UC San Diego

MicroHD: An Accuracy-Driven Optimization of Hyperdimensional Computing algorithms for TinyML systems

Flavio Ponzina and Prof. Tajana Rosing
System Energy Efficiency Lab
Department of Computer Science and Engineering
University of California San Diego (UCSD)

Contact: fponzina@ucsd.edu

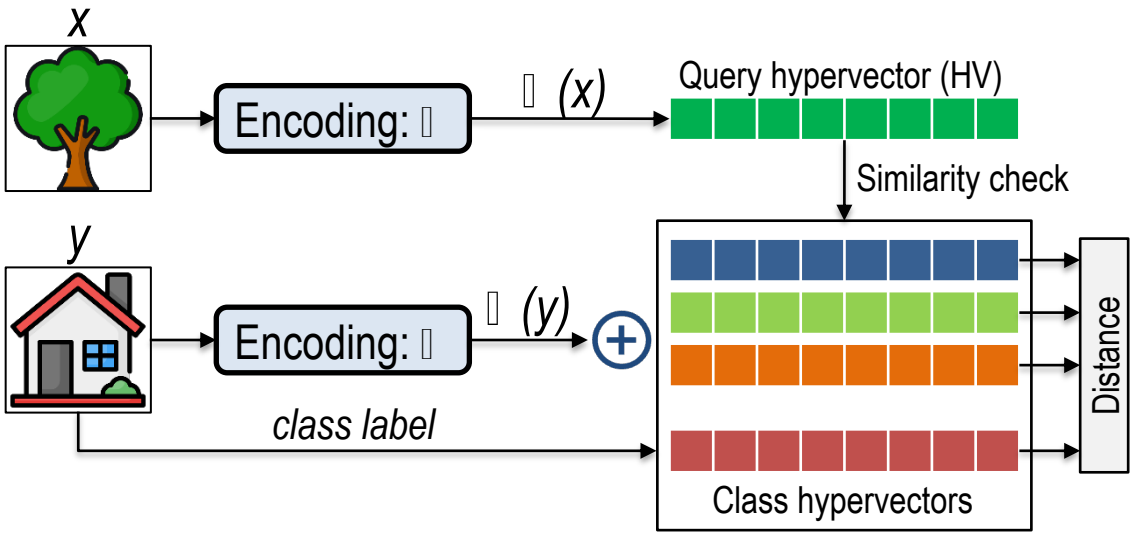
System Energy Efficiency Lab

seelab.ucsd.edu

Hyperdimensional Computing (HDC)

Brain-inspired computing approach

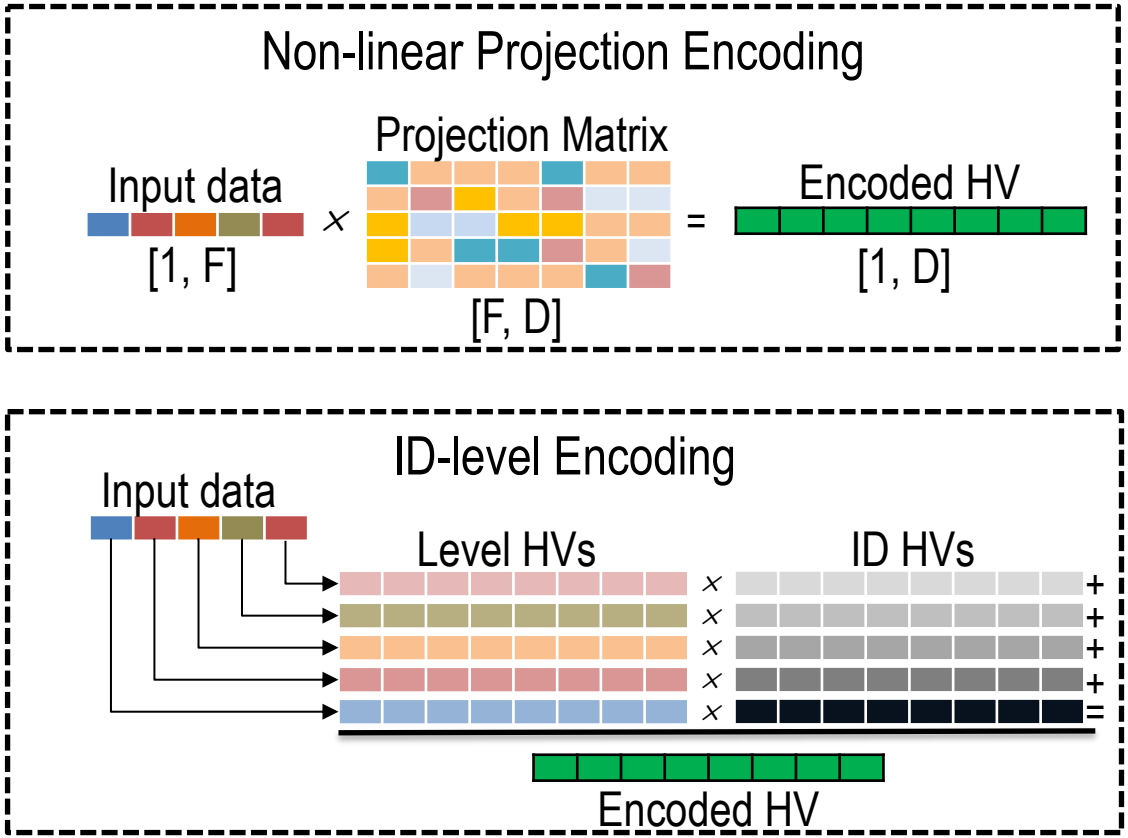
1. Distributed feature representation in a holistic high-dimensional space
2. Lightweight training and inference
3. Robustness against errors/noise
4. Well-suited for hardware acceleration



Applications

- Autonomous driving
- Voice recognition
- DNA sequencing
- Mass spectrometry

Encoding function: two examples



HDC Optimizations

Objective: Finetune *HDC hyperparameters* to reduce resource requirements

What are HDC hyperparameters?

Dimensionality, quantization bitwidth, sparsity, bandwidth, learning rate, …

Dimensionality
From 10k to 1k dimensions

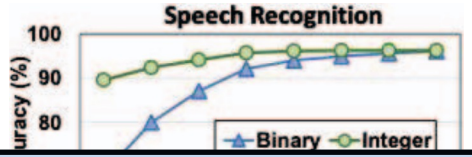
Bitwidth
From integer to binary

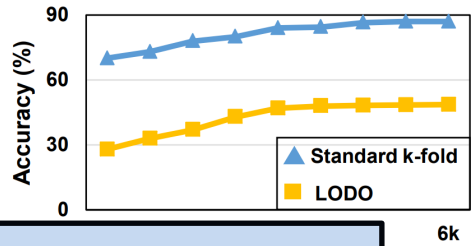
Encoding-based
Num. of level hypervectors

Related works and challenge

Previous works exploring <u>dimensionality reduction</u>

1. QuantHD [TCAD'19],
2. OnlineHD [DATE'21],
3. DOMINO [ICCAD'23]

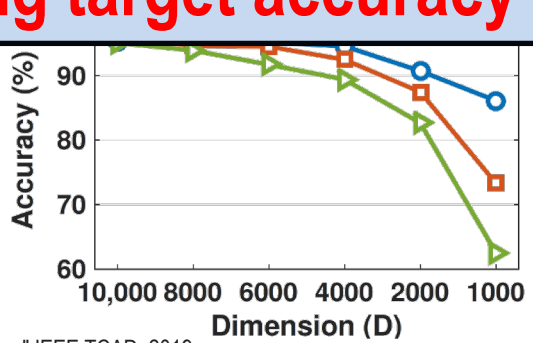


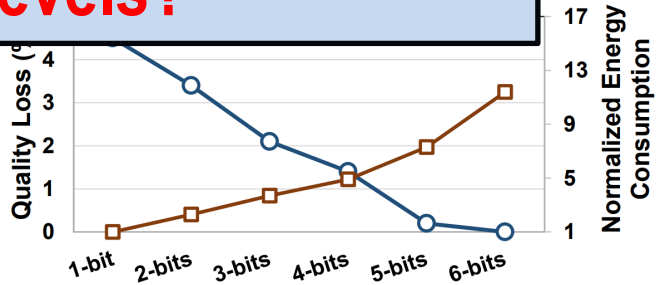


How to select proper HDC hyper-parameters while ensuring target accuracy levels?

P

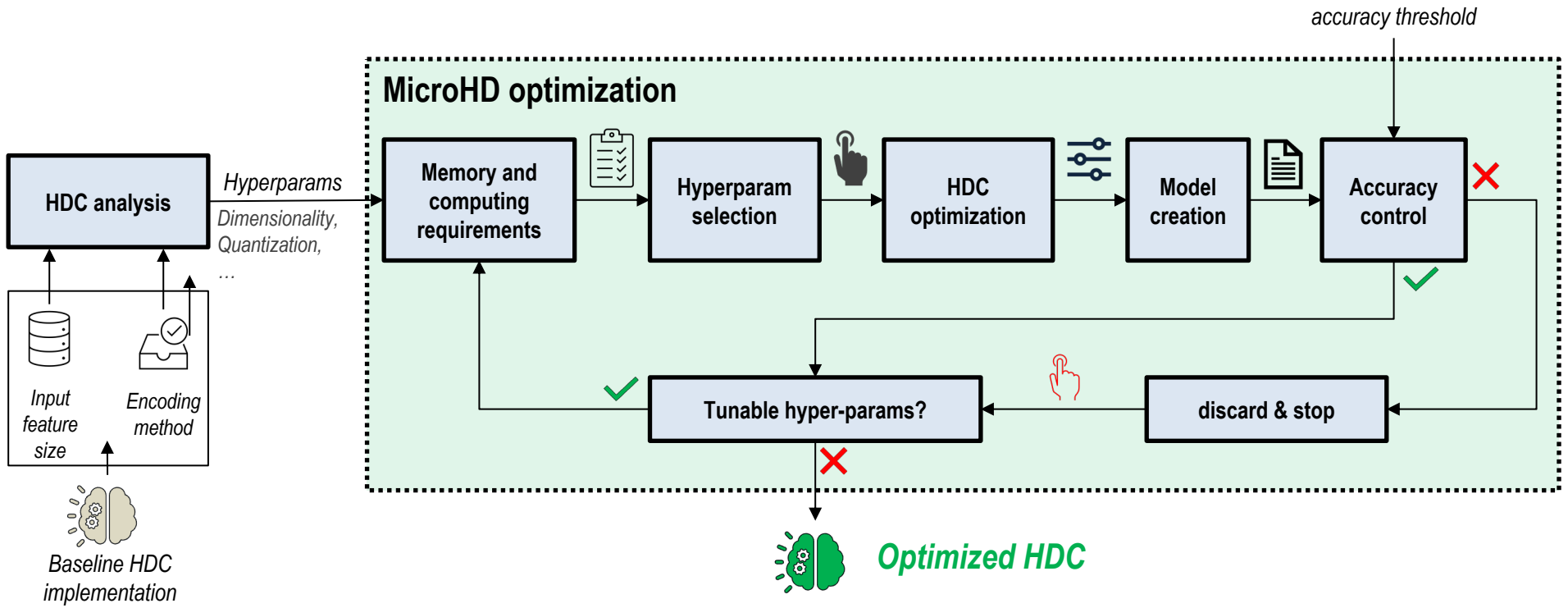
1. [illegible]
2. ManiHD [DATE'21],
3. QubitHD [arXiv'22]



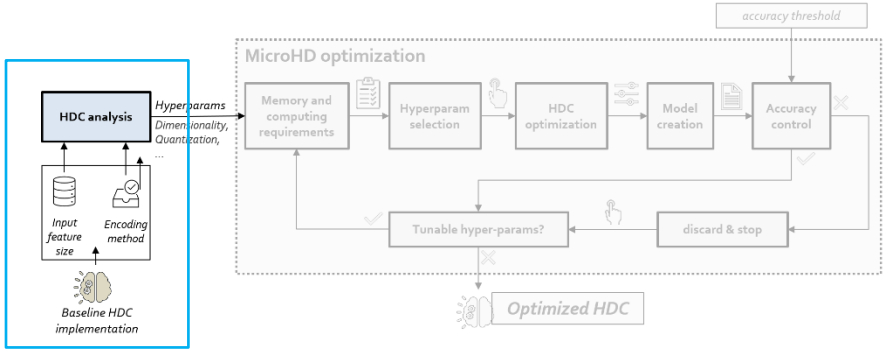


Imani et al. "Quanthd: A quantization framework for hyperdimensional computing." IEEE TCAD, 2019
Hernández-Cano et al. "Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system.", IEEE *DATE* 2021
Wang et al. "DOMINO: Domain-Invariant Hyperdimensional Classification for Multi-Sensor Time Series Data." IEEE ICCAD 2023
Zou et al. "Manihd: Efficient hyper-dimensional learning using manifold trainable encoder." IEEE DATE 2021
Bosch et al. "QubitHD: A stochastic acceleration method for HD computing-based machine learning." arXiv 2022

MicroHD: overview



MicroHD: step 1
HDC Analysis

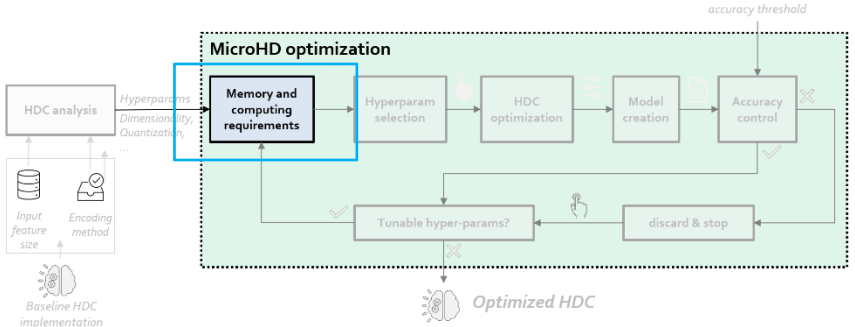


Baseline HDC model → **HDC analysis** → **List of adjustable hyperparameters**

Encoding method: ID-Level encoding
✓ Quantization scheme: integer (16-bit)
Input Features: 50
Output Classes: 10
✓ Hyperspace dimensionality: 10k
✓ #Levels: 1024
Accuracy: 95%

MicroHD: step 2

Resource requirements



How much memory does the application need?

Encoding

1. ID hypervectors (binary, size = # input features × dimensionality)
2. Level hypervectors (binary, size = # levels × dimensionality)

Model

1. Class hypervectors (bitwidth B, size = # output classes × dimensionality)

How many bitwise operations does the application need?

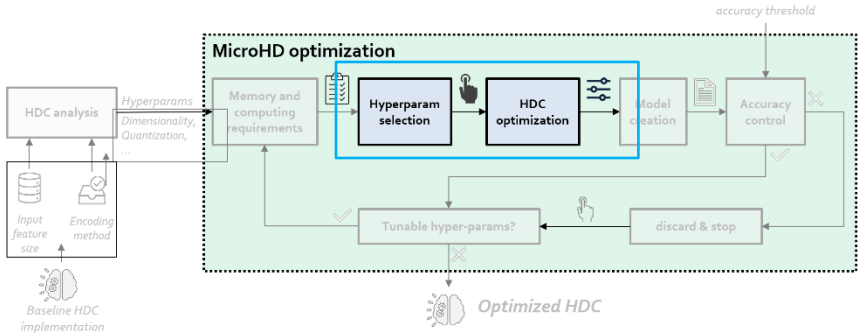
Encoding

1. # bindings = dimensionality
2. # bundlings = dimensionality

Similarity check

1. # output classes × dimensionality dot-products (elements having bitwidth B)

MicroHD: step 3
Optimization phase



Lists of candidate hyperparameter values

Dimensionality: [200, 500, 1k, 2k, 3k, 4k, 5k, 6k, 7k, 8k, 9k, **10k**]
Levels: [2, 4, 8, 16, 32, 64, 128, 256, 512, **1024**]
Quantization: [1, 2, 3, 4, 5, 6, 7 ,8, 9, 10, 11, 12, 13, 14, 15, **16**]

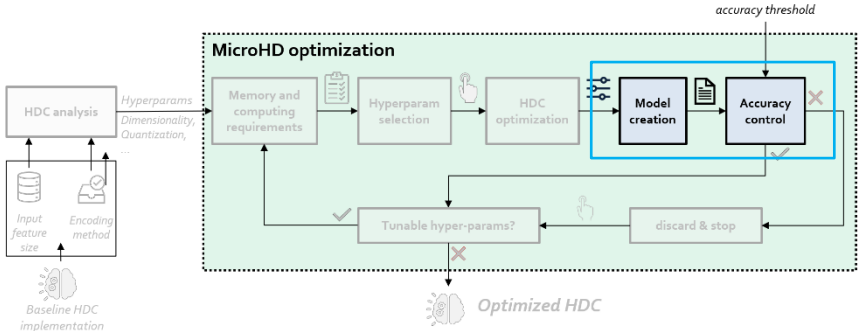
Binary design space exploration

[200, 500, 1k, 2k, 3k, 4k, 5k, 6k, 7k, 8k, 9k, **10k**]

Co-optimization approach

Consider all hyperparameters. Which one leads to larger savings if optimized?
Greedy approach

MicroHD: step 4
Model evaluation



A new model is defined and trained

Training

1. One-pass training
2. Finetuning based on *OnlineHD* [DATE'21]

$$\vec{C}_l \leftarrow \vec{C}_l + \eta\ (1 - \delta_l) \times \vec{\mathcal{H}}$$

$$\vec{C}_{l'} \leftarrow \vec{C}_{l'} - \eta\ (1 - \delta_{l'}) \times \vec{\mathcal{H}}$$

learning rate

input hypervector

Class hypervectors update

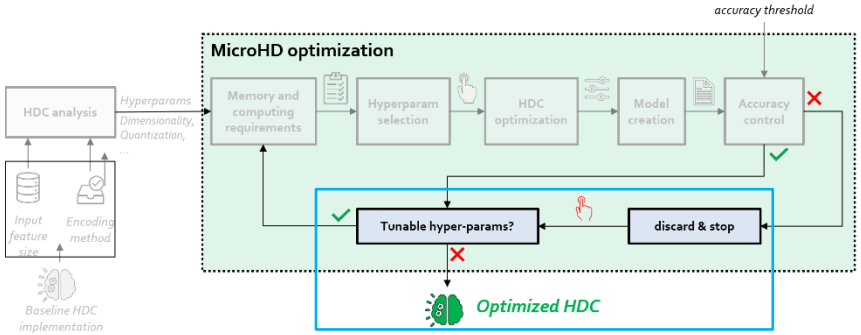
cosine distance

Accuracy evaluations

Is accuracy acceptable?
(higher than the 90% constraint, in this example)

Hernández-Cano et al. "Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system.", IEEE *DATE* 2021

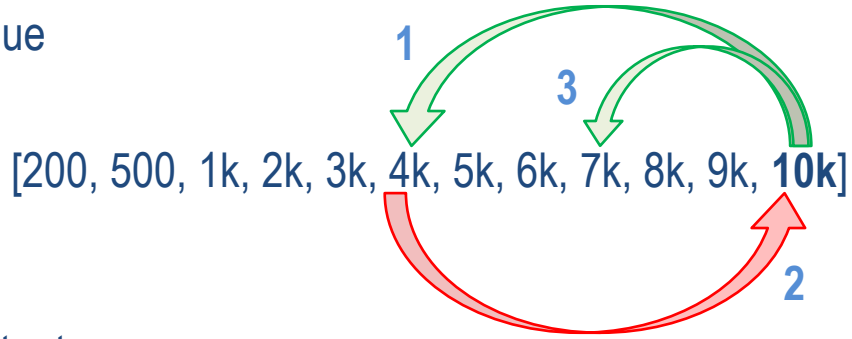
MicroHD: step 5
Loop and termination



What if accuracy is too low?

Discard the last optimization step

1. Return to the previous hyperparameter value
2. Halve the step size for the binary search



When does the loop terminate?

No further hyperparameter values to explore
The last HDC configuration is the produced output

Experimental Setup

MicroHD implementation

Python-based

1. PyTorch for data loader and vector manipulation
2. TorchHD for HDC functionalities

Benchmarking

Baselines: 10k-HDC, QuantHD [TCAD'19], OnlineHD [DATE'21], Basaklar et al. [tinyML'21] , Zeulin et al., [MLSys'23]

Datasets: ISOLET, UCIHAR, MNIST, FMNIST, PAMAP, Connect-4

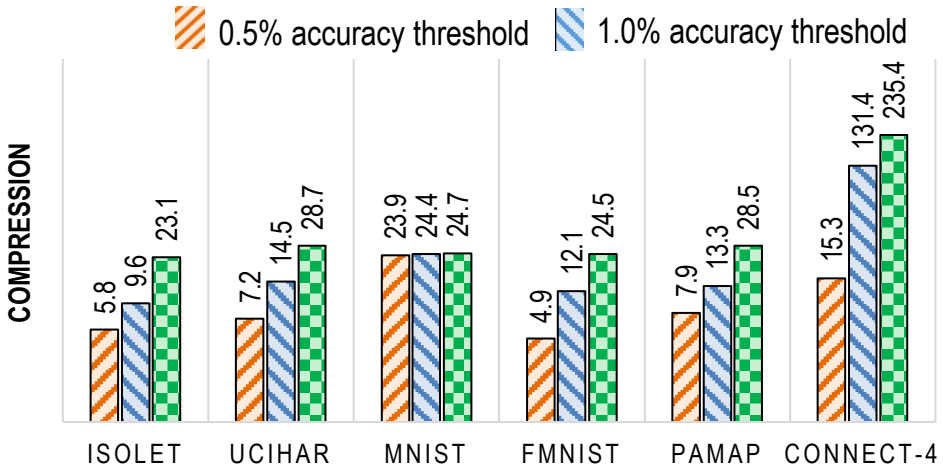
Accuracy thresholds: 0.5%, 1.0%, 5.0%

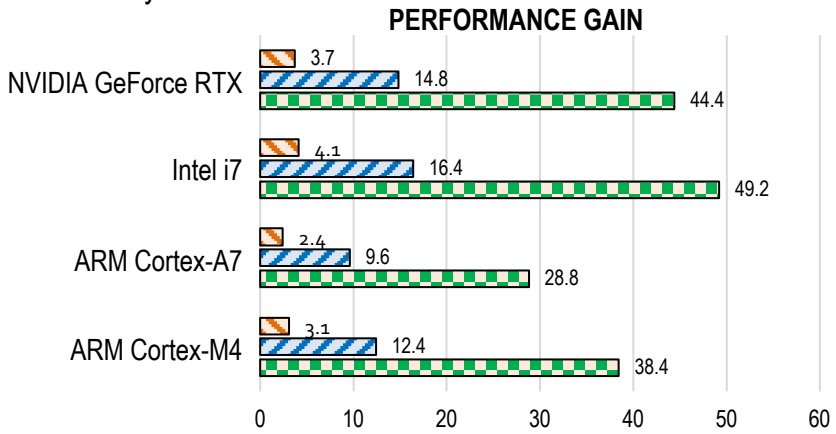
Performance evaluation

Multiple processing elements: Nvidia GeForce RTX, Intel i7, ARM Cortex-A7, ARM Cortex-M4

Imani et al. "Quanthd: A quantization framework for hyperdimensional computing." IEEE TCAD, 2019
Hernández-Cano et al. "Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system.", IEEE *DATE* 2021
Basaklar et al. "Hypervector design for efficient hyperdimensional computing on edge devices." tinyML'21
Zeulin et al. "Resource-Efficient Federated Hyperdimensional Computing." MLSys'23

Experimental Results





Average compressions of:

10.8× *(<0.5% accuracy drop)*

34.2× *(<1.0% accuracy drop)*

60.8× *(<5.0% accuracy drop)*

Average performance gains of:

3.3× *(<0.5% accuracy drop)*

13.3× *(<1.0% accuracy drop)*

40.2× *(<5.0% accuracy drop)*

Experimental Results

Comparison with previous works

Previous works DO NOT ensure desired accu

1. Consider common benchmarks
2. Take the best point in previous works' Pareto curve
3. Run MicroHD imposing equivalent accuracy thresholds

MicroHD takes 2h on GPU
Exhaustive explorations would take years

Compression improvements

QuantHD [TCAD'19]	**+7.5×**
OnlineHD [DATE'21]	**+1.9×**
Basaklar et al. [tinyML'21]	**+1.3×**
Zeulin et al., [MLSys'23]	**+6.4×**

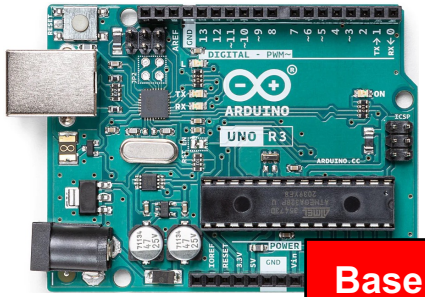
Higher compression while also ensuring target accuracy levels. How?

Co-optimization of HDC hyperparameters

MicroHD for tinyML: a practical implementation

HDC classification using A

Inference performance: 75*ms* / image
Accuracy degradation: <5%

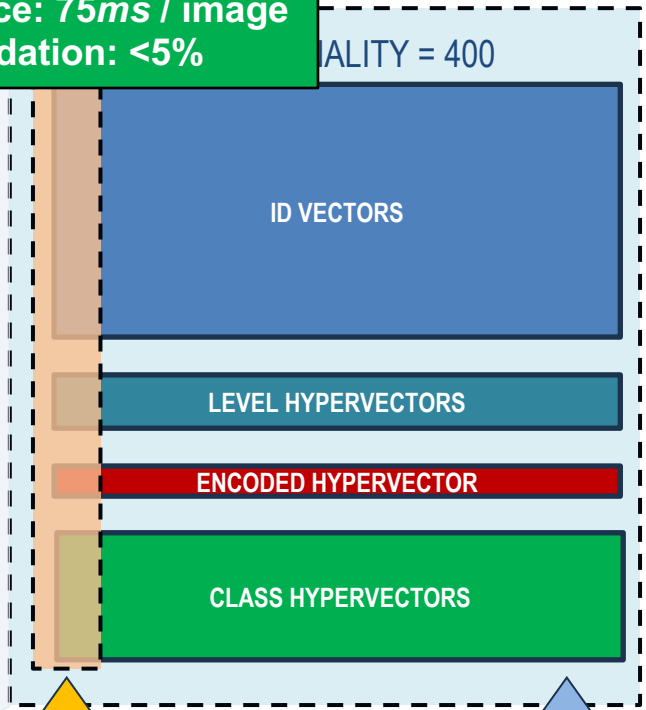


Arduino UNO
1. SRAM: 2 KB
2. Flash: 32 KB

Baseline HDC requires 2.5MB!

MNIST image
28x28px

MicroHD



ID Hypervectors
784

Level
Hypervectors
4

Class
Hypervectors
10

Load in SRAM

Store in Flash

Conclusion and future works

MicroHD optimization methodology

1. First accuracy-driven optimization for HDC workloads
2. Efficient exploration of the design space
3. Co-optimization of HDC hyperparameters
4. When compared to the state-of-the-art:
    - Compression gains up to **7×**
    - Performance gains up to **6×**
      (for <1% accuracy degradation)

Future works

1. HW-SW Co-design optimization for processing in-memory (PIM) HDC
2. Extending MicroHD including memory and performance constraints

Thank you

Flavio Ponzina and Prof. Tajana Rosing
System Energy Efficiency Lab
Department of Computer Science and Engineering
University of California San Diego (UCSD)

Contact: fponzina@ucsd.edu

System Energy Efficiency Lab

seelab.ucsd.edu

Copyright Notice

www.tinyml.org