UNIVERSITY OF OXFORD

GROUP DESIGN PRACTICAL

GROUP 14

Machine Learning in the Browser with the BBC Micro:Bit

Louis-Emile Ploix[1], Ike Glassbrook[2], Joseph Simkin[3], Alikhan Murat[4], Andy van Horssen[5], and Ewan Hawkrigg[6]

Internal supervisor: Qian Xie

External supervisor: Robert Knight

May 2024

[1]louis-emile.ploix@stcatz.ox.ac.uk
[2]isaac.glassbrook@lmh.ox.ac.uk
[3]joseph.simkin@some.ox.ac.uk
[4]alikhan.murat@magd.ox.ac.uk
[5]andy.vanhorssen@sjc.ox.ac.uk
[6]ewan.hawkrigg@keble.ox.ac.uk

Contents

1 Introduction

The Group Design Practical is a course taken by all 2nd year undergraduate students at the University of Oxford studying for a degree in Computer Science, Mathematics and Computer Science, or Computer Science and Philosophy. This report details the work of Group 14 from February to May 2024 to design, implement, and deploy a product satisfying the specification as provided by Micro:Bit.

The project itself may be viewed on the page `https://ox24-cctd-ml-machine.pages.dev`, and the source code is publicly accessible at `https://github.com/Ike-G/ox24-cctd-ml-machine`.

1.1 Technical context

Our project is based on a machine learning tool developed by researchers at the Centre for Computational Thinking and Design at Aarhus University. This tool is publicly deployed at `https://ml-machine.org`, with its source code available at `https://github.com/microbit-foundation/cctd-ml-machine`. Our project is a fork of this repository, extending the existing project.

ML-Machine is an application for users with access to a physical micro:bit device to train a machine learning model that uses the micro:bit's sensors as input data. The user is given a 3 step process:

1. Named 'Gestures' are added, for the trained model to eventually use as classification categories. The user then adds to each gesture several recordings of the accelerometer that correspond to the particular gesture. Alternatively, they can use the example dataset, containing the categories 'shake' (with recordings of the micro:bit being shook), 'still' (with recordings of the micro:bit not moving), and 'circle' (with recordings of the micro:bit being moved in a circle).

2. The user trains a model – either a dense layers model, or a k-nearest neighbours model – to classify the data. The recordings are pre-processed using a set of filters, which build the feature set. Before training, the user may select in particular which filters they would like to use.

3. The micro:bit then has its sensors periodically polled in order to predict, of the labelled gestures, the one most similar to its current movement.

The web application is written using the Svelte framework[7], acting essentially as an HTML template language. The micro:bit itself requires additional drivers to interface with the application, as first a bluetooth connection must be established, and then the appropriate sensor data must be streamed to the application. These are written in C++, and then converted into `.hex` files, which users can install on their micro:bits in order to allow this procedure to work. Both models are trained in the browser using TensorFlow.js[8].

1.2 Project Specification

In consideration of the existing context, the specifications set out by Micro:Bit were designed with a view to utilise our work for experimental purposes, and to provide a benchmark for evaluating the feasibility of future projects. On this basis, the specification gave the following requirements of a final product:

- That a user should be able to train any model on not just the Micro:bit's accelerometer, but also an additional sensor.
  - The user should be able to choose which sensor's data is to be streamed into the training data (and thus, which sensor's data is polled once the model is trained).
  - Information given by the sensor should be visualised to the user in real-time in a manner which is understandable.
  - The sensor data should be amenable to machine-learning analysis via the models available.
- In addition to the Dense Neural Network, and the k-Nearest Neighbours models of the base application, a new neural network architecture should be investigated for implementation.
  - This network should be capable of predicting on simple patterns with reasonable accuracy.
  - The technical details of the network should be of pedagogical value, in addition to being amenable to high-level explanation.
  - The trained model should be of a size that could fit on the micro:bit itself, rather than needing to run on a connected device[9].
  - Model training should be responsive on standard browsers ran on computers with low-end modern hardware.

2 Logistics

This section describes the timeline of our development, and how we went about structuring the delegation of work.

2.1 Timeline

- 24th January – First briefing meeting, team members were introduced to one another, and communication channels were created.
- 26th January – First internal meeting, we decided on our three preferred projects, one of which was the project we were ultimately assigned.
- 20th February – First meeting with both supervisors. The project and existing code were explained, with directions given to relevant tools. It was decided that two weeks were to be given to all group members to analyse and understand the existing code, and decide on their areas of focus.

[7]`https://svelte.dev`
[8]`https://www.tensorflow.org/js`
[9]The Micro:Bit team have expressed a desire to run the prediction models directly on the micro:bit in future, rather than in the browser, allowing for the model to be stored independently of the application. This was not itself within the scope of our specification however.

- 3rd March – Project repository fork was created at `https://github.com/Ike-G/ox24-cctd-ml-machine`.
- 5th March – Initial internal meeting to organise plans for initial development over the Hilary vacation. Plans were set to have weekly internal meetings, and weekly external meetings, on Mondays and Tuesdays respectively. An initial role distribution was decided.
- 9th March – Beginning of Hilary vacation.
- 12th March – Updated drivers to expose the magnetometer service. Consideration of initial ideas for new ML models.
- 18th March – Introduced functionality to work with multiple sensor types, including addition of sensor visualisations.
- 24th March – Improvements to magnetometer preprocessing.
- 25th March – Implementation of ML training with multiple sensors.
- 31st March – Implementation of filter UI with multiple sensors.
- 3rd April – Deployed project as web application via Cloudflare.
- 7th April – Added fully functional Mixture of Experts model.
- 9th April – Work on implementing audio as a new sensor put to the side, focus placed instead on researching future solutions.
- 13th April – Implemented sensor selection in the UI.
- 15th April – Start of 0th week: added the light sensor, using previous work to implement sensor selection, visualisation, filtering, training, and predictions.
- 16th April – Gave initial demonstration to developers at Aarhus University.
- 28th April – Finalising quality of life changes. Focus placed on the final report.

2.2 Role Delegation

Roles were initially distinguished to place focus separately on:

- Implementing drivers.
  - Louis – Magnetometer (done directly through the Magnetometer service).
  - Alikhan – Microphone (done through `UART`).
- Joseph – Implementing a new ML model.
- Ike – Adding visualisation graphs of new sensors.
- Andy / Ewan – Testing of existing code / improvements to sensor pre-processing.

As the project progressed, it was quickly identified that significant alterations to existing code were necessary to implement multiple sensors. Ike dealt with this primarily once basic visualisation had been dealt with. Additionally, Louis was able to implement the magnetometer driver very on in the process, and consequently quickly moved to working on pre-processing and testing. Later down the line the multisensor implementation required more extensive UI, which Ike and Louis dealt with together.

Due to the size of the job, after deciding with the rest of the team on a Mixture of Experts model as the best option, Joseph spent most of the project working on developing and upgrading the model. As part of this, he worked in conjunction with those developing new sensors as the processing methods were changed, and new sensors were implemented.

Ultimately, the group eventually decided not to pursue training on the microphone sensor, due to audio data presenting far more complex problems than had previously been dealt with in the other sensors. Alikhan thus refocused his work from that point on researching means of circumventing our problems for a future implementation by the original project's developers. Instead of audio data, Louis implemented a light sensor later on in the project, meaning the project concluded with three available sensor types.

3 Implementation

3.1 Streaming of new sensors from the micro:bit

3.2 Sensor choice in the UI

3.3 Visualisation of new sensors

3.4 Application deployment

3.5 Normalisation of data

3.6 Mixture of Experts Model

4 Areas of Further Development

5 Concluding Remarks