


Differentiable Logic for Interactive Systems and Generative Music - Ian Clester


BeagleBoard.org
Apr 17, 2024

Table of contents

Chapter 1

Introduction

1.1 Summary links

- **Contributor:** Ian Clester
- **Mentors:** Jack Armitage, Chris Kiefer
- **GSoC:** Google Summer of Code

1.2 Status

This project is currently just a proposal.

1.3 About

- **Forum:** u/ijc (Ian Clester)
- **OpenBeagle:** ijc (Ian Clester)
- **Discord:** ijc (Ian Clester)
- **Github:** ijc8 (Ian Clester)
- **School:** Georgia Institute of Technology
- **Country:** United States
- **Primary language:** English
- **Typical work hours:** 9AM-6PM US Eastern
- **Previous GSoC participation:** Better Faust on the Web (2023)

Chapter 2

Project

Project name: Differentiable Logic for Interactive Systems and Generative Music

2.1 Description

The general aim of this project is to enable the development of models that are suitably efficient for use in real-time interactive applications on embedded systems (particularly the BeagleBone-based Bela). At the project's core is difflogic[1], a recent technique that employs sparsely-connected network composed of basic logic gates (rather than densley-connected neurons with complex activation functions) to obtain small models and fast inference. Thus, the first and foremost goal of the project is to enable a convenient workflow for developing difflogic models and running them on the Bela. The expected use case is developing and training models on a larger machine (e.g. a laptop, desktop, or server), followed by exporting the model to C and cross-compiling it for the BeagleBone - either the main CPU (ARM Cortex-A8) or the PRUs. To support this workflow, I will develop wrappers for exporting compiled difflogic models for use in the various languages supported on Bela (C++, Pure Data, SuperCollider, Csound). These wrappers will likely take inspiration from other projects that bring machine learning into computer music environments, such as nn~ and FluCoMa. This first goal, along with profiling and benchmarking the performance of difflogic models on both the main CPU and the PRUs, constitutes roughly the first half of the project.

The other, more exploratory half of the project consists of building out integrations and applications of difflogic for the rapid development of useful audio models. To that end, I intend to explore the possibilities of combining difflogic networks with techniques such as DDSP (differentiable digital signal processing)[2], possibly also leveraging Faust auto-differentation. I also intend to investigate the feasibility of "porting" well-known ML architectures such as VAEs to difflogic networks, and of training difflogic networks to approximate the behavior of existing neural networks (i.e. knowledge distillation). Audio models such as RAVE[3], PESTO[4], and Whisper[5] may be of particular interest. Furthermore, I will explore opportunities to combine difflogic networks with other cheap, effective techniques like the $Q recognizer[6] for gestural control, linear predictive coding for audio analysis & resynthesis, and toolkits such as RapidLib. Such combinations may be particularly useful for interactive machine learning (as in Wekinator[7]), should fine-tuning difflogic models on-device prove too costly. In this

[1] Petersen, F. et al. 2022. Deep Differentiable Logic Gate Networks. Proceedings of the 36th Conference on Neural Information Processing Systems (Oct. 2022).

[2] Engel, J. et al. 2020. DDSP: Differentiable Digital Signal Processing. Proceedings of the International Conference on Learning Representations (2020).

[3] Caillon, A. and Esling, P. 2021. RAVE: A variational autoencoder for fast and high-quality neural audio synthesis. arXiv.

[4] Riou, A. et al. 2023. PESTO: Pitch Estimation with Self-supervised Transposition-equivariant Objective. Proceedings of the 24th International Society for Music Information Retrieval Conference (Sep. 2023).

[5] Radford, A. et al. 2023. Robust Speech Recognition via Large-Scale Weak Supervision. Proceedings of the 40th International Conference on Machine Learning (2023).

[6] Vatavu, R.-D. et al. 2018. $Q: a super-quick, articulation-invariant stroke-gesture recognizer for low-resource devices. Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services (New York, NY, USA, Sep. 2018), 1–12.

[7] Fiebrink, R. et al. 2009. A Meta-Instrument for Interactive, On-the-fly Machine Learning. Proceedings of the International Conference on New Interfaces for Musical Expression (2009), 280–285.

phase of the project, I will develop example applications involving sound analysis, classification, and synthesis, and experiment with interactive machine learning.

Finally, I intend to dedicate some time to a specific creative application: generating networks of logic gates to approximate particular sounds and exploring the space of such sound-generating networks. This application is inspired by bytebeat[8], a practice which involves writing short expressions that describe audio as a function of time, generating music sample-by-sample. Typically, these expressions involve many bit-twiddling operations, consisting primarily of logic gates (bitwise AND, OR, XOR, NOT) and shifts — a fact that suggests a remarkably good fit for difflogic, wherein models consist of networks of gates. Other inspirations include work on sound matching: reproducing a given sound or family of sounds by estimating synthesizer parameters[9], generating patches[10], or training models[11]. In this vein, I will attempt to train difflogic gates to reproduce particular sounds, treating the entire network as a bytebeat-style function of time (sample index) that outputs samples. Thanks to the tricks difflogic employs to train a network of discrete gates, this approach will enable sound matching via gradient descent and backpropagation (as in e.g. DDSP) rather than evolutionary methods, while still ultimately generating a discrete function. Lastly, I will build an interactive application to explore the space of sound-generating networks (e.g. by mutating a network, or morphing between two networks) and visualize the execution of logic gate networks.

2.2 Software

- C
- C++
- Python
  - PyTorch
  - difflogic
  - dasp
- Faust
- Linux

2.3 Hardware

- Bela
  - BeagleBone Black
  - Bela Cape
- Microphone
- Speaker
- OLED screen

[8] Heikkilä, V.-M. 2011. Discovering novel computer music techniques by exploring the space of short computer programs. arXiv.

[9] Yee-King, M. and Roth, M. 2008. Synthbot: An unsupervised software synthesizer programmer. ICMC (2008).

[10] Macret, M. and Pasquier, P. 2014. Automatic design of sound synthesizers as pure data patches using coevolutionary mixed-typed cartesian genetic programming. Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (New York, NY, USA, Jul. 2014), 309–316.

[11] Caspe, F. et al. 2022. DDX7: Differentiable FM Synthesis of Musical Instrument Sounds. Proceedings of the 23rd International Society for Music Information Retrieval Conference. (2022).

Chapter 3

Timeline

Note: This timeline is based on the official GSoC timeline

3.1 Timeline summary

Date	Activity
February 26	Connect with possible mentors and request review on first draft
March 4	Complete prerequisites, verify value to community and request review on second draft
March 11	Finalized timeline and request review on final draft
March 21	Submit application
May 1	Start bonding
May 27	Start coding and introductory video
June 3	Release introductory video and complete milestone #1
June 10	Complete milestone #2
June 17	Complete milestone #3
June 24	Complete milestone #4
July 1	Complete milestone #5
July 8	Submit midterm evaluations
July 15	Complete milestone #6
July 22	Complete milestone #7
July 29	Complete milestone #8
August 5	Complete milestone #9
August 12	Complete milestone #10
August 19	Submit final project video, submit final work to GSoC site and complete final mentor evaluation

3.2 Timeline detailed

3.2.1 Community Bonding Period (May 1st - May 26th)

GSoC contributors get to know mentors, read documentation, get up to speed to begin working on their projects

3.2.2 Coding begins (May 27th)

3.2.3 Milestone #1, Introductory YouTube video (June 3rd)

- Setup development environment
- Train trivial difflogic network on laptop & run generated C on Bela (main CPU)

3.2.4 Milestone #2 (June 10th)

- Run difflogic network on PRU
- Perform feature extraction (FFT, MFCCs) on PRU

3.2.5 Milestone #3 (June 17th)

- Build wrappers to simplify use of difflogic networks in Bela projects
  - C++ (namespace & wrapper around difflogic-generated C)
  - SuperCollider (UGen)

3.2.6 Milestone #4 (June 24th)

- Build wrappers to simplify use of difflogic networks in Bela projects
  - Pure Data (external)
  - Csound (UDO)

3.2.7 Milestone #5 (July 1st)

- Explore feasibility of combining difflogic with DDSP techniques (via dasp and possibly Faust auto-differentiation)
- Use difflogic network to control synthesizer parameters

3.2.8 Submit midterm evaluations (July 8th)

Important: July 12 - 18:00 UTC: Midterm evaluation deadline (standard coding period)

3.2.9 Milestone #6 (July 15th)

- Investigate feasibility of interactive machine learning (e.g. fine-tuning) with difflogic networks
- Combine difflogic network with complementary cheaply techniques (e.g. LPC, template matching via $Q, RapidLib)

3.2.10 Milestone #7 (July 22nd)

- Work on example applications
  - Classify short mouth sounds for interactive system control (à la parrot.py)
  - Perform real-time pitch estimation (à la PESTO)

3.2.11 Milestone #8 (July 29th)

- Experiment with implementing popular architectures (e.g. VAEs, as in RAVE) as difflogic networks
- Experiment with difflogic knowledge distillation: training a difflogic network to approximate the behavior of a pre-trained, conventional neural network (student/teacher)

3.2.12 Milestone #9 (Aug 5th)

- Experiment with training difflogic networks for sound reconstruction
  - Bytebeat-inspired: feed increasing timestamps to network, get subsequent audio samples out

3.2.13 Milestone #10 (Aug 12th)

- Creative application: Interactive exploration of space of difflogic sound reconstruction models
  - “Glitch” - random perturbations of network (mutate gates & connections)
  - “Morph” - interpolate (in terms of tree edit-distance) between different sound-generating networks
  - Visualize difflogic networks & their execution

3.2.14 Final YouTube video (Aug 19th)

Submit final project video, submit final work to GSoC site and complete final mentor evaluation

3.2.15 Final Submission (Aug 24nd)

Important: **August 19 - 26 - 18:00 UTC:** Final week: GSoC contributors submit their final work product and their final mentor evaluation (standard coding period)

August 26 - September 2 - 18:00 UTC: Mentors submit final GSoC contributor evaluations (standard coding period)

3.2.16 Initial results (September 3)

Important: **September 3 - November 4:** GSoC contributors with extended timelines continue coding

November 4 - 18:00 UTC: Final date for all GSoC contributors to submit their final work product and final evaluation

November 11 - 18:00 UTC: Final date for mentors to submit evaluations for GSoC contributor projects with extended deadline

---

Chapter 4

Experience and approach

I have extensive experience with embedded systems and real-time audio. As an undergraduate, I worked on embedded systems during internships at Astranis and Google. For a final class project, I developed a multi-effects pedal with a configurable signal chain in C using fixed-point arithmetic on the Cypress PSoC 5 (an ARM-based system-on-a-chip with configurable digital and analog blocks). My master's work involved localizing RFID tags using software-defined radios with framerates sufficient for interactive systems. Currently, I am a teaching assistant for a class on Audio Software Engineering (in Rust, with a focus on real-time audio software), in which I have been responsible for preparing much of the material and lectures. I have worked with a variety of microcontrollers and single-board computers, from writing assembly on the Intel 8051, to C++ on Arduinos and ESP32s, to Python and JS on Raspberry Pis.

I have also employed machine learning techniques to build interactive systems. In a graduate course on multimodal user interaction, I gained experience with classic machine learning techniques, and employed cheap techniques for gesture recognition in a tablet-based musical sketchpad. In the meantime, I have been following developments in machine learning for audio (particularly those that are feasible to run locally, especially sans GPU), and I have experimented with models such as RAVE and Whisper (using the latter for an recent interactive audiovisual hackathon project).

Much of my graduate work has focused on generative music and computational representations of music. My recent work on ScoreCard has put an extreme emphasis on fitting music-generating programs (typically written in C) into efficient, self-contained packages that are small enough to store in a QR code (< 3kB). Previous projects such as Blocks (an audiovisual installation) and kilobeat (a collaborative livecoding tool) have probed the musical potential of extremely short fragments of code (bytebeat & floatbeat expressions). These projects also explore methods of visualizing musical programs, either in terms of their output or their execution. More information about my work is available on my website and GitHub.

I am particularly interested in difflogic because it occupies an intersection between lightweight machine learning techniques (cheaper is better!) and compact representations of musical models (less is more!), and I am strongly motivated to see what it can do.

4.1 Contingency

If I get stuck on something related to BeagleBoard or Bela development, I plan to take advantage of resources within those communities (such as documentation, forums, and Discord servers).

If I get stuck on something related to ML or DSP, I plan to refer back to reference texts and the papers and code of related work (DDSP, RAVE, PESTO, etc.), and I may reach out to colleagues within the ML space (such as those in the Music Information Retrieval lab within my department) for advice.

If I get stuck on something related to music or design, I plan to take a break and go on a walk. :-)

4.2 Benefit

The first half of this project will provide a straightforward means to develop models with difflogic and run them on embedded systems such as BeagleBoards and particularly Bela. (The wrappers for Bela's supported languages may also prove generally useful outside of embedded contexts.) Making it easier for practitioners to use difflogic models in creative applications will, in turn, aid in the development of NIMEs and DMIs that can benefit from the small size and fast inference (and corresponding portability and low latency) of difflogic networks.

The second half of this project, depending on the results of my explorations, may demonstrate useful ways to combine difflogic with other ML & DSP techniques, and provide some useful and interesting audio-focused applications to serve as effective demonstrations of the possibilities for ML on the BeagleBoard and possible starting points for others.

4.3 Misc

Here is my pull request demonstrating cross-compilation and version control.

4.4 References