Hand Gesture Recognition on Arduino Using Machine Learning

Matthew Lipski
Supervisors: Mingkun Yang , Ran Zhu
EEMCS, Delft University of Technology, The Netherlands
m.s.lipski@student.tudelft.nl, m.yang-3@tudelft.nl, r.zhu-1@tudelft.nl

1 Introduction

1.1 Motivation

Traditionally, physical buttons have been by far the most common way for users to interact with electronic devices in public settings, whether these are coffee machines, elevator panels, or train ticket machines. However, the concern of disease transmission has become increasingly prevalent in recent years due to the COVID-19 pandemic, making it enticing to develop an alternative solution which does not require touch. One such solution is the use of hand gestures to interact with public devices instead. By performing hand motions such as swiping and tapping, users can effectively and intuitively interact with electronic devices with minimum risk of disease transmission.

However, there are a few key challenges to this approach which stand in the way of it replacing physical buttons is real-world applications.

1. Additional hardware is needed to detect hand gestures, as a microcontroller on its own has no way of knowing the positions of a user's hand while performing a gesture.
2. Additional software must be implemented to recognize gestures. While buttons are just simple digital inputs, the same gesture can be performed somewhat differently across different users, yet the system must be able to accurately classify it regardless.
3. Gesture recognition must be done in real-time. Since the process of recognizing which gesture is being performed is quite complex, the latency introduced by this could be significant. However, the user should not feel any perceptible lag while using the system for a positive experience.

This research attempts to overcome these issues to create a gesture recognition system that runs on an Arduino Nano 33 BLE microcontroller. In this system, OPT101 photodiodes are used to detect the position of the user's hand at various points in time, while a machine learning model was trained to recognize which gesture is being performed. A more detailed description of the system pipeline can be found in section X. Similar research which involves using photodiodes and machine learning to recognize hand gestures has already been conducted, but this project improves on existing solutions in a number of ways:

1. The microcontroller used has less processing power (CITATION NEEDED), making it more representative of the hardware used in real-world applications.
2. The number of photodiodes needed is much lower (CITATION NEEDED), requiring only 3 while most existing solutions use an array of at least 8.
3. The data from photodiodes is 3D-formatted, which better preserves temporal information and is more suitable for classifying sequences. An explanation for what 3D-formatting involves can be found in section X.
4. The neural network used to classify hand gestures uses fewer layers than competing solutions (CITATION NEEDED) while maintaining a high level of accuracy and reducing system latency. More information regarding the neural networks tested can be found in section X.

1.2 Research Question

Given the challenges overcome and contributions to the field made, the goal of this project can be summarized with the following research question:

"Which machine learning model yields the lowest error rate for real-time gesture recognition on an Arduino Nano 33 BLE, using 3D-formatted data from photodiodes?"

This question can then be segmented into the following sub-questions:

1. What does "3D-formatting" mean in the context of photo diode data?
2. What does "error rate" mean in the context of machine learning models?

3. What does ”real-time” mean in the context of gesture recognition?
4. How can we ensure that a given machine learning model will run in real-time on an Arduino Nano 33 BLE?
5. How can we determine which machine learning model yields the lowest error rate for gesture recognition?
6. How can we pre-process the 3D-formatted photodiode data to minimize machine learning model error rate in the context of gesture recognition?
7. Why it be useful for embedded devices to be able to process data using machine learning models?

2 Background

2.1 Machine Learning & Neural Networks

Machine learning is a sub-field of artificial intelligence which gives a machine the capacity to ”imitate the way that humans learn, gradually improving its accuracy” (https://www.ibm.com/cloud/learn/machine-learning). More specifically, machine learning models ”learn” patterns and trends from existing data in order to make accurate predictions on new, unseen data.

Neural networks, meanwhile, are a type of machine learning model which draws inspiration from the human brain. These types of models feature ”neurons” organized into layers, with each layer typically being densely connected to the next. However, the structure of different neural networks can vary massively, but their main advantage over simpler models is their flexibility and improved ability to deal with highly non-linear data.

2.2 Machine Learning on Embedded Hardware

Machine learning and artificial intelligence have traditionally been restricted to the realms of high-performance and in turn, high power devices. Unfortunately, this means that it has been previously unfeasible to use these technologies with embedded hardware as it lacks the performance to run machine learning models locally, and lacks the dedicated power to be able to transmit sensor data to a remote processor. However, recent advances into machine learning model compression and optimization have changed this, allowing deep neural networks to be run on devices even powered by coin batteries, meaning that low-power microcontrollers can make sense of sensor data in much more sophisticated ways than previously possible.

The most prominent development in this field has been TensorFlow Lite for Microcontrollers, which is a Python framework specifically made for running machine learning models on microcontrollers. The original Tensorflow has been an extremely popular machine learning framework for over a decade, but the extension to make it work effectively on microcontrollers has only been developed in recent years. Tensorflow, as well as Tensorflow Lite for Microcontrollers, are also specifically designed for developers to work with neural networks rather than other types of machine learning models.

2.3 Hand Gesture Data

In this research, neural networks are used to detect gestures, but to do this, data from some sensor(s) must be fed into the network. Based on this data alone, it must be possible to distinguish which gesture was performed for the neural network to achieve an acceptable classification accuracy.

There are a variety of sensors which could be used to record hand gestures and provide this data. One of these is an accelerometer, which lies on the user’s wrist (typically from a smartwatch) to track the direction and acceleration of the their hand movements. Another option is using a camera to record the user’s hand, which provides a huge amount of information but is susceptible to noise and requires significantly more intensive processing. Photodiodes are sensors which output a signal which increases with the amount of light that hits them. This means they can exploit ambient light by tracking shadows cast by the user’s hand, therefore making it possible to recognize which gesture is being performed. This project uses photodiodes for their much lower cost compared to cameras as well as the fact that they don’t require the user to wear a device on their wrists, unlike accelerometers.

3 System Overview

3.1 3D-Formatted Data

The term ”3D-formatted data” is specific to this research, and must be explained to understand the choice of machine learning models tested. This is best done by comparing it to ”2D-formatted data”.

2D-formatted data can be represented as an image, with some horizontal resolution x and vertical resolution y. In this project, each photodiode outputs values at a predetermined sampling rate over the course of a gesture. This means that we can format said data as a 2D image in which x is the number of photo diodes we use and y is the number of total samples we receive from any of the photo diodes, while the value of each ”pixel” in the image is a reading from a single photo diode at a single point in time.

3D-formatted data can meanwhile be thought of as a video, which splits this 2D-image into a sequence of n frames. 3D-formatting is generally more appropriate when the data is sensitive to time, i.e. when data points should be considered in a specific sequence. Therefore, by 3D-formatting the photo diode data, lower error rates should be achievable as temporal information from the photo diodes isn’t lost.

3.2 Project Pipeline

This research focuses on training an appropriate machine learning model for gesture recognition that can be run on a microcontroller, but it is only part of a larger project pipeline. This pipeline is composed of the following tasks:

1. Optimizing the number and placement of photo diodes.
2. Reading, processing, and sanitizing data from photo diodes.
3. Creating an appropriate dataset for the ML classifier to recognize gestures.
4. Training an appropriate ML model on the created dataset and ensuring it can run in real-time on an Arduino Nano 33 BLE.

Although tasks 1–3 were completed by other group members and are beyond the scope of this research, they all affect the structure of the photodiode data used to train models in task 4. This can be seen with task 1 as the number of photodiodes influences input shape (the size of at least one input dimension) for any machine learning model used to classify gestures. Task 2 affects the number of total samples received from a given photodiode during a gesture, which also affects machine learning model input shape. Finally, task 3 determines how many distinct gestures the system is able to recognize, which in turn determines the number of output neurons that the machine learning models must have.

Based on the findings from these tasks, the final system was implemented using 3 photodiodes and can recognize 8 different gestures, as illustrated in figures X and Y, while each gesture is composed of (VALUE NEEDED) samples from each photodiode.

(FIGURE NEEDED)

Is is also important to note that other design decisions made for each previous task also affect final classification performance, but these are beyond the scope of this paper.

4 Model Design

4.1 Architectures Tested

Recognizing hand gestures based on photodiode data can be thought of as a time-series classification task, which is a type of problem that recurrent neural networks (RNNs) are especially well suited for. RNNs differ from conventional neural networks as they do not process the input in its entirety, but instead process each time step or sample from the input sequentially. Each time step is used to update a ”hidden state”, which is fed back into the RNN along with the next time step. This gives RNNs the ability to exploit the time-sensitive nature of time-series data, as each time step also has temporal data associated with it. This desirable property means that RNNs were the first type of network tested.

Although RNNs are highly suitable for time-series classification, they suffer from the ”vanishing gradient problem”, which has already been overcome by long short-term memory cells (LSTMs) and gated recurrent units (GRUs). To briefly explain this problem, the weight of the hidden state of a time step in an RNN tends to exponentially decrease for future time steps. This means that in practice, RNNs tend to ignore if earlier time steps are out of order, which limits performance for long data sequences (CITATION NEEDED). LSTMs and GRUs solve this issue using so-called ”gates”, which determine what contents of the previous hidden state should be kept and what contents of the current hidden state should be propagated to the next time step, which largely mitigates the vanishing gradient problem as only relevant information is kept, therefore greatly reducing the rate at which hidden state weights decay. Given that LSTMs and GRUs are should effectively guarantee better classification performance than RNNs due to their inherent advantages, these were also investigated.

One issue of using LSTMs and GRUs, however, is that they only accept a single data sequence with multiple channels as input. This is relevant as with 3D-formatted data, each time step is a 2D frame, which can’t be used as input for these types of neural networks as they only expect a single value from each channel per time step, i.e. a 1D array. Therefore, each frame has to be flattened first, which removes some spacial data. This issue can be solved by using convolutional LSTMs, which use convolutional kernels in place of traditional neurons, treating each time step as an image, which is exactly how 2D frames should be processed. Due to this, both 1D and 2D convolutional LSTMs were investigated for this paper, with the main difference being that the former performs 1D convolutions on each row of the frame individually, therefore keeping the data from each photodiode separate, while the latter performs 2D convolutions on the entire frame.

Finally, transformer encoders were also tested for this project as a novel alternative to RNN based architectures. Transformers are typically used in language translation and are made up of two parts, an encoder and a decoder. The encoder converts the input into an internal representation, which can then be manipulated and decoded back into the the same data format as the input. However, by using only the encoder, the internal representation of the input can instead by directly mapped to a gesture, therefore making this architecture suitable for classification problems. Transformers also use a concept called self-attention which identifies which elements in a sequence carry most semantic value based on the other elements of the sequence, which is fundamentally different to the hidden state mechanism used by RNNs and their derivatives.