Goal: to practice embedded machine learning engineering.

Ideas: Robotic Arm/Vehicle.

Background: having thought about it the only time it makes sense to do machine learning and/or any intensive computation on an embedded device as opposed to the cloud (server/computer) is in real time application wherein the network delay is unacceptable or internet access is restricted/intermittent/slow.

Microcontrollers are slow (and memory limited), especially in comparison to GPUs. Moreover, there is significantly less ML/DL library support on microcontrollers (although not none, which is impressive).

Overwhelming majority of IOT in Healthcare applications that I have seen so far do not have that requirement. They either do real time computation on a large processor (medical equipment) or they send data to a smarphone, smartwatch, etc. which then may send it further to the cloud/server.

**Therefore, instead of requiring myself to do ML on a microcontroller, the only requirement is to use ML on data attained from embedded system (whether directly within embedded system, or on phone/computer/server does not matter).**

Alternative Idea: FPGA accelerate ML. Same as above but use an FPGA instead.

Robotic Arm / Vehicle:

Pros:

* Some ML/DL support (Tensorflow Lite)
* Arduino + Raspberry PI have extensive resources, inexpensive
* I have past Arduino + Raspberry PI experience
* Uses C/C++ Coding
* Simple projects can be done in ~ 50-100 hours

Cons:

* Not cutting edge
* Constrained to typical sensor data (no medical records, genomics)

FPGA NN:

Pros:

* Cutting Edge Applications
* Can utilize any data
* Power Efficient
* Extremely Versatile

Cons:

* Similar difficulty to designing a chip. Requires strong computer engineering knowledge.
* Programming FPGA requires Verilog (digital circuit level). Bluespec, OpenCL, etc. can be used, but still tricky.
* Completing a simple project is likely to take over 100 hours (due to time spent learning Verilog, reviewing digital circuit design, and etc.).
* Performing ML/DL on FPGAs is a hot topic which is being pursued by several companies and many research groups. It is typically done by electrical and computer engineers with deep hardware knowledge, not computer scientists.

Raspberry PI / Arduino Projects Focus on building small robotic, sensor, and other breadboard systems. Example projects: line following robot, controllable robotic arm, digital thermometer, etc.

FPGA is focused on acceleration. It is hardware reprogrammable unlike microcontrollers. The big drawback of FPGAs is that they require significant computer engineering experience to use.

That said FPGA can be used in a fashion similar to a microcontroller once it is mastered (to collect data from sensors (input), control a device (output), etc.)

**FPGA and Microcontroller**

From our previous discussions, it is clear that for consumer specific projects that require customization, FPGA or microcontroller is preferred. Customization is the main advantage of FPGA.

**Microcontroller**

Microcontroller is a CPU with RAM, ROM, I/O ports and other peripheral devices integrated as one, to perform specific tasks. Some examples of microcontrollers are Arduino microcontroller and Pic Microcontroller.

**Programming a Microcontroller**

Microcontroller is programmed using assembly language. High-level programming languages are also used such as JavaScript, Python, and C. Steps involved in programming a microcontroller are

1. Writing a program code, usually done in C
2. Compile the code in a Compiler
3. Upload the compiled version of the program to the microcontroller

Microcontroller programming is simple.

**Advantages of Microcontroller**

* Microcontrollers are simple to program
* Microcontrollers are the best for a simple and hardware specific application. That is if only limited hardware is required
* Microcontrollers are cost effective than FPGA

**Disadvantages of Microcontroller**

* Microcontrollers can perform limited tasks. Because they have a limited instruction set. The firmware loaded into the controller can make the controller perform only the pre-loaded instructions.
* Microcontroller relies on sequential processing. That is one instruction at a time. Hence programming using interrupts (ISR) becomes complex and hence microcontroller consumes considerable time in executing instructions.
* Hardware is limited. Designer can utilize only the hardware available on the board.

**Field Programmable Gate Array**

FPGA is an integrated circuit that contains millions of logic gates that can be electrically configured, through programmable interconnects to perform the required task. Xilinx, Intel FPGA packages are available in the market.

**Programming a FPGA**

FPGA programming is comparatively complex than that of a Microcontroller. FPGA programming requires specialized software such a Xilinx, Intel Quartus. FPGA is programmed using Verilog or VHDL. System Verilog is usually used for verification. FPGA programming follows hardware design flow. Steps in FPGA programming are

1. Creating a Verilog code or VHDL code
2. Create a module in the software
3. Complete pin assignments
4. Create an SDC file. SDC file is design constraint file. This file contains timing and design constraints
5. Convert netlist into Binary Format
6. Place and Route
7. Compile the code – Generate a bit file
8. Program the FPGA.
9. Analyse the report and reprogram.

**Advantages of FPGA**

* In FPGA the hardware itself is programmable. Meaning new hardware or logical functions can be programed by altering the programmable blocks in the FPGA by installing a new FPGA firmware.
* FPGA does not have a fixed instruction set.
* FPGA process instruction in parallel processing. This capability of FPGAs allows us to control the interrupts effectively by using Finite State Machines (FSMs).
* Trial and error method is possible in FPGA. This allows a steep learning curve.

**Disadvantages of FPGA**

* It is complex to configure an FPGA. As the designer needs to compile all the codes from scratch and then convert them into machine language.
* Power consumption. Since not all blocks may be utilized for designing an application, FPGA tends to consume more power than a Microcontroller or an ASIC.
* For a simple application, FPGA can be bulky and costly.
* No internal oscillator: Clock for FPGA has to be provided from an external source.

**Comparison table: FPGA vs Microcontroller**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **FPGA** | **Microcontroller** |
| **Flexibility** | Hardware and firmware reprogrammable. Superior customization. | Reprogramming is possible in firmware only |
| **Programming Firmware** | Comparatively complex | Simple programming |
| **Tools** | No portability across tools. | Open source tools available |
| **Cost** | Costly | Cost effective |
| **Processing power/speed** | Higher | Lower |

Decision:

Do all of the above. Attain an Arduino, Raspberry Pi, and a FPGA.

Do a simple project: LED Blinking – Hello World of Embedded Systems. Use this project to develop a workflow.

Then move onto a more complex project involving a sensor. – I think some sort of music visualizer could be very cool (microphone + display)! Measuring (& displaying) body temperature, heart rate, pressure, and etc. could be awesome as well (various sensors). Emotion (face) recognition is another neat project (camera).

Then slowly build up the sensor project into an supervised or unsupervised ML project.

Cluster or Classify music genre based on audio or visualization data. Infer stress level based on voice audio. Classify/Cluster activity based on heart and breathing rate. Classify emotion based on face images/video.

Optional: conduct a reinforcement learning project. This will introduce control theory and will likely involve robotics.

Attain additional components as necessary (ex: breadboard, battery, wires, resistors, LEDs, etc.).

Use the remainder of my PhD to develop these skills (~1.5 years).

Completion of this project alongside my PhD will prepare me to conduct MLE with both typical sensor and genomic data.