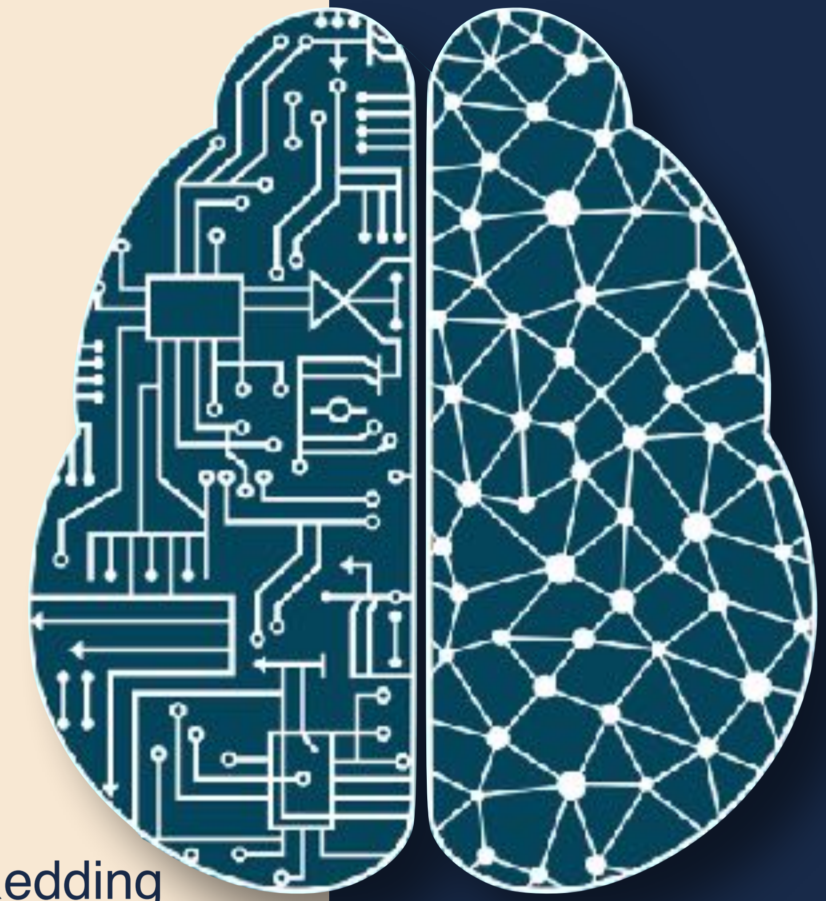
NET4us ’23, October 6, 2023

EmbHD

A Library for Hyperdimensional Computing Research on MCU-Class Devices

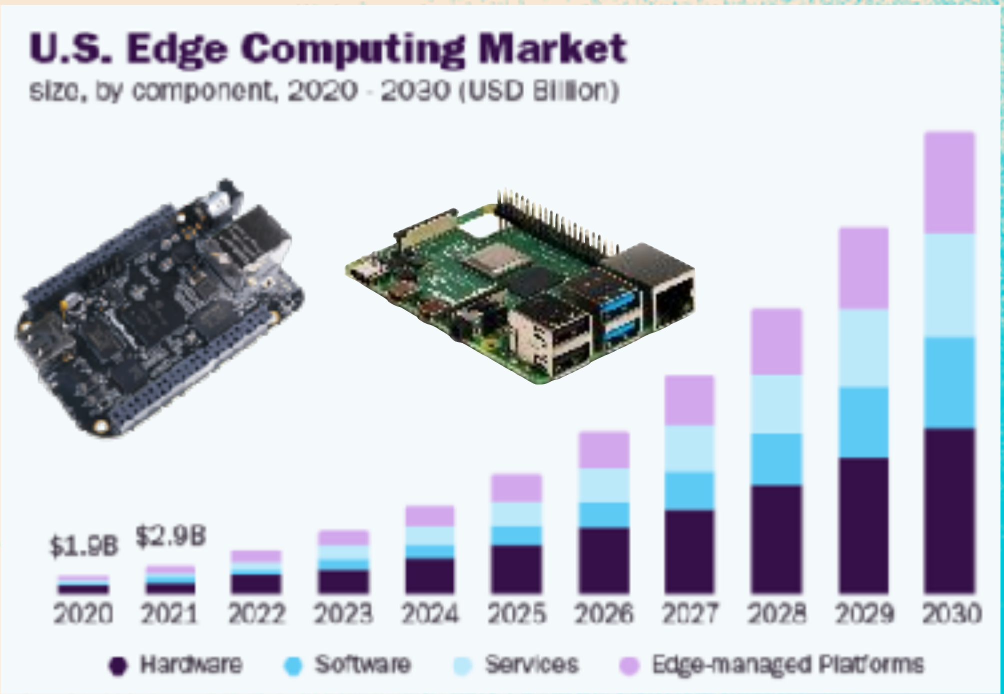


Alexander Redding
Xiaofan Yu
Shengfan Hu
Pat Pannuto
Tajana Rosing

UC San Diego

Edge Computing

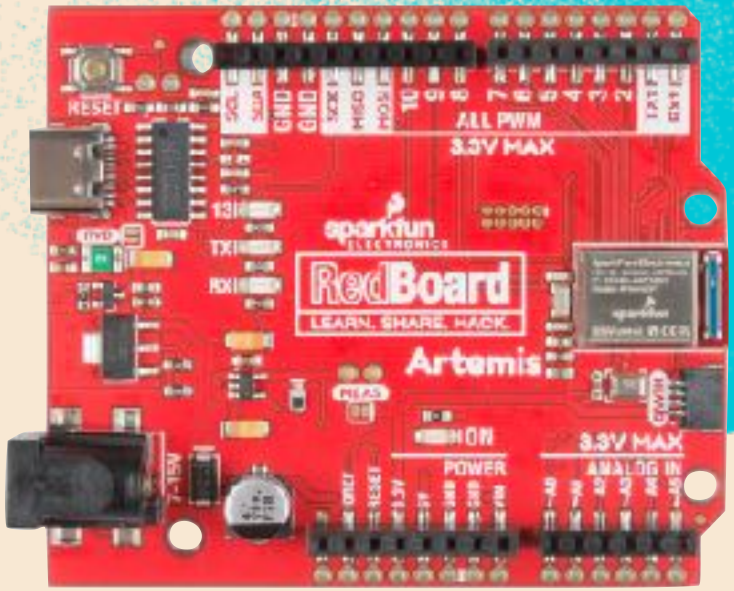
- The global edge computing market size is expected to expand at a compound annual growth rate (CAGR) of 37.9% from 2023 to 2030[1]
- Benefits of on-device inference and training
  - Timely decision making
  - Potentially lower power due to less comm. costs
  - More secure
- Typically *low-power SoCs w/ multi-core application processors
  - ***Tethered to power source, limited long-term mobility**



1. https://www.grandviewresearch.com/industry-analysis/edge-computing-market

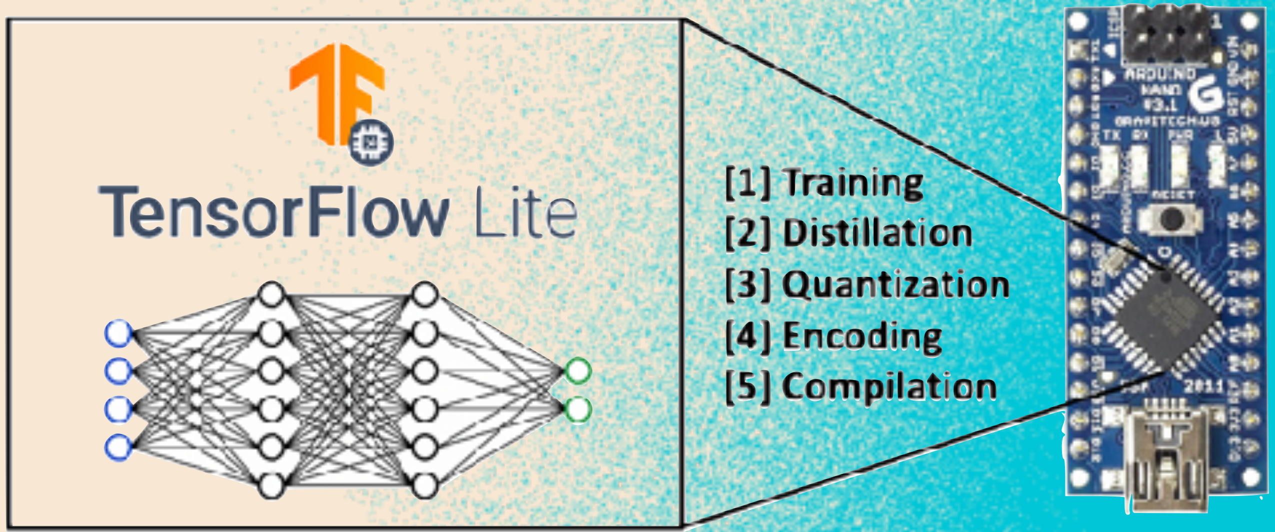
Modern Microcontrollers

- Historically simple 8/16-bit
- Newest generation has seen transition to more capable 32-bit processors (ex, ARM Cortex-M family)
- Not as powerful as multicore SoCs
- But…
    - Unparalleled in power-efficiency
    - Lower cost



TinyML

- Running optimized ML models on MCUs
  - Neural networks
- MCU-class hardware:
  - < 1mW, <100KB memory, 1-2MB flash
- MCU runs same tasks as multi-core system



Source: [Matthew Stewart]()

Neural Networks

- Training is resource-intense
- Noise sensitive
- Best accuracy

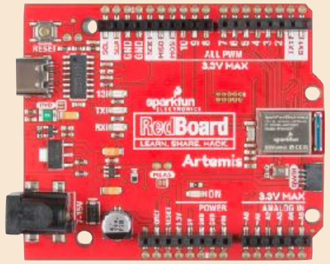
Hyperdimensional Computing

- Training is fast + efficient
- Robust to noise
- Resilient

Our Work

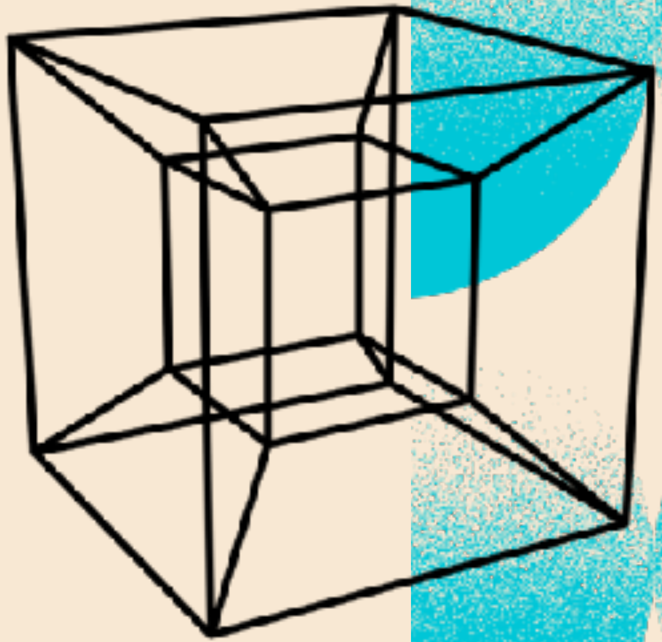
TensorFlow Lite Micro

EmbHD



Hyperdimensional Computing

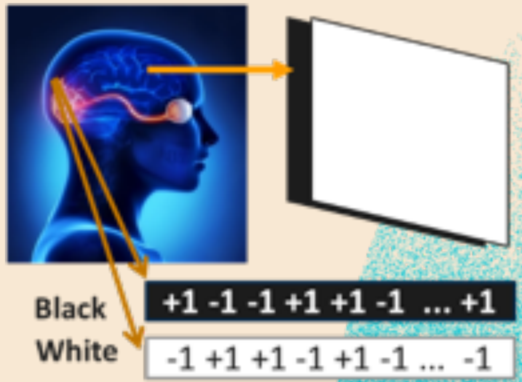
A crash course

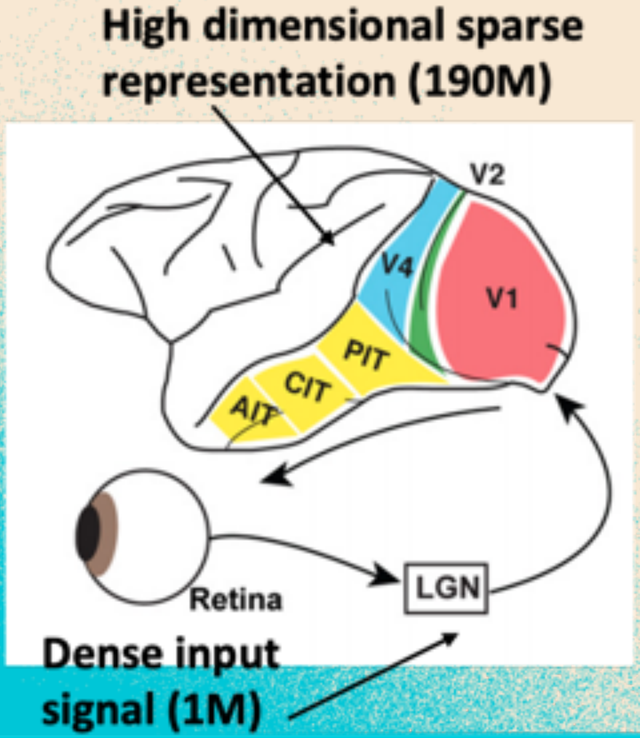


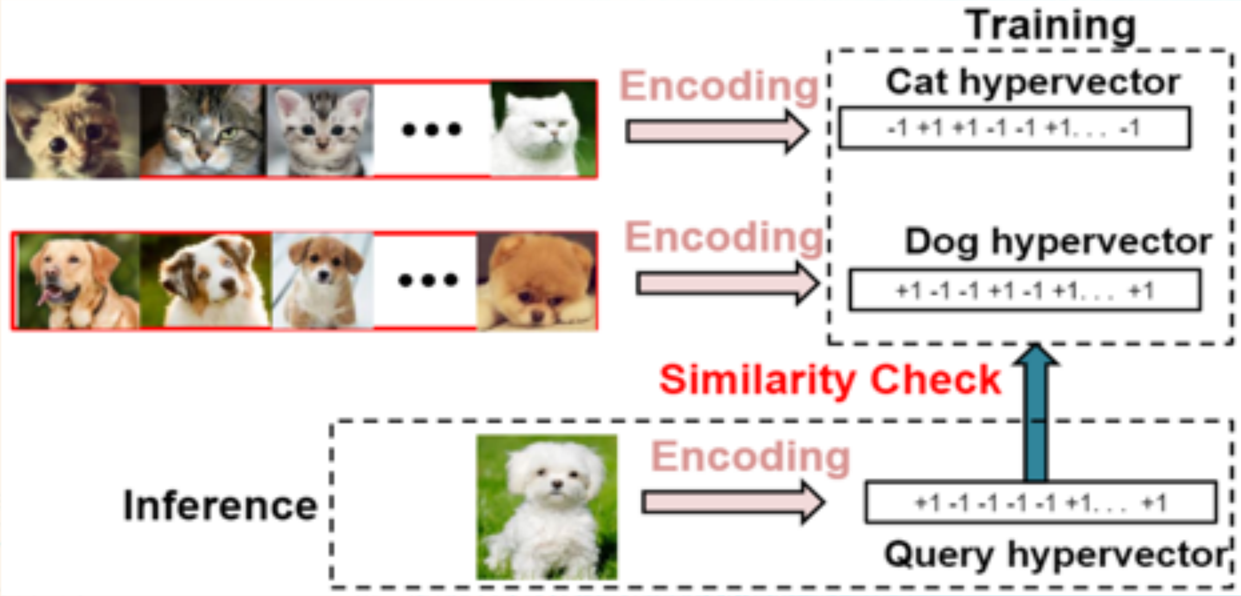
UC San Diego

What is Hyperdimensional Computing?

Dense sensory input is mapped to **high-dimensional sparse representation** on which brain operates [Babadi and Sompolinsky 2014]







Benefits of HD computing:

- Easy-to-parallelize and hardware-friendly operations
- Fast single-pass training
- Energy-efficient & robust to noise

Operations

Binding 

Combine 2 HVs into 1

$$\boxed{\otimes : H_a \times H_b \rightarrow H_y}$$

Ints: Cross product

Binary: XOR

Bundling 

Create unordered collection of HVs

$$\boxed{\oplus : H_a + H_b \rightarrow H_y}$$

Ints: Sum

Binary: OR

Operations

Similarity

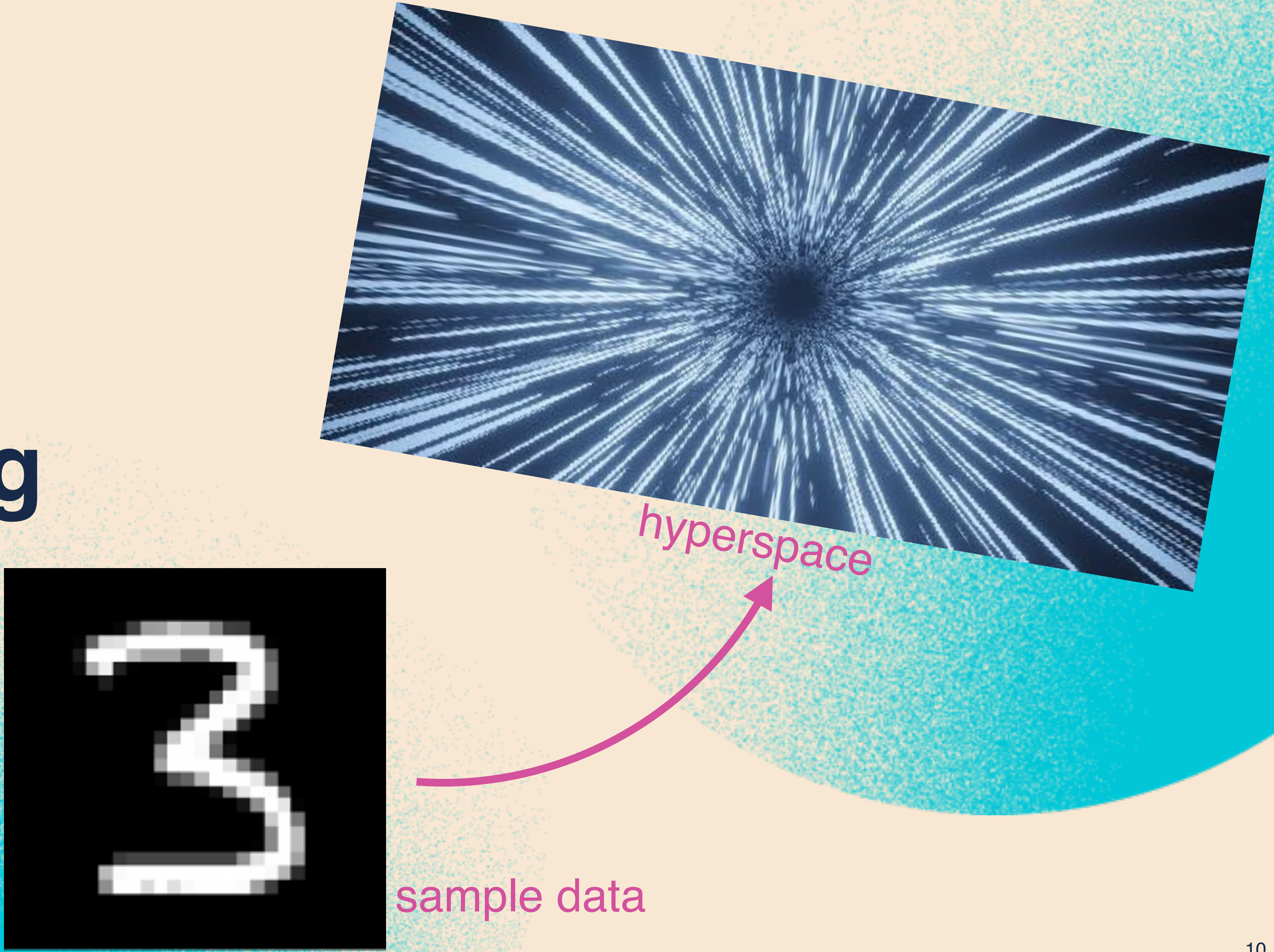
How close are 2 hypervectors in hyperspace?

$$\boxed{\delta(H_a, H_b) \to d}$$

Ints: Cosine-similarity

Binary: Hamming-distance

Encoding



Preserve data correlation between dimensions

ID-Encoding

pixels

intensity	45	50	54	57	39	…
location	0	1	2	3	4	

Similar pixel intensities = Similar hypervectors

Location = No correlation

hypervectors

ID-Encoding

Random

0	100101110011…
1	010101001111…
2	111000111000…
…	

Level

0	011011011001…
1	010011011001…
2	010011011101…
…	

Pixel intensities = Level Hypervectors

Location = Random Hypervectors

 =

feature vector (pixels)

45	50	54	57	39	…
0	1	2	3	4	

ID-Encoding

Random

location 3 $\boxed{001111000011\ldots}$

$\otimes$

Level

intensity 57 $\boxed{100001111000\ldots}$

$HV_0 \oplus HV_1 \oplus HV_2 \oplus \ldots$ = encoded image

encoded pixels

(hypervector)

encoding

Training

sample hypervector 110010001100…

$\oplus$

class hypervector 001111000011…

Repeat for all samples

encoding

Inference

sample hypervector 110010001100…

δ

label

class hypervectors

0 111100110011…

1 010010101101…

2 101011010010…

Prediction = Most similar

TensorFlow Lite Micro

- Traditional Neural Networks
- Inference
- Optimal performance

Previous Work

EmbHD

- Hyperdimensional Computing (first)
- **Training** + Inference
- Enable new capabilities

Our Work

Not a replacement

- We introduce EmbHD, the first library for Hyperdimensional Computing (HDC) on MCU-class devices
- EmbHD is a tool for researchers to test HDC conveniently on MCUs

EmbHD System Design

- ~HDC virtual machine written in C
- Built on generic matrix representation
  - HDC operations map to matrix operations
  - Maximum re-usability for future additions (ex, binary NNs)
- ARM Cortex-M4 DSP instruction optimizations

```
MData  binary_hv_data[313];
Matrix binary_hv = {
     .dtype  = MBin,
     .height = 1,
     .width  = 10000,
     .size   = 313,
     .data   = binary_hv_data;
};
```

D = 10, 000 Binary Hypervector

binding

```
MMult( dst, row/HV, src0, row/HV, src1, row/HV );
```

```
extern Matrix Random;
extern Matrix Level;
extern Matrix weights;
extern Matrix tempint8;
extern Matrix tempbin;

void encode(const uint8_t * image){
    for (unsigned int pix = 0; pix < IMG_SIZE; pix++){
        MMult(&tempbin, 0, &Random, pix, &Level, image[pix]);
        if (pix == 0) { // Reset
            MConvert(&tempint8, 0, &tempbin, 0);
        } else {
            MConvert(&tempint8, 1, &tempbin, 0);
            MAdd(&tempint8, 0, &tempint8, 0, &tempint8, 1);
        }
    }
}
```

hyperspaces (rows are hypervectors)

← binding

bundling*

*majority rules

EmbHD Workflow

Torchhd

1. Pre-generate hypervectors w/ Torchhd* (random, level, class, temp)
   *Python library for Hyperdimensional Computing built on PyTorch
2. *Optionally: train model in Python (ie, fill class hypervectors)
3. EmbHD Python library to export Torchhd hypervectors to C-header file
4. Write C source w/ EmbHD library functions for encoding
5. Compile and deploy

Generating 100 random hypervectors of $D = 10{,}000$

```
import torch, torchhd
import export_matrix_lib

DIMENSION = 10000
NUM_HV = 100

hv = torchhd.random(NUM_HV, DIMENSION)
convert_mdata(hv, "randhv", static=True)
```

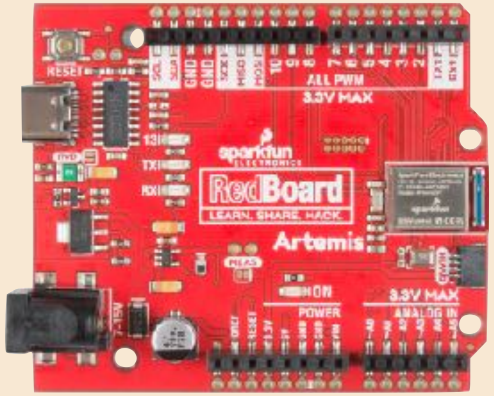
EmbHD

- SparkFun Redboard Artemis
- Cortex-M4 DSP Instructions
- Binary Hypervectors
- MNIST + ISOLET
- Baseline HDC

TFLite Micro

- SparkFun Redboard Artemis
- ARM CMSIS-NN Kernel
- Float and 8-bit int
- MNIST + ISOLET

Evaluation



Results

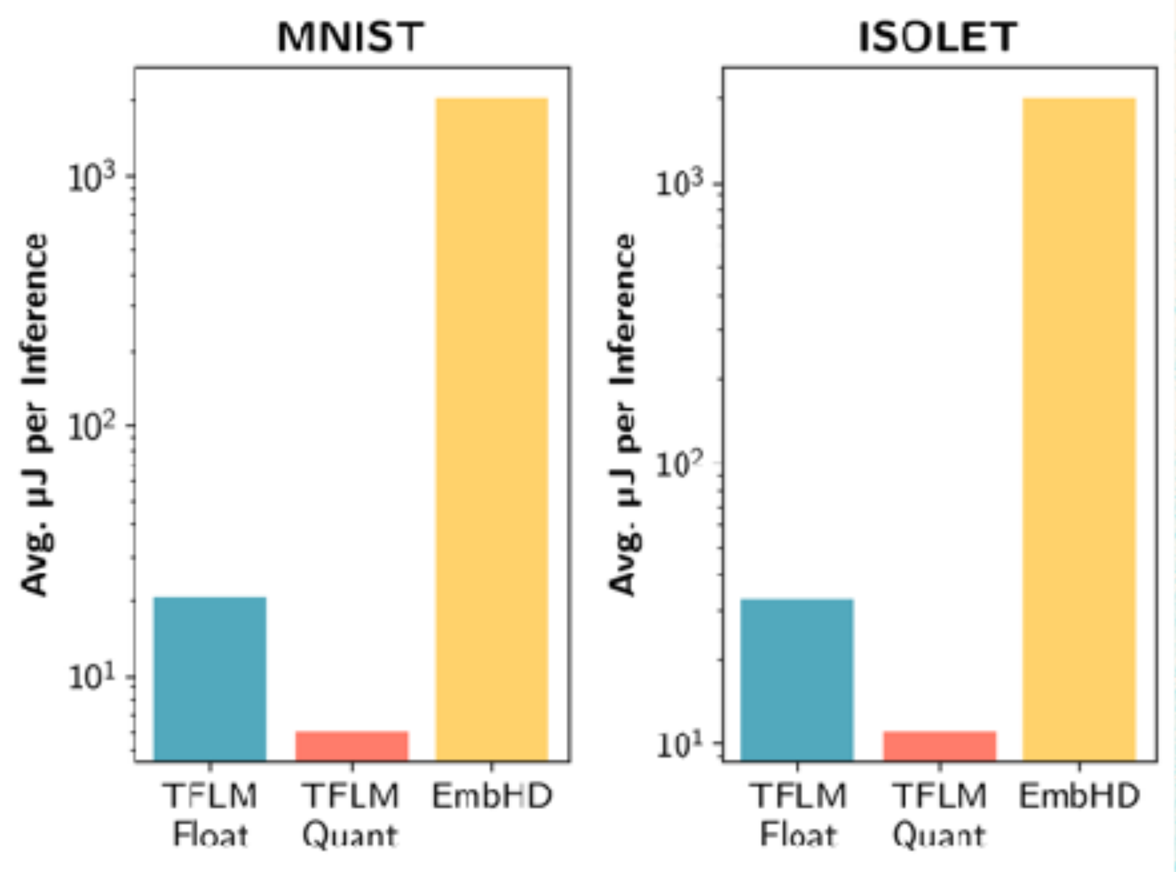


Table 3: Performance Results of EmbHD and TFLM

Dataset	Library	Parameters	Accuracy	µJ per Inference
MNIST	TFLM Float	1 hidden layer of 64 nodes	96%	20.6
	TFLM Quant		96%	6.05
	EmbHD	$D = 7,000$	80%	2036.68
ISOLET	TFLM Float	1 hidden layer of 128 nodes	95%	32.82
	TFLM Quant		95%	11.17
	EmbHD	$D = 10,000$	81%	1999.86

Conclusion

- In this paper, we introduce EmbHD, the first library supporting Hyperdimensional Computing on MCU-class devices
- Hyperdimensional Computing is a new brain-inspired computing paradigms that features lightweight operations, single-pass training and robustness to noise.
- We conduct preliminary experiments on the SparkFun Redboard Artemis board
- EmbHD is **NOT** a replacement for traditional ML libraries (TFLite Micro), but instead a tool for researchers to evaluate HDC for deployment

Thank You

Check out EmbHD: github.com/alexredd99/EmbHD