grades." IEEE Engineering Education Conference, Athens Greece, 2017

[10] Momeni, Ali, and David Wessel. 2003 "Characterizing and controlling musical material intuitively with geometric models." Proceedings of the 2003 conference on New interfaces for musical expression. National University of Singapore, 2003.

[11] Schnell, Norbert, et al. "Collective Sound Checks: Exploring Intertwined Sonic and Social Affordances of Mobile Web Applications." Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction. ACM, 2015.

[12] Wright, Matthew. 2005 "Open Sound Control: an enabling technology for musical networking." Organised Sound 10.03 (2005): 193-200.

[13] Roberts, Charles, Graham Wakefield, and Matthew Wright. 2013, "The Web Browser As Synthesizer And Interface." NIME. 2013.

[14] Wyse, Lonce, and Srikumar Subramanian. "The viability of the web browser as a computer music platform." Computer Music Journal 37.4 (2013): 10-23.

Web Audio Conference WAC-2017, August 21–23, 2017, London, UK.