Lightweight Hyperdimensional Computing (HDC) Algorithm for Microcontrollers

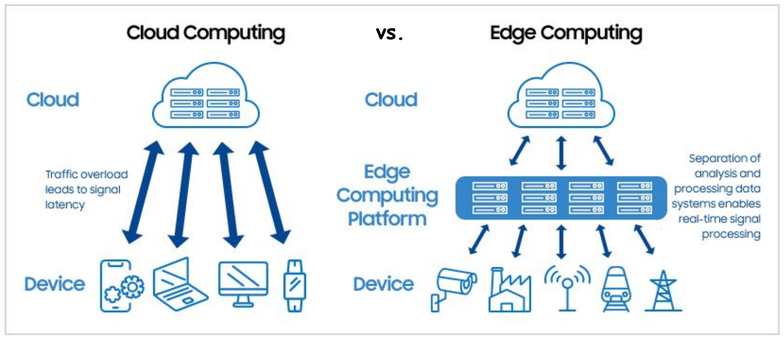
UC San Diego
JACOBS SCHOOL OF ENGINEERING
Computer Science and Engineering

Aammya Sapra, Duanhui Li, Jennifer Tanurdjaja, Jezebel Yangari
Advisors: Tajana Rosing, Xiaofan Yu

ERSP
UC San Diego
Early Research Scholars Program

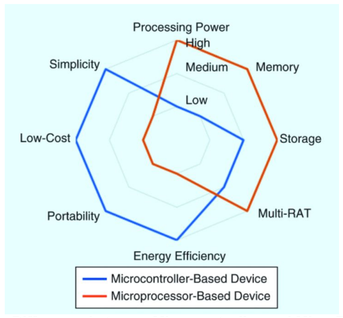
Motivation

Edge computing is the collection and processing of data collected by edge devices (e.g. sensors) in a local network instead of the cloud.



Edge computing reduces network pressure and increases security, but there is a lack of a universal algorithm that can be implemented across edge devices of varying hardware/software limitations.

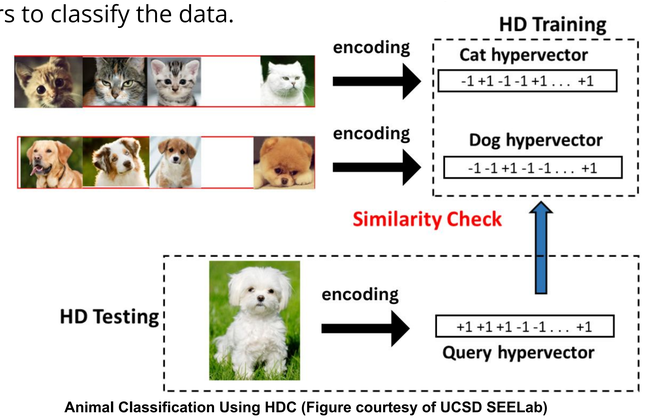
Previous works have implemented edge computing algorithms on microprocessors but not microcontrollers due to hardware limitations.



Hardware Difference between Microcontroller and MicroProcessor (Sanchez-Iborra and Skarmeta, 2020.)

Hyperdimensional computing (HDC) is a promising learning paradigm for implementing a lightweight algorithm across varying edge devices.

HDC casts the input data into large binary vectors (10,000+ bits), called **hyper vectors,** and performs arithmetic operations on the hyper vectors to classify the data.



Animal Classification Using HDC (Figure courtesy of UCSD SEELab)

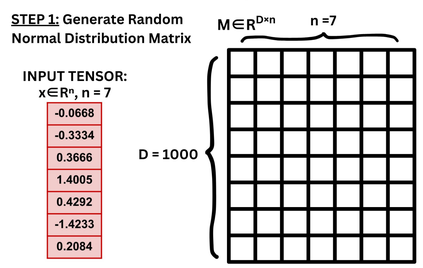
Problem Statement

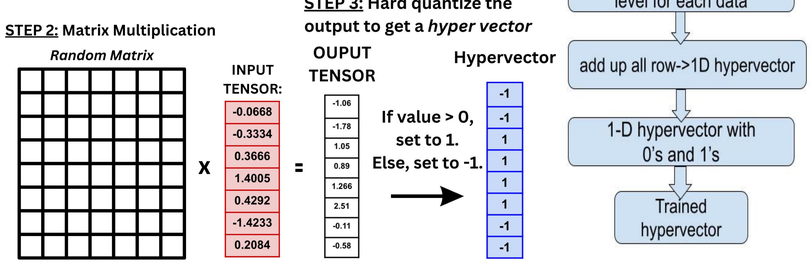
The aim of this project is to create a **lightweight** HDC encoding method that can be **implemented on variety of low powered devices (i.e. microcontrollers like ESP8266 and Arduino Uno)**.

Methodology

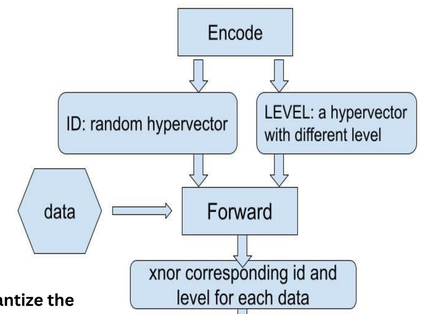
We implemented two hyperdimensional computing encoding methods, **Random Projection** and **ID Level,** in C**.** Our lightweight implementation was able to run on microcontrollers (**ESP8266 boards and Arduino Unos**), in comparison to previous Python implementations that support microprocessors at best.

Random Projection



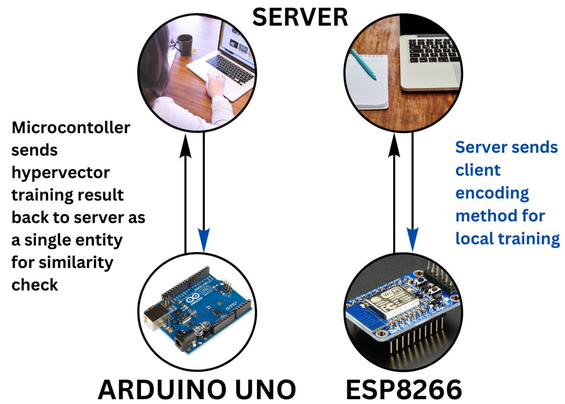


Id-level



These encoding methods map the input data to a high dimensional space as a hyper vector, where more similar data inputs are nearer to one another.

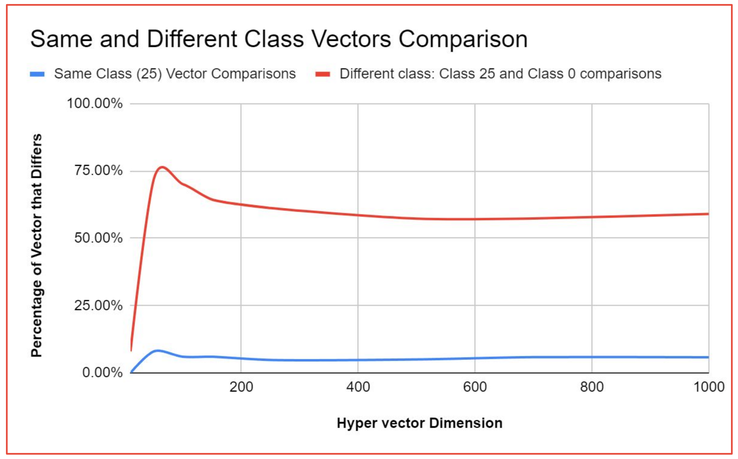
The encoding methods were deployed onto **2 Arduino Uno boards** and **2 ESP8266 board** for testing and evaluation.



Testing and Results

To test the encoding, we select tensors from the **ISOLET** dataset:

- Select **two** tensors from the **same** class
- Select **one** tensor from a **different** class
- Increase generated hyper vector dimension
- Use **encoding** to generate hyper vectors for the tensors
- Compare the **three** hyper vectors to one another



Tensors that are in the **same** class will have a lower percentage in the difference of their hyper vectors.

Tensors that are in **different** classes will have a higher percentage in the difference of their hyper vectors.

Laptop vs. Arduino Uno vs. ESP8266

Energy consumption	Laptop	Arduino Uno	ESP8266
Random Projection	0.045 W	—--	0.0496 W
ID-Level	0.17 W	0.091 W	—--

This is a comparison of energy consumption of the two encoding methods on different devices.

$$Energy\ Consumption = Execution\ Time\ of\ Encode \times Power\ Consumption\ of\ Device$$

Acknowledgements

We would like to thank Professor Mai ElSherief and Vaidehi Gupta for the opportunity to participate in ERSP as well as their support over the course of the year. We would also like to thank Professor Tajana Rosing and Xiaofan Yu for their guidance and support throughout this project.