Exploring a potential solution for deploying machine learning technology in embedded systems

Background

Machine learning (ML) is a field of computer science that enables computers to learn without being explicitly programmed. In practical applications, a machine learning model is created and tailored to solve a specific problem through a training process. Models can take a variety of different inputs and provide a range of outputs based on the underlying patterns they find during the training process. For example, a model designed to predict the temperature of the climate on a given day could be trained using climate data from previous years.

Machine learning models are composed of three main components, each serving a unique function in the training process:

1. **Decision-Making Process:** An ML model always includes a decision-making process. This could involve predicting the number of sales a company will have in the next quarter or classifying an image as containing a chair or a table.
2. **Loss Function:** To improve, the model needs a way to evaluate the efficiency of its decision-making process. Loss functions quantify the accuracy of the decision-making-process by comparing its results to manually labelled data. These functions are predefined by a human developer and remain constant throughout the training process of the model. Various metrics can be produced by different loss functions, with a popular one being Mean Squared Error (MSE). MSE calculates the squared difference between the estimated and correct answers determined by developers, where squaring removes any negative numbers.
3. **Optimization Process:** After evaluating its performance, the model must modify its decision-making process to minimise the error calculated by the loss function. The optimizer is a function that attempts to minimise the calculated loss as much as possible. It does this by adjusting the decision-making-process's parameters after each iteration of training. This is largely a trial-and-error process. Each iteration of the training process brings slightly better results. However, excessive iteration can result in 'overfitting.'[1]

[1] Overfitting occurs when a model has been trained on a particular dataset for so long that it is unable to generalise what it has learned to new data, simply memorising the answers to the training data.

Categories of Machine Learning

There are four broad categories of machine learning:

- **Supervised Machine Learning:** This involves the highest level of human intervention. The model updates its parameters based on labelled data provided during training.
- **Unsupervised Machine Learning:** This method involves training models on data that lacks labels, allowing the model to identify patterns which may not be apparent to the human eye on its own.
- **Semi-Supervised Machine Learning:** This approach uses a small amount of labelled data and a large amount of unlabelled data, combining elements of both supervised and unsupervised learning.
- **Reinforcement Learning:** This method involves training a model to make sequences of decisions by rewarding desirable outcomes and penalising undesirable ones.

Rationale

Machine learning has vastly increased in popularity and relevance recently, with developments in image processing, generative AI, and classification built on over six decades of research. From media creation to disease diagnosis, the influence of machine learning technology on daily life has grown dramatically. However, machine learning technology has not seen much deployment in embedded systems. An embedded system is a computer that works closely with mechanical parts to achieve a specific purpose, such as a washing machine or a microwave.

I believe that embedded ML is a field with great potential, as groundbreaking developments are already being made, such as in self-driving cars. Exploring this field could push the boundaries of what is possible in consumer electronics, allowing product designers to focus more on making complex functions a reality rather than on the minute details of feature implementation. This is especially true with the rise of more beginner-friendly machine learning programming tools which simplify the model creation process, shortening the gap between design and implementation.

To address the challenge of implementing machine learning in embedded systems, I decided to build a small robot that would take an input and make a prediction using a machine learning model. I chose to use an Arduino system for this project. An Arduino is a simple computer, allowing electronics developers to construct prototypes before settling on a final design.

The Arduino caught my attention because it uses a microcontroller instead of a microprocessor to execute computer programs. A microprocessor, also known as a central processing unit (CPU), is the component of a computer that carries out most calculations and logical operations. Microprocessors are optimised for general-purpose applications, executing instructions quickly and are commonly found in devices such as laptops, smartphones, and tablets. In contrast, a microcontroller contains most of the components of an entire computer system on a single chip. Microcontrollers are typically designed for embedded systems, prioritising power efficiency and small size over performance. For this reason, I chose to build my artefact around an Arduino, as I thought it could accurately model an embedded system.

Machine learning and recognition technology

Recognition technology has advanced dramatically through machine learning methods. Many consumer-grade electronics, such as smartphones and laptops, use biometric recognition like fingerprint and face recognition as security measures. I determined that this application of machine learning technology was appropriate for my project because (1) it has a broad niche with plenty of real-world applications, and (2) it is a well-explored field with many available tools and resources.

Once I established the purpose of embedded machine learning in my project, I needed to choose its output. To maintain the theme of embedded systems, I wanted the artefact to work closely with mechanical parts. My initial idea was to create a robot that uses face detection technology to align a nerf pistol with a person's face before firing a foam dart. However, I modified this idea to create an artefact with broader appeal and lower complexity, ultimately deciding that the output would be a photograph instead. In January, I began expanding on my initial project idea by researching the components and programming tools I would need.

Selecting components

Motor: After researching the functions and use cases for many types of electric motor, I decided that a servo motor was appropriate for this project. Unlike other types of AC and DC motors, servo motors can be controlled precisely with electric signals. This lends itself to being controlled with programs running on the Arduino. I chose the SG90 servo motor because it met the torque requirements I had calculated, with a maximum rotational torque of 2 kg/cm.

Arduino: As previously justified, I chose the Arduino due to its simplicity, which more accurately represents the capabilities of computers found in embedded systems. Additionally, my prior experience with Arduino provided an advantage, easing the programming aspect of this project and helping me meet deadlines.

Camera: I selected the OV7670 camera module because it has a low output resolution of 640x480 pixels which could reduce the processing load on the Arduino. During my January research phase, I also found a guide on using this specific camera with an Arduino, which influenced my decision.

Methodology and Findings

At the beginning of my research project, I identified the need to expand my knowledge about machine learning. I began reading general literature regarding machine learning and spotted many new key terms. I decided to explore one of these further: application programming interface (API)[2]. I considered basing my artefact around a machine learning API, if the face detection program could not run locally on the Arduino. To my surprise, I discovered that this suspicion was a reality. By exploring the literature in embedded machine learning, I found that the processing capabilities of the microcontroller of the Arduino are simply not enough to handle machine learning programs. In fact, prototypes in the field of embedded machine learning are typically made with another small computer which works closely with electronics called a Raspberry Pi. Despite their very similar appearances, the Arduino and Raspberry Pi have a key difference between them which leads to vastly different levels of performance. Chiefly, the Raspberry Pi uses a microprocessor for executing instructions, whereas the Arduino uses a microcontroller. As I outlined earlier, this simple change can mark the difference between the computing performance of a vending machine and a personal computer. At this point, I knew I would need to follow a similar model to the APIs I had discovered shortly beforehand. The processing of machine learning programs is offloaded to a computer more capable than the Arduino, and the outputs are simply returned to it.

I attacked this problem with a new design in mind. Inspired by APIs, the role of the Arduino radically changed to simply acting as a conduit for the data to be exchanged between the artefact and the computer to be running the machine learning program. From this new framework, I developed a new procedure that would satisfy my initial objectives for the project.

1. The camera module captures an image and sends it to the Arduino.

2. This image is received by the Arduino and sent to the computer to be analysed.

3. The image is passed through the object detection algorithm stored on the computer to find any faces within it.
4. Next, depending on the position of the face within the image, the computer will take one of three actions:
    a. If there is no detected face in the image, the computer will request the next image to be sent from the Arduino and repeat the process.
    b. If the detected face is far from the centre of the image, the computer will request the Arduino to move the motors in a specified direction to draw the face closer to the centre.
    c. If the detected face is within a tolerated distance from the centre, the image is stored onto the computer.

After conceptualising this design, I researched the technologies that would be necessary to move the project forward. The model of Arduino I used contains an integrated Bluetooth and

[2] An API is a program which allows applications to exchange data or functions with each other. For example, a web developer creating a website could use an API from to easily implement features from other sources, such as a ‘sign in with Google’ feature, with little effort.

Wi-Fi transceiver, in addition to a USB-C port. As a result, for communication between the computer and the Arduino, I considered these transmission media. After careful deliberation of the strengths and weaknesses of each medium, I decided to use 2.4 GHz Wi-Fi. I have outlined the main strengths and weaknesses I considered in *Table 1*.

Transmission medium	**Advantages**	**Disadvantages**
Bluetooth	● Convenient. Provides a direct connection between two devices.	● Data transmission rate is too low. Maximum rate of 2Mbps. This would result in a single image taking at least 3.69 seconds to transmit. ● Connection deteriorates quickly over large distances. ● More likely to experience data loss.
2.4 GHz Wi-Fi	● Devices can be connected to each other wirelessly. ● Higher maximum data transmission rate of 150Mbps.	● Introduces more hurdles to overcome in implementation. ● Both devices must be connected indirectly via a wireless access point and a router. ● More likely to experience data loss.
USB – C	● Physical transmission medium, more reliable. ● Less likely to experience packet loss.	● Also limited by a low data rate of 115250bps

Table 1

After this decision, I researched how I can enable my personal computer and Arduino to communicate programmatically over Wi-Fi. After identifying programming tools named libraries which could enable this for my personal computer and the Arduino, I employed test programs which existed for both libraries. In doing this, my personal computer was able to connect to itself over Wi-Fi to send itself a message. However, my experiments with Wi-Fi connectivity on the end of the Arduino were unsuccessful. Connections between existing websites and my own computer refused to be established.

Following this, I took a step back and tried setting up the camera with the Arduino. To extract camera data, I looked into two Arduino libraries extensively: *OV767X* and *Adafruit OV7670*. Little official documentation was available, so I inferred the majority of what I learned about them from published code examples and the files that made up the library itself. Eventually, I interfaced the camera by following its circuit diagrams and running a test program for its library, which produced a slew of errors. After troubleshooting and scouring forums for a solution to my problem, I discovered that the model of Arduino I was using was incompatible with the libraries I had sought out.

Conclusion

There are still many hurdles in the world of embedded machine learning which must be overcome before this technology can become more widespread in systems such as household appliances. However, more computationally powerful embedded systems like self-driving cars are landmark achievements in the right direction. Offloading the machine learning process to more computationally powerful computers with APIs could allow microcontroller-based embedded systems to carry out complex functions without any additional processing power. A key example of this is the Amazon Echo series of devices, which allow users to interact with an AI assistant named Alexa. These devices record audio of the user's request and send it via the Internet to servers owned by Amazon which interpret the sound and return the corresponding commands to the Echo device. This allows complex functions to be delivered to a consumer with a device with limited computing capabilities.