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 2030¹

• 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
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

\[ \otimes : H_a \times H_b \rightarrow H_y \]

Ints: Cross product
Binary: XOR

**Bundling**

Create unordered collection of HVs

\[ \oplus : H_a + H_b \rightarrow H_y \]

Ints: Sum
Binary: OR
Operations

Similarity

How close are 2 hypervectors in hyperspace?

$\delta(H_a, H_b) \rightarrow d$

Ints: Cosine-similarity
Binary: Hamming-distance
Encoding

code

sample data

hyperspace
ID-Encoding

Preserve data correlation between dimensions

Similar pixel intensities = Similar hypervectors
Location = No correlation
ID-Encoding

Pixel intensities = Level Hypervectors
Location = Random Hypervectors
ID-Encoding

feature vector (pixels)

Random

Level

encoding image (hypervector)
Training

encoding

sample hypervector $[10010001100\ldots]$  

class hypervector $[001111000011\ldots]$  

Repeat for all samples
Inference

Sample hypervector

Class hypervectors

\[ \delta \]

Prediction = Most similar

0 111100110011...
1 01001011101...
2 101011010010...

110010001100...

111100110011...

010010101101...

010101011010...
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

```c
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;

hyperspaces (rows are hypervectors)

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);
        }
    }
}

*majority rules
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 \( D = 10,000 \)

```python
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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Library</th>
<th>Parameters</th>
<th>Accuracy</th>
<th>µJ per Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>TFLM Float</td>
<td>1 hidden layer of 64 nodes</td>
<td>96%</td>
<td>20.6</td>
</tr>
<tr>
<td></td>
<td>TFLM Quant</td>
<td></td>
<td>96%</td>
<td>6.05</td>
</tr>
<tr>
<td></td>
<td>EmbHD</td>
<td>1 hidden layer of 7,000</td>
<td>80%</td>
<td>2036.68</td>
</tr>
<tr>
<td>ISOLET</td>
<td>TFLM Float</td>
<td>1 hidden layer of 128 nodes</td>
<td>95%</td>
<td>32.82</td>
</tr>
<tr>
<td></td>
<td>TFLM Quant</td>
<td></td>
<td>95%</td>
<td>11.17</td>
</tr>
<tr>
<td></td>
<td>EmbHD</td>
<td>1 hidden layer of 10,000</td>
<td>81%</td>
<td>1999.86</td>
</tr>
</tbody>
</table>
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