Synergy of Hyperdimensional Computing and Neural Network for Efficient Intelligence on Edge

Qualification Exam

Fatemeh Asgarinejad

Committee

Co Chairs: Prof. Tajana Rosing, Prof. Baris Aksanli

Prof. Sahar Baghdadchi

Prof. Ryan Kastner

Prof. Junfei Xie

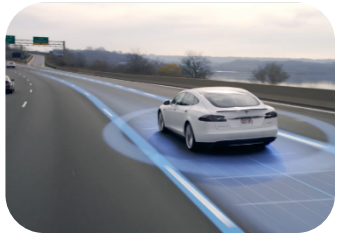
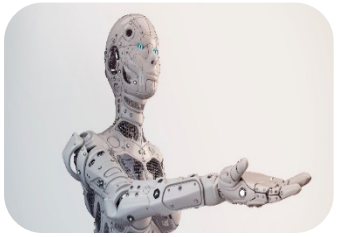
System Energy Efficiency Lab

SAN DIEGO STATE UNIVERSITY

College of Engineering

Machine Learning Challenges

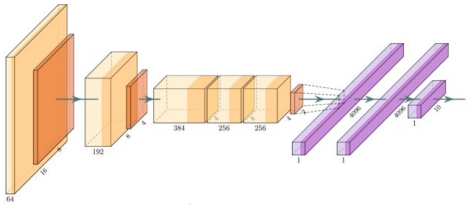
Machine learning is becoming ubiquitous!

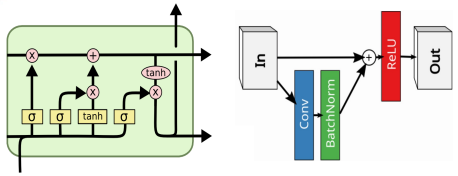


But it comes with challenges

- Large models
  - Millions of parameters, billions of operations
- Complex to understand and tune
  - And getting worse with raise of heterogeneous data
- Needs a lot of data and iterations to train
- Highly vulnerable to error and noise

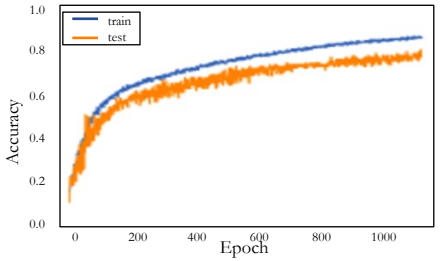


large models

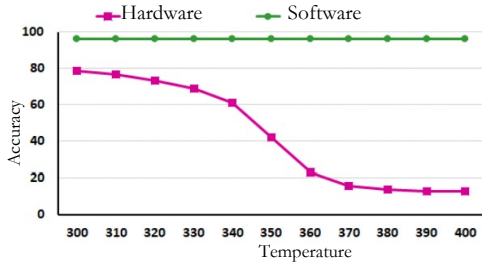


complex to understand

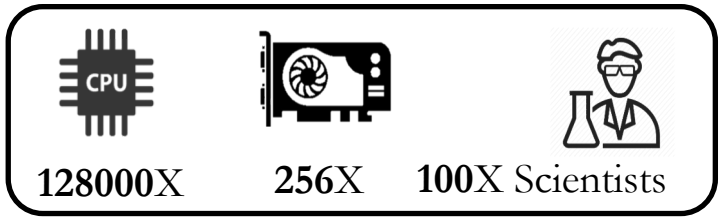
needs multiple epochs

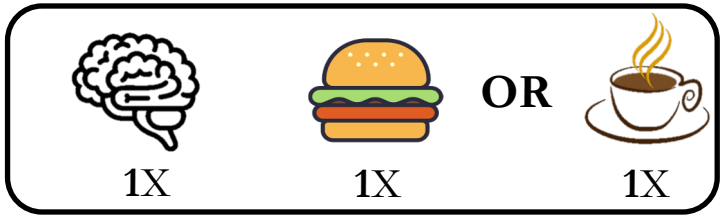


susceptible to hardware noise



Hyperdimensional Computing (HDC)





- Similar to brain, HDC is based on neural representation of data

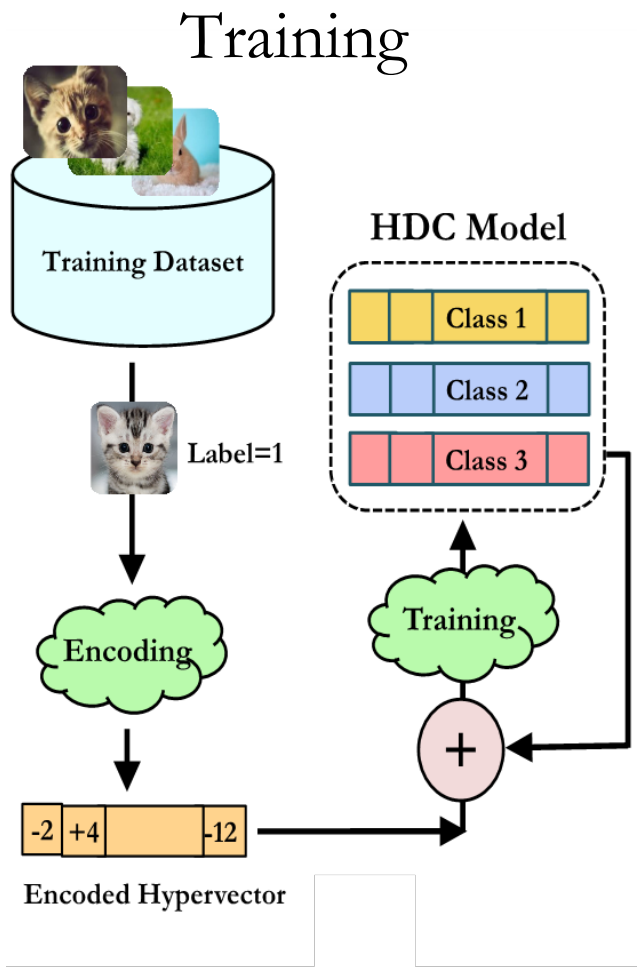
Lightweight: Smaller model, simple computation

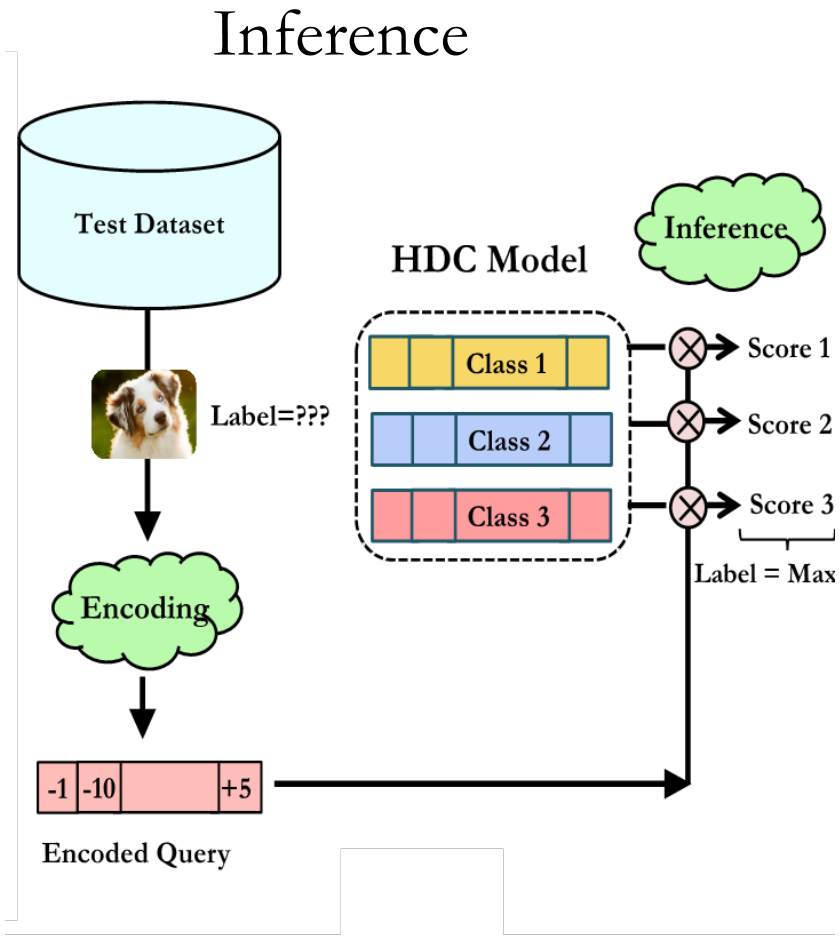
Simple: Easy to understand and calibrate for different data types

Energy efficient: Low power consumption, fewer epochs for training convergence

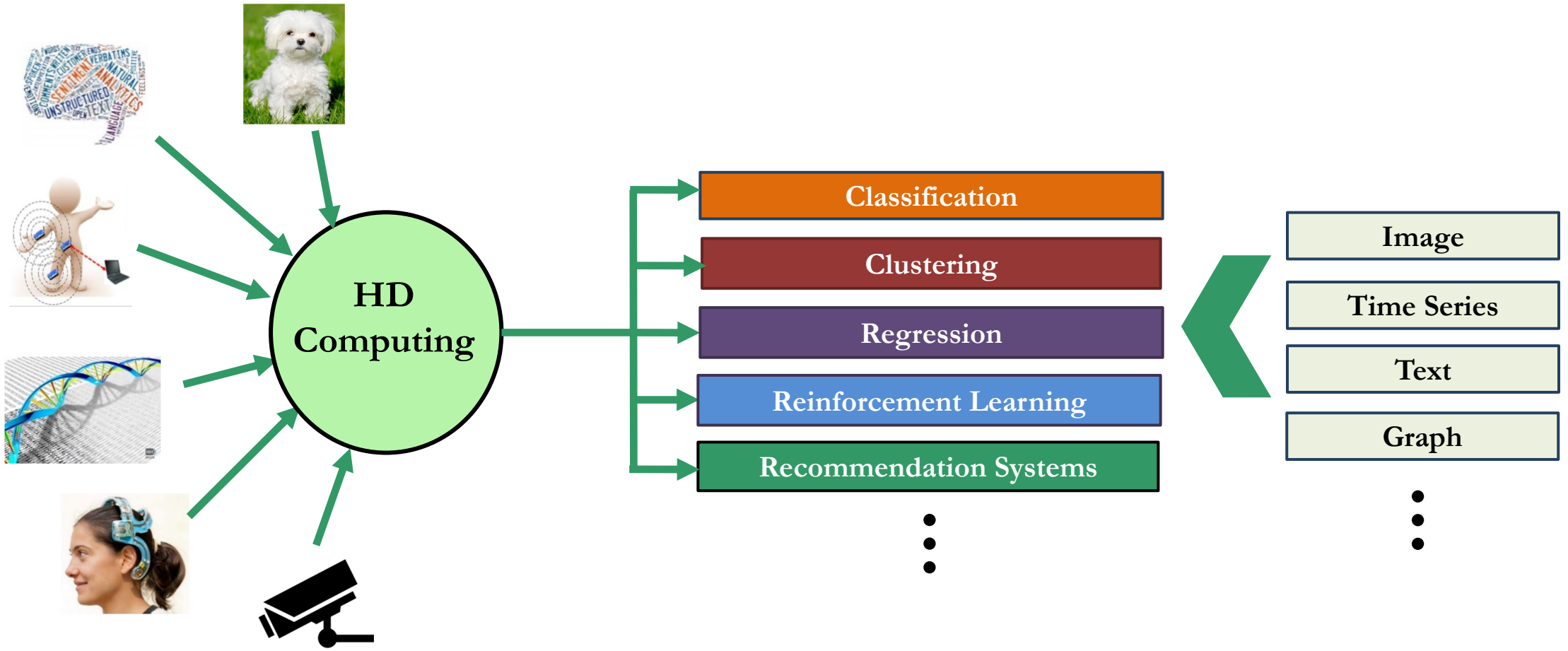
Robust: High noise/error tolerance; can leverage unreliable hardware & communication

Hyperdimensional Computing in a Nutshell



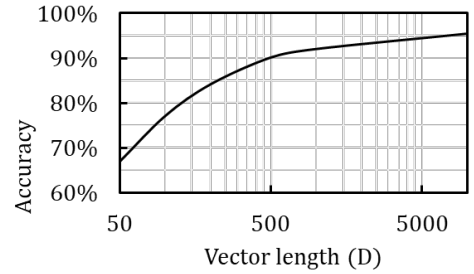


HDC Applications

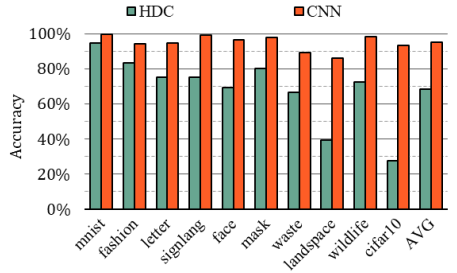


HDC Challenges

- Accuracy-efficiency trade-off [Duan, DAC'22; Cano, DATE'21]
  - Computation and memory scales proportionally



- Low accuracy on image data [Duan, NanoArch'21; Imani, DAC'22]
  - 26% lower than a shallow CNN

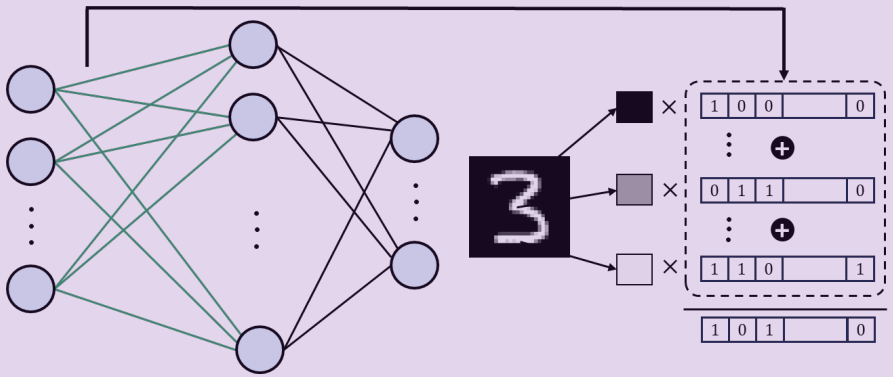


- Low privacy and security [Khaleghi, DAC'20; Cano, DAC'21]
  - Data can be reconstructed from hypervectors
  - HDC model can be fooled with small perturbation in input

- Low efficiency in integrating CNN feature extraction with HDC [Chandrasekaran, DAC'22, Dutta, GLSVLSI'22]
  - Overall system efficiency will be governed by CNN component

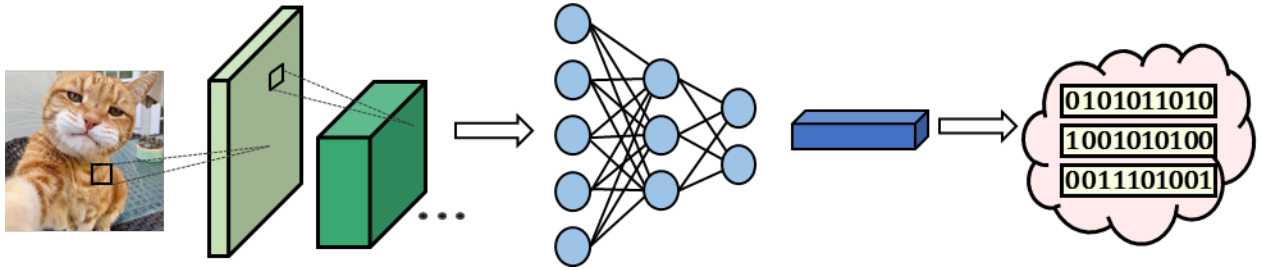
Contribution

Synergy of HDC and NN for efficient and private intelligence on edge



Efficiency **Accuracy**

ASP-DAC 2024



Images **Privacy**

DAC 2024 (WIP)

ISLPED 2024 (submitted)

Previous Work

Accuracy-Efficiency Trade-off

Novel Encoding & Training

- Non-linear encoding [Imani, Micro'20]
- Isometric mapping [Zou, DATE'21]
- Weighted updates [Cano, DATE'21]
- Window-based encoding [Khaleghi, DAC'22; Imani, HPCA'21]

- More complex (e.g., floating point) operations
- Higher number of operations
- Low accuracy under quantization and dimensionality reduction

Previous Work

Accuracy-Efficiency Trade-off

Novel Encoding & Training

Classical Feature Extraction

- Stochastic HDC-like arithmetic [Poduval, DAC'21; Imani, DAC'22]
- Histogram of Gradients (HOG) features [Duan, NanoArch'2021; Imani, DAC'22]
- Feature extraction bottleneck
- Low accuracy under quantization and dimensionality reduction
- Subpar performance compared to neural networks

Previous Work

Accuracy-Efficiency Trade-off

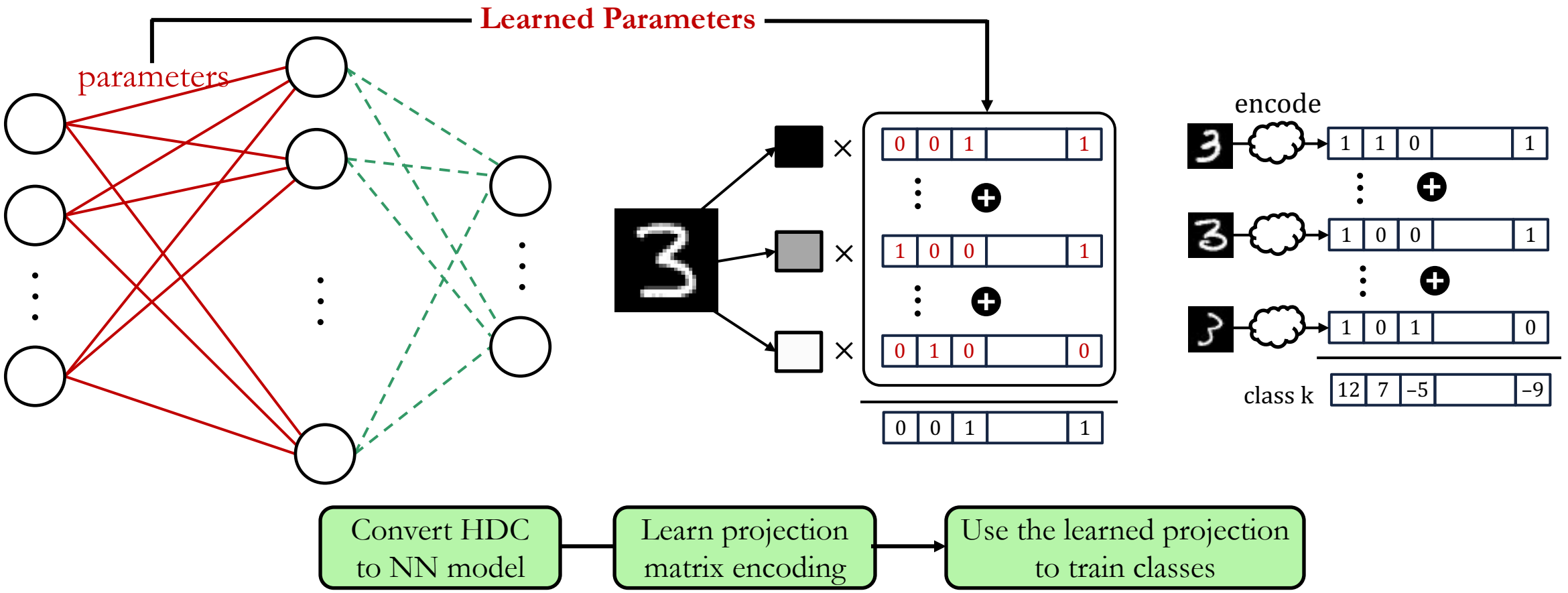
Novel Encoding & Training

Classical Feature Extraction

Leveraging Neural Networks

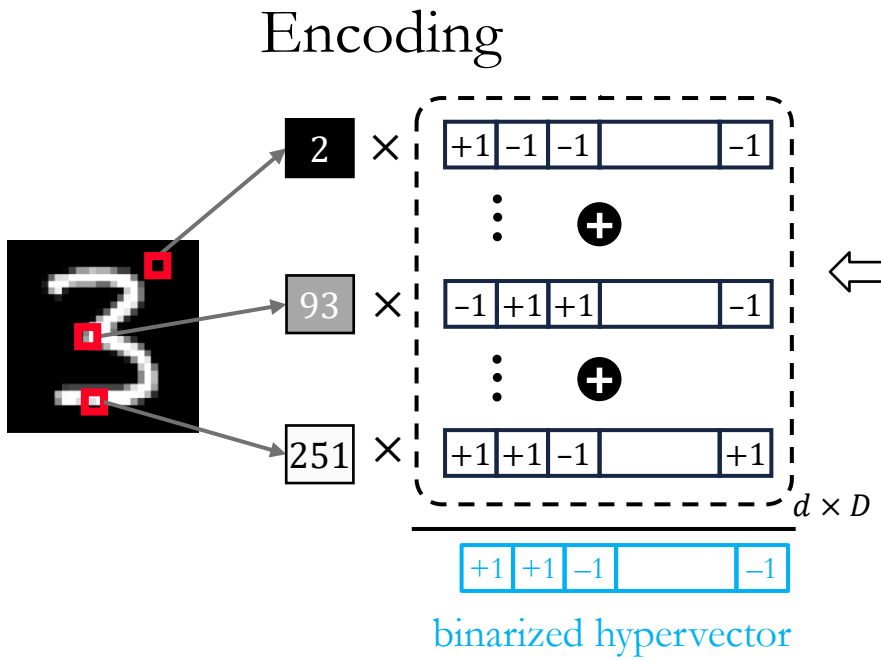
- Model HDC as NN to learn classes [Duan, DAC'22; Mao, arXiv'22]
- Complex training (backpropagation)
- Not suitable for in-field learning

Proposed Method: Learned Projection



Projection-Based Encoding

- Projection-Based encoding can be mapped as a vector-matrix multiplication

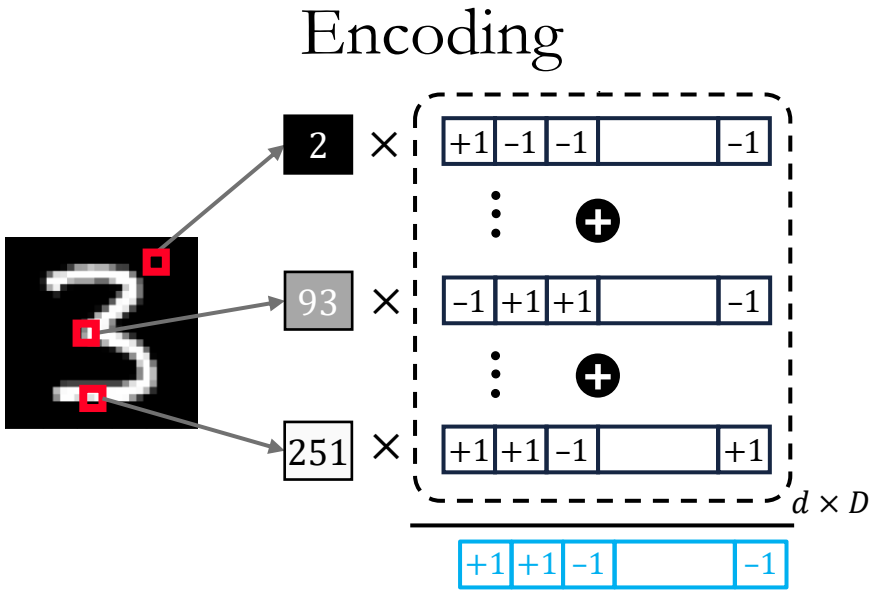


$$\Leftrightarrow [f_1 \quad f_2 \quad f_3 \quad \dots \quad f_d]_{1\times d} \times \begin{bmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,D} \\ B_{2,1} & B_{2,2} & \cdots & B_{2,D} \\ B_{3,1} & B_{3,2} & \cdots & B_{3,D} \\ \vdots & \vdots & \vdots & \vdots \\ B_{\mathrm{d},1} & B_{\mathrm{d},2} & \cdots & B_{\mathrm{d},D} \end{bmatrix}_{\boldsymbol{d\times D}} = [h_1 \quad h_2 \quad h_3 \quad \dots \quad h_D]_{1\times D}$$

Every feature gets a random initialization ($B_{i,j} \in \{1, -1\}$)

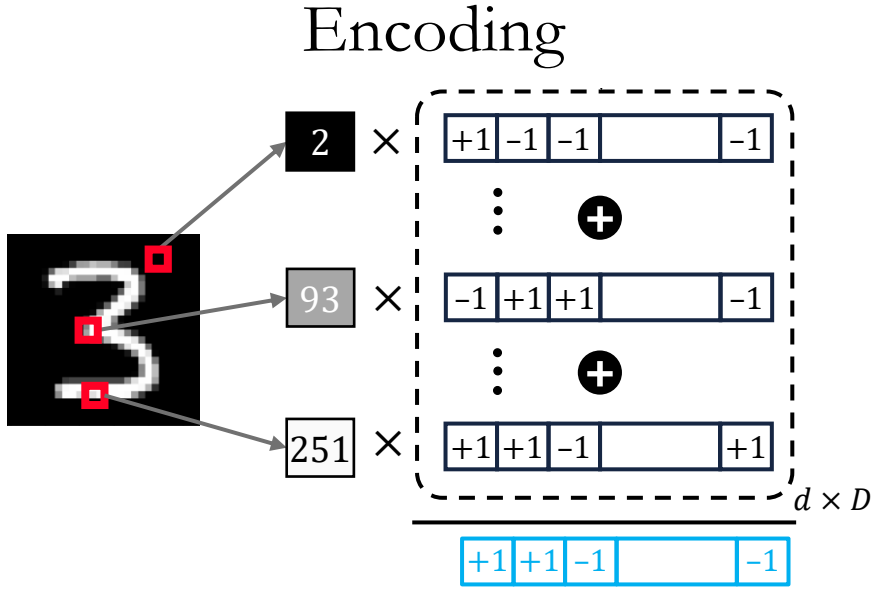
$$h_i = [f_1 \quad f_2 \quad f_3 \quad \dots \quad f_d]_{1\times d} \times \begin{bmatrix} B_{1,i} \\ B_{2,i} \\ \vdots \\ B_{\mathrm{d},i} \end{bmatrix}_{d\times 1} = f_1 \cdot B_{1,i} + f_2 \cdot B_{2,i} \dots + f_d \cdot B_{d\times i}$$

Proposed Method: Learning the Projection Matrix

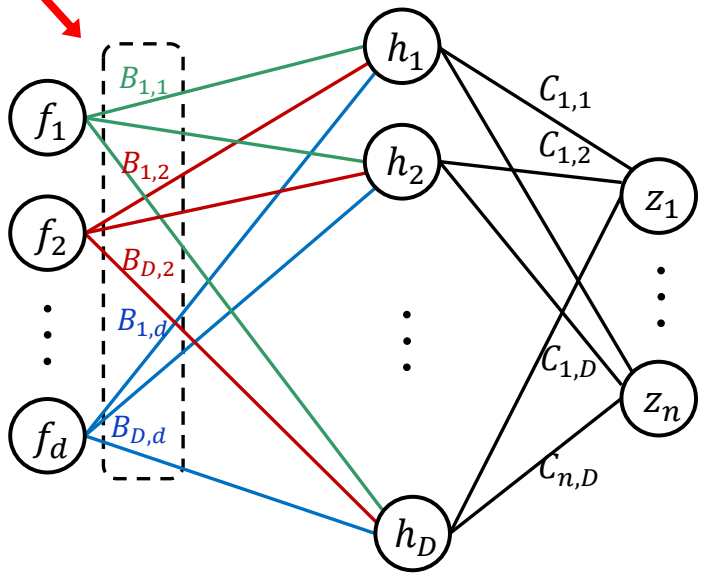


$$[f_1 \quad f_2 \quad f_3 \quad \dots \quad f_d]_{1\times d} \times \begin{bmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,D} \\ B_{2,1} & B_{2,2} & \cdots & B_{2,D} \\ B_{3,1} & B_{3,2} & \cdots & B_{3,D} \\ \vdots & \vdots & \vdots & \vdots \\ B_{d,1} & B_{d,2} & \cdots & B_{D,d} \end{bmatrix} = [h_1 \quad h_2 \quad h_3 \quad \dots \quad h_D]_{1\times D}$$

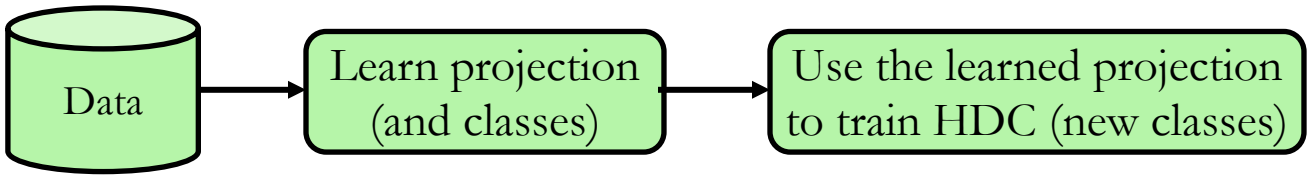
PIONEER: Learning the Projection Matrix



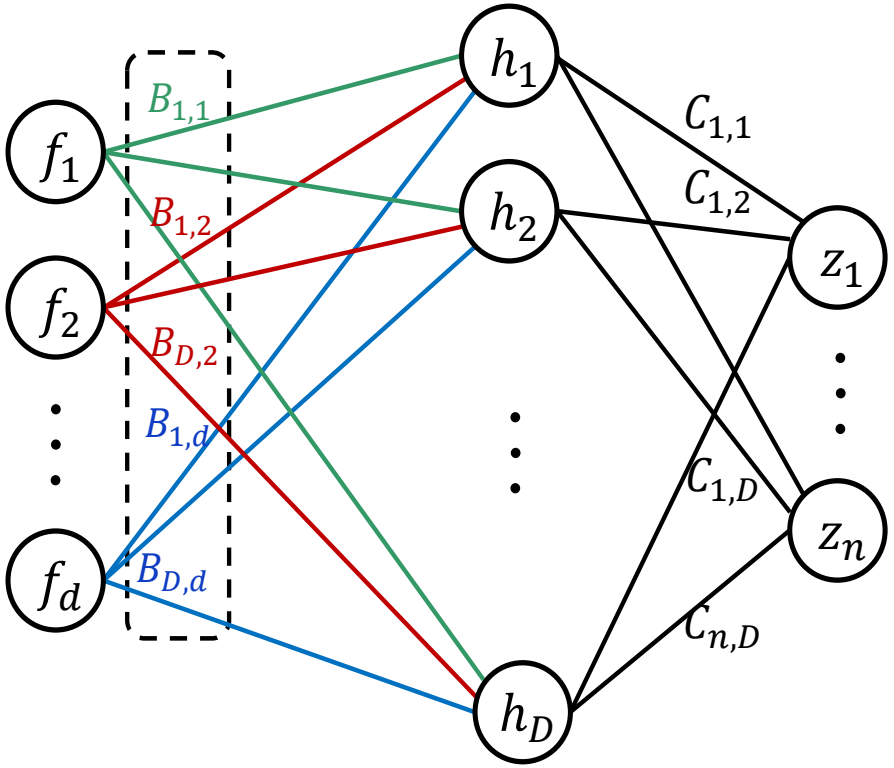
$$[f_1 \quad f_2 \quad f_3 \quad \dots \quad f_d]_{1\times d} \times \begin{bmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,D} \\ B_{2,1} & B_{2,2} & \cdots & B_{2,D} \\ B_{3,1} & B_{3,2} & \cdots & B_{3,D} \\ \vdots & \vdots & \vdots & \vdots \\ B_{d,1} & B_{d,2} & \cdots & B_{d,D} \end{bmatrix} = [h_1 \quad h_2 \quad h_3 \quad \dots \quad h_D]_{1\times D}$$



- We adopt a NN-based mechanism to learn the projection



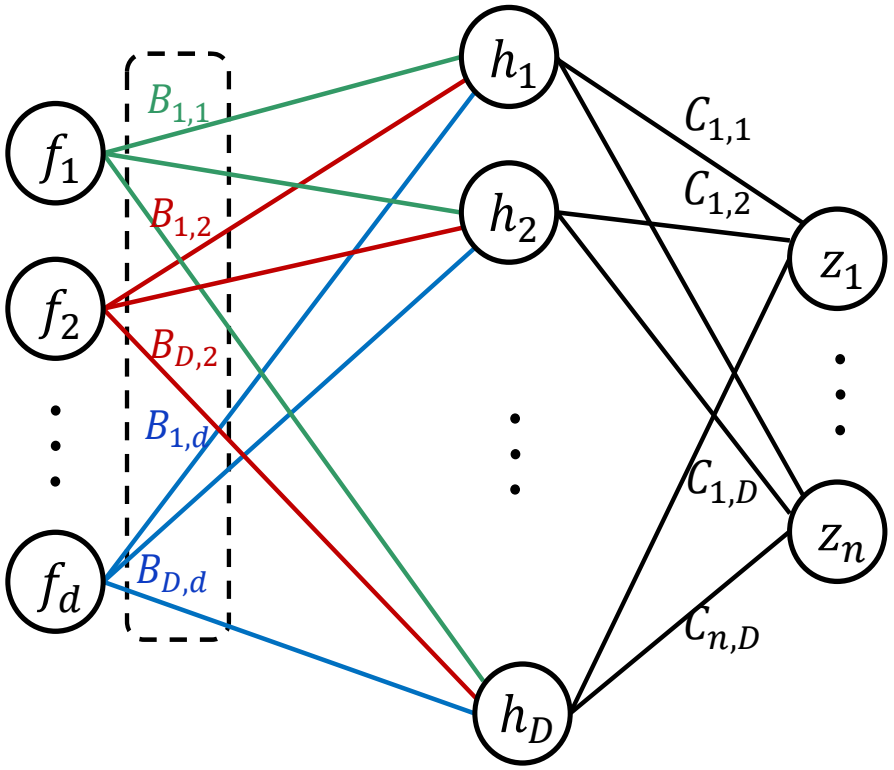
Learning the Projection Matrix



$$h_i = B_{i,1}.f_1 + B_{i,2}.f_2 + \cdots + B_{i,n}.f_n = B_i.\mathcal{F}$$

$$z_j = h_1.C_{j,1} + h_2.C_{j,2} + \ldots + h_D.C_{j,D} = H.C_j$$

Learning the Projection Matrix Mathematics



(1) $L = -\sum_{i=1}^{n} y_i \log \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}} = -\sum_{i}^{n} y_i \log s_i$ loss function

(2) $C_{1,1}^{(t+1)} = C_{1,1}^{(t)} - \lambda \frac{\partial L}{\partial C_{1,1}}$ (3) $\frac{\partial L}{\partial C_{1,1}} = \frac{\partial L}{\partial z_1} \frac{\partial z_1}{\partial C_{1,1}}$

(4) $\frac{\partial L}{\partial z_i} = s_i - y_i$ (5) $\frac{\partial z_1}{\partial C_{1,1}} = \frac{\partial (h_1 . C_{1,1} + \cdots + h_D . C_{1,D})}{\partial C_{1,1}} = h_1$

(6) $\rightarrow \frac{\partial L}{\partial C_{1,1}} = (s_1 - y_1) h_1 = \left(\frac{e^{z_1}}{\sum_{k=1}^{n} e^{z_k}} - y_1 \right) h_1 = \alpha_1 h_1$

When label is, e.g., C1 ($y = [1,0,\cdots,0]$):

$$y_1 = 1 \rightarrow \quad C_{1,i}^{(t+1)} = C_{1,i}^{(t)} + \lambda(1 - \alpha_1) h_i = C_{1,i}^{(t)} + \lambda \alpha_2 h_i$$

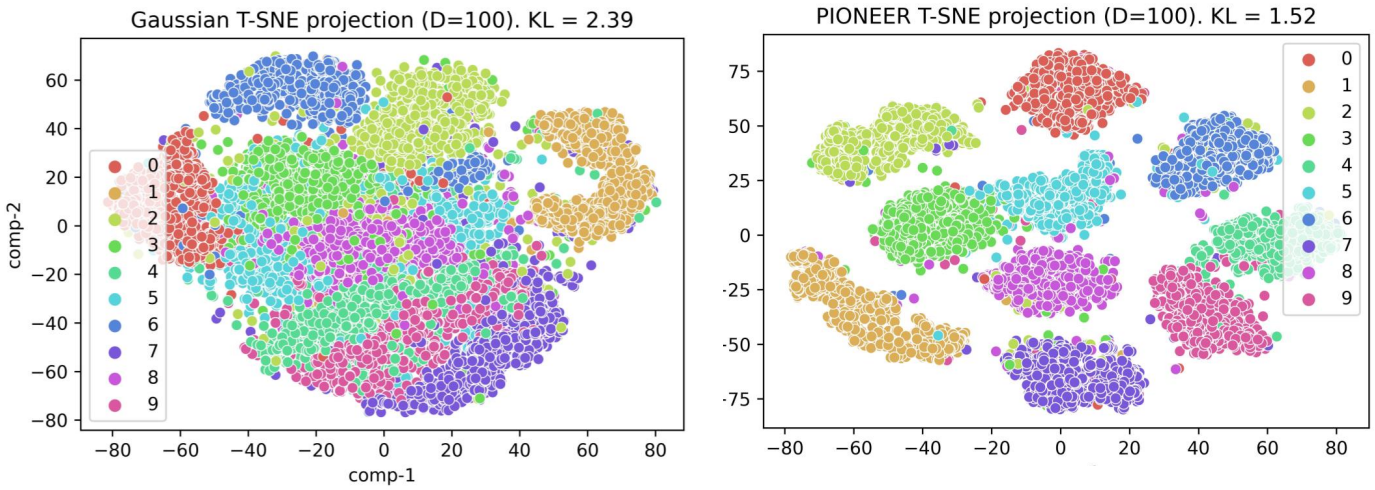
When label is not C1 ($y = [0,\cdots]$):

$$y_1 = 0 \rightarrow \quad C_{1,i}^{(t+1)} = C_{1,i}^{(t)} - \lambda \alpha_3 h_i$$

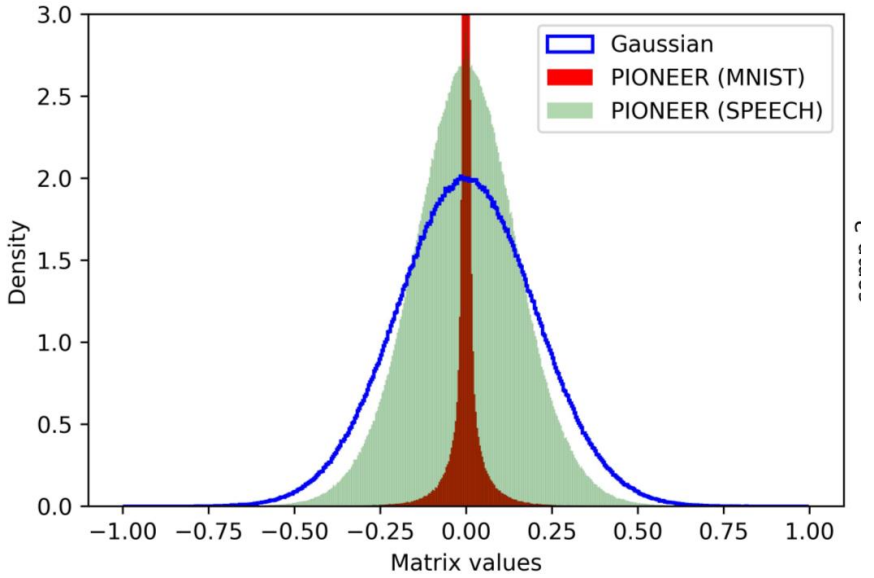
Update in class hypervectors in above NN aligns with HDC class updates

$$C^T = C^T + H$$
$$C^F = C^F - H$$

Discriminative Quality of PIONEER



t-SNE plots of MNIST (D = 100)



Distribution of the learned matrix elements

- PIONEER learns to distinguish classes better
- PIONEER calibrates the projection matrix based on data characteristics

Matrix Compression: Post-Training Quantization

$$\begin{bmatrix} 0.53 & -0.82 & -0.04 & 0.12 & -0.44 & -0.77 & 0.22 & 0.85 \\ -0.90 & -0.15 & -0.62 & 0.48 & -0.43 & -0.25 & -0.17 & -0.58 \\ 0.17 & 0.99 & 0.97 & 0.36 & -0.22 & 0.54 & 0.75 & 0.21 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.86 & -0.58 & 0.50 & 0.52 & -0.33 & -0.71 & 0.58 & 0.06 \end{bmatrix}_{D \times d}$$

Learned matrix (floating point)

- The learned matrix is not binary → large size; high-bit width operations
- Utilize HDC's tolerance to approximation to improve memory and computation

$$\begin{bmatrix} 0.5 & -0.875 & 0 & 0.125 & -0.5 & -0.75 & 0.25 & 0.875 \\ -0.875 & -0.125 & -0.625 & 0.5 & -0.375 & -0.25 & -0.125 & -0.625 \\ 0.125 & 1.0 & 1.0 & 0.375 & -0.25 & 0.5 & 0.75 & 0.25 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.875 & -0.625 & 0.5 & 0.5 & -0.375 & -0.75 & 0.625 & 0 \end{bmatrix}$$

INT4 (0000, …., 1111)

$$\begin{bmatrix} +1 & -1 & -1 & +1 & -1 & -1 & +1 & +1 \\ -1 & -1 & -1 & +1 & -1 & -1 & -1 & -1 \\ +1 & +1 & +1 & +1 & -1 & +1 & +1 & +1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 \end{bmatrix}$$

INT1 (binary)

Matrix Compression: Sparsification

$$\begin{bmatrix} 0.53 & -0.82 & -0.04 & 0.12 & -0.44 & -0.77 & 0.22 & 0.85 \\ -0.90 & -0.15 & -0.62 & 0.48 & -0.43 & -0.25 & -0.17 & -0.58 \\ 0.17 & 0.99 & 0.97 & 0.36 & -0.22 & 0.54 & 0.75 & 0.21 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.86 & -0.58 & 0.50 & 0.52 & -0.33 & -0.71 & 0.58 & 0.06 \end{bmatrix}_{D\times d} \Rightarrow \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & -1 & 0 & +1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & +1 & 0 & 0 & 0 & +1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Learned matrix

Sparse binary matrix

- Matrix-level sparsification is not hardware-efficient
  - Large overhead of storing non-zero indexes information

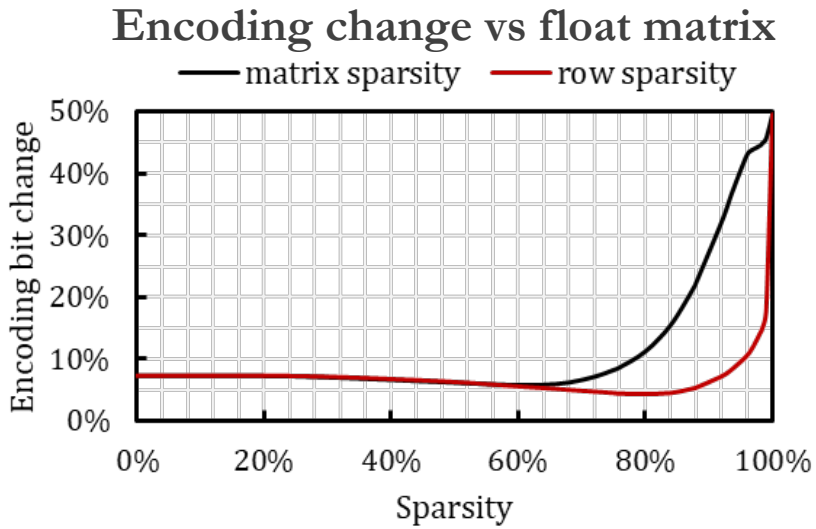
- Select K largest magnitude of each row
  - Minimal discrepancy to encoding output
  - No need to store row information

$$\begin{bmatrix} 0.53 & -0.82 & -0.04 & 0.12 & -0.44 & -0.77 & 0.22 & 0.85 \\ -0.90 & -0.15 & -0.62 & 0.48 & -0.43 & -0.25 & -0.17 & -0.58 \\ 0.17 & 0.99 & 0.97 & 0.36 & -0.22 & 0.54 & 0.75 & 0.21 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.86 & -0.58 & 0.50 & 0.52 & -0.33 & -0.71 & 0.58 & 0.06 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 & 0 & +1 \\ -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & +1 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ +1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

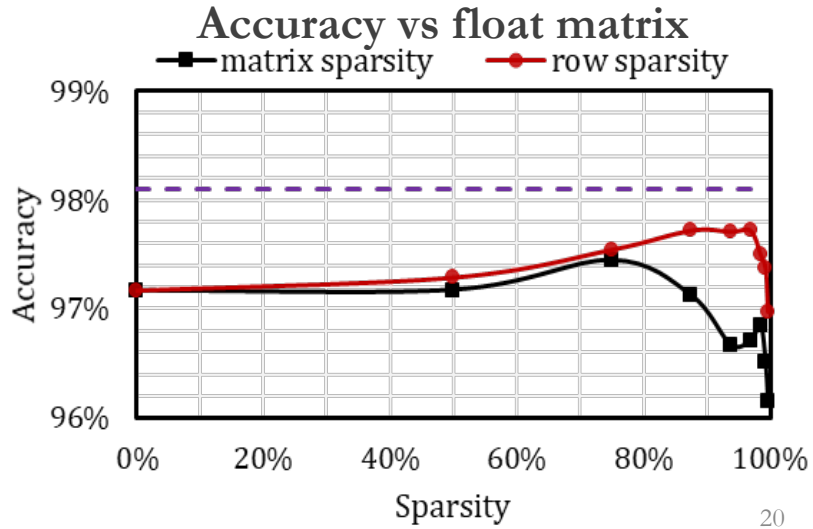
Row-sparse matrix

Row-Sparse Compression

- Lower encoding bit-change compared to matrix-level
  - i.e., keeps the mapping quality of learned projection



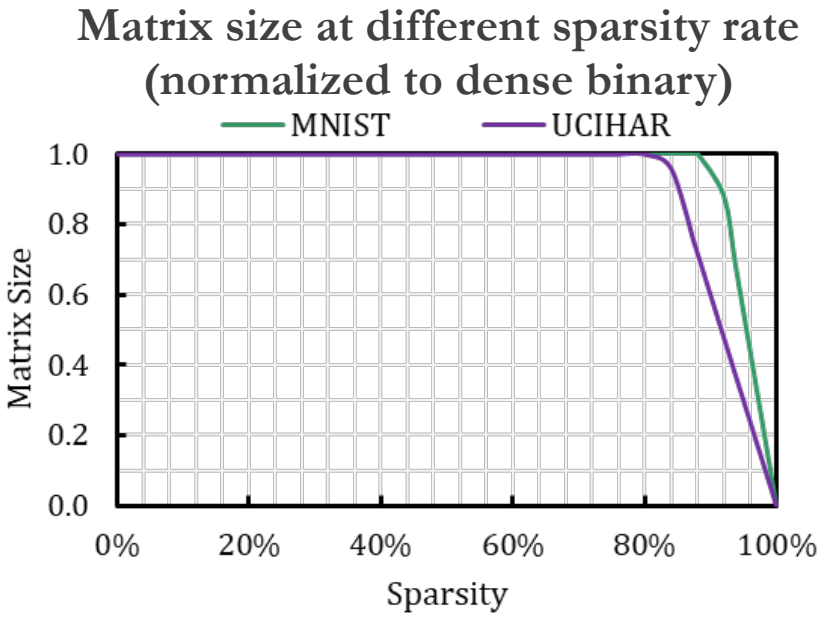
- Better accuracy versus matrix sparsity



Row-Sparse Compression

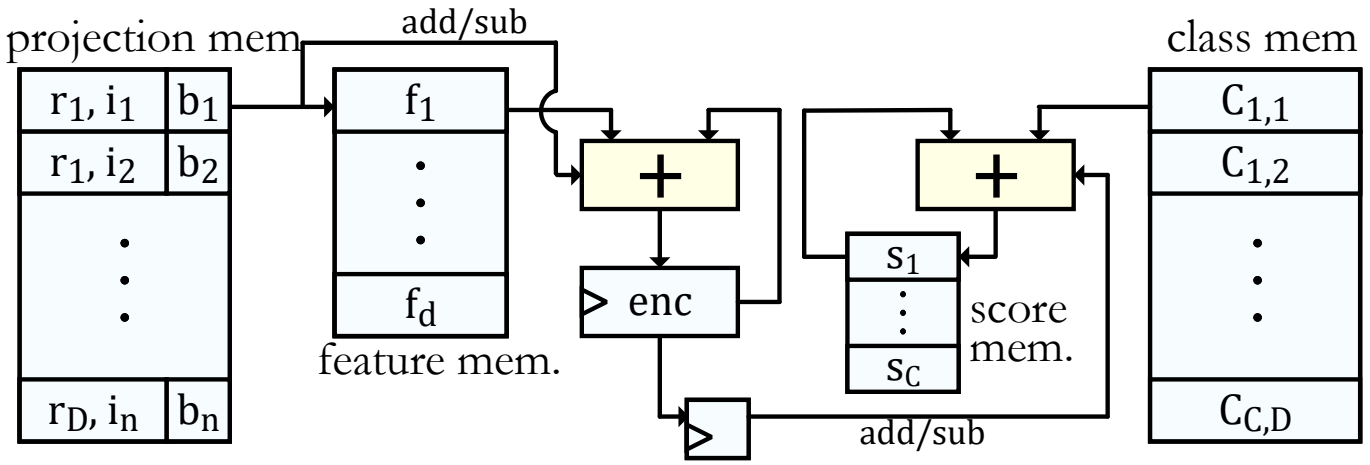
- Total bits: $D \times d \times (1 - s) \times (1 + \log d)$
  - D: dimensions d: features s: sparsity rate

- MNIST at 91% and UCIHAR at 83% sparsity compensate the index overhead and save memory



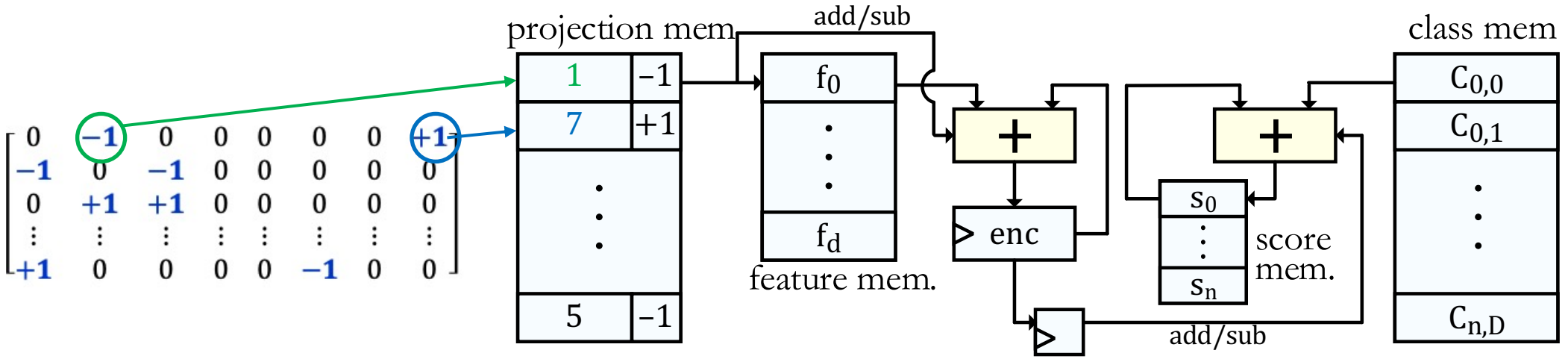
FPGA Implementation

- FPGAs are inimitable platform to realize HDC advantages
  - Customizable bit-width for operations and memory
  - Less data movement (single-cycle access to projection, classes, etc.)
  - Balance resource usage and power
  - …
- Datapath for generating one encoding element



FPGA Implementation

- Each cycle, one non-zero value of projection matrix is processed



- Cycle 1: $H[0] \leftarrow H[0] + \overbrace{B[0][1]}^{-1} \times F[1]$
- Cycle 2: $H[0] \leftarrow H[0] + B[0][7] \times F[7]$
- Cycle 3: $H[1] \leftarrow H[1] + B[1][0] \times F[0]$
- Cycle 5: …

$S[0] \leftarrow S[0] + H[0] \times C[0][0]$

$S[1] \leftarrow S[1] + H[0] \times C[1][0]$

Experimental Setup

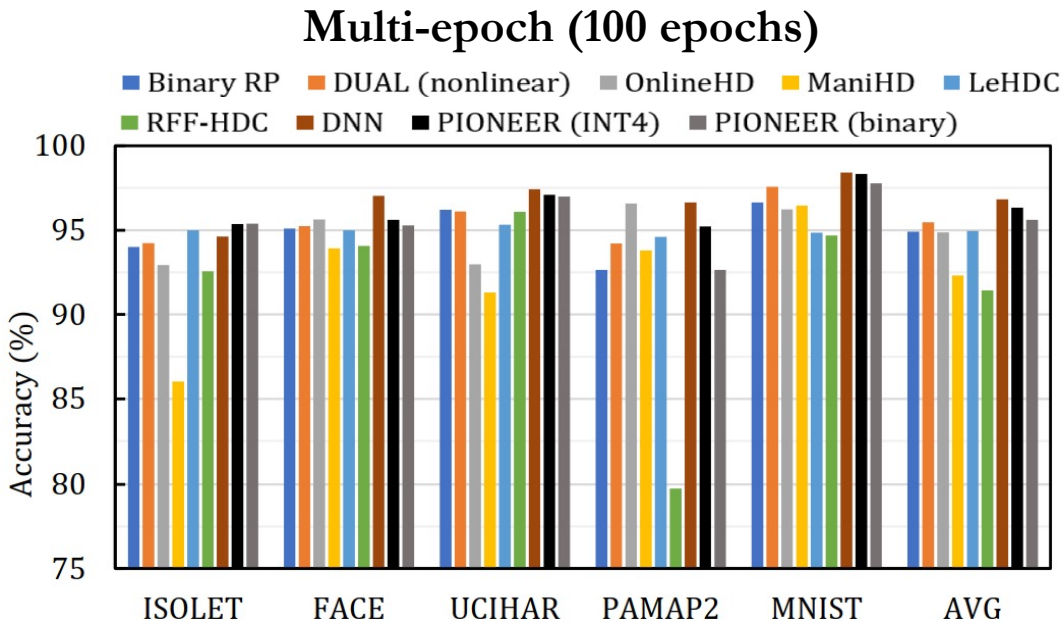
- Datasets

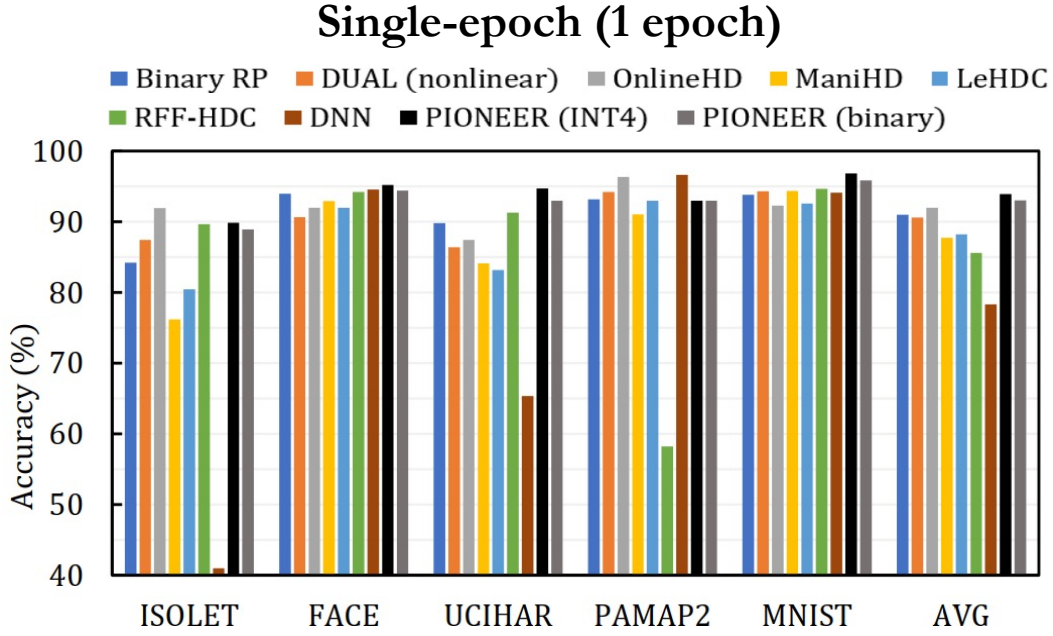
Dataset	Features	Classes	Train size	Test size	Description
CIFAR-10	1024	10	50,000	10,000	Vision dataset
FACE	608	2	22,441	2,494	Face recognition
ISOLET	617	26	6,238	1,559	Voice recognition
MNIST	784	10	60,000	10,000	Handwritten digits recognition
PAMAP2	27	5	16,384	16,384	Activity recognition
UCIHAR	516	12	6,213	1,554	Human activity recognition

- Baseline Comparison: Binary random projection, DUAL nonlinear encoding [Imani, Micro'20], OnlineHD [Cano, DATE'21], ManiHD [Zou, DATE'21], LeHDC [Duan, DAC'22], RFF-HDC [Yu, NeurIPS'22], DNN (three layer NN tuned using Ray Tune)

- Hardware Setup:
  - Implemented using Vivado HLS on Xilinx Kintex-7 Kit XC7K325T FPGA at 100 MHz frequency
  - Xilinx Power Estimator (XPE) for power estimation
  - DNNWeaver [Sharma, Micro'2016] to generate the Verilog code given the hyperparameters

Results: Accuracy



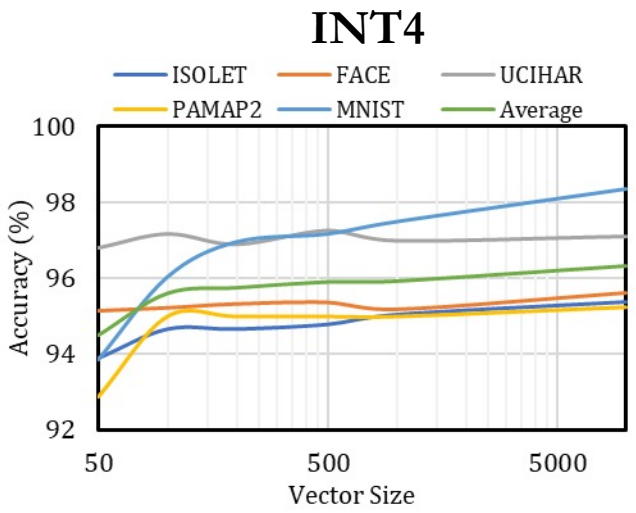


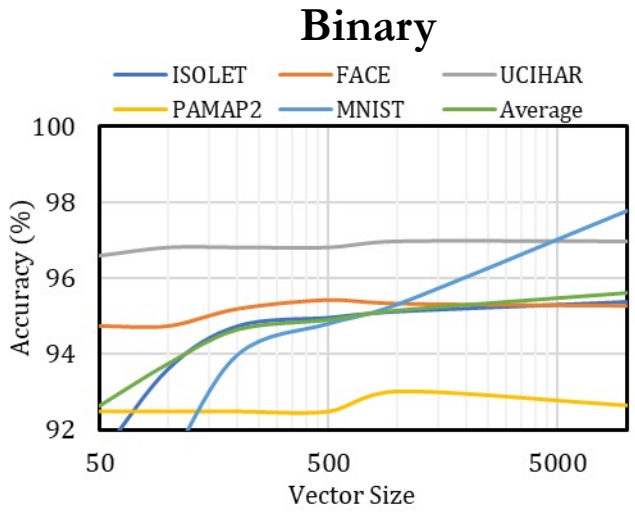
- PIONEER INT4: 96.33% (+0.86% vs nonlinear (float))
- PIONEER Binary: 95.61% (+0.14% vs nonlinear(float))
- 0.5% (INT4) and 1.22% (binary) drop vs DNN

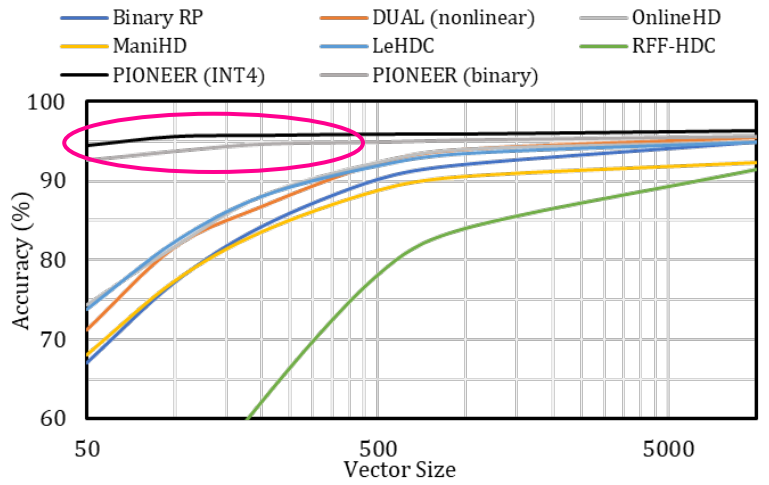
- PIONEER INT4: 93.93% (+1.92% vs OnlineHD)
- PIONEER Binary: 95.61% (+1.02% vs OnlineHD)
- 15.6% better than DNN

DUAL [Imani, Micro'20], OnlineHD [Cano, DATE'21], ManiHD [Zou, DATE'21], LeHDC [Duan, DAC'22], RFF-HDC [Yu, NeurIPS'22]

Results: (Down)-scalability



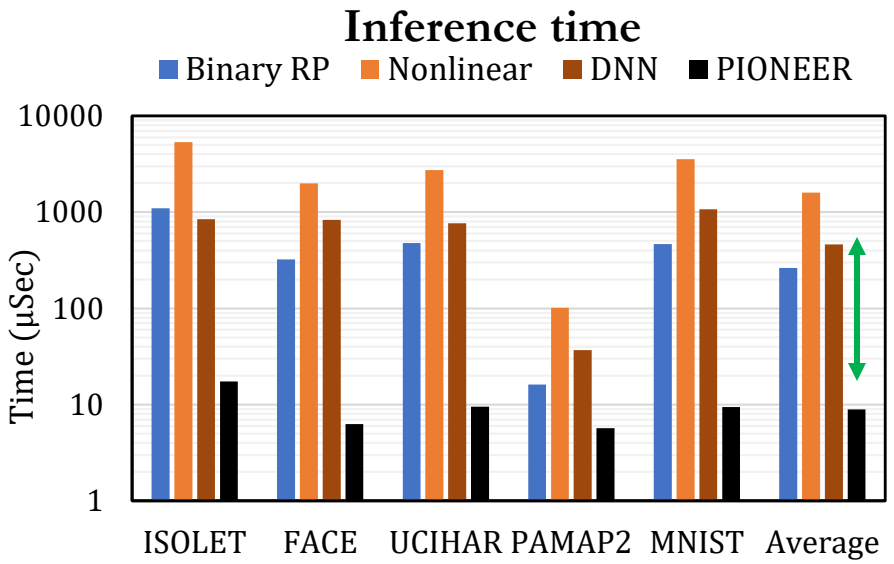


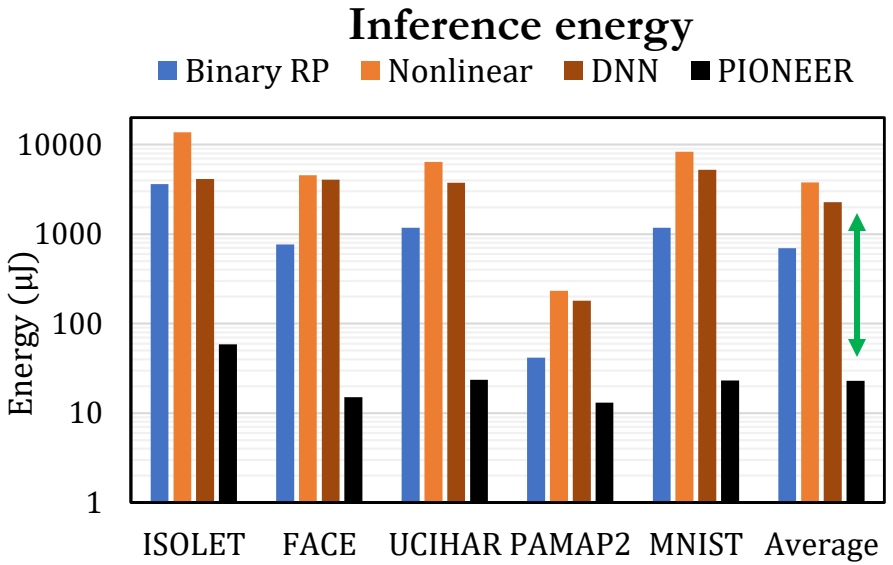


- At D = 100, PIONEER-INT4 achieves average of 95.61% (0.72% drop vs 10K)
- At D = 100, PIONEER-Binary achieves average of 93.75% (1.8% drop vs 10K)

- PIONEER demonstrates significantly better down-scalability
  - E.g., at D = 50, PIONEER achieves 26% higher accuracy over randomly initialized projection, which emphasizes the importance of learned projection

Results: Efficiency



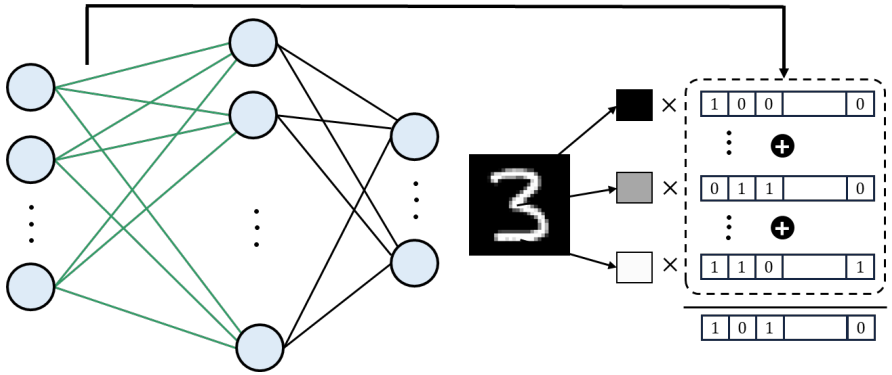


- PIONEER achieves faster inference time: 29× vs baseline binary RP, 179× vs nonlinear, and 52× vs DNN
  - PIONEER skips non-zero elements (98.4% sparsity rate, i.e., 63 out of 64 matrix elements are 0)

- PIONEER saves energy by 30× over binary RP, 165× over nonlinear HDC, and 98× over DNN
  - Designs consume similar power, so the energy saving is mainly affected by run-time

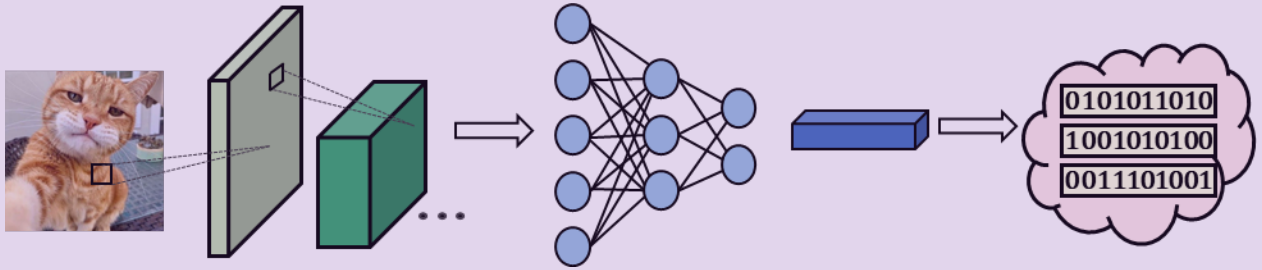
Contribution

Synergy of HDC and NN for efficient and private intelligence on edge



Efficiency **Accuracy**

ASP-DAC 2024

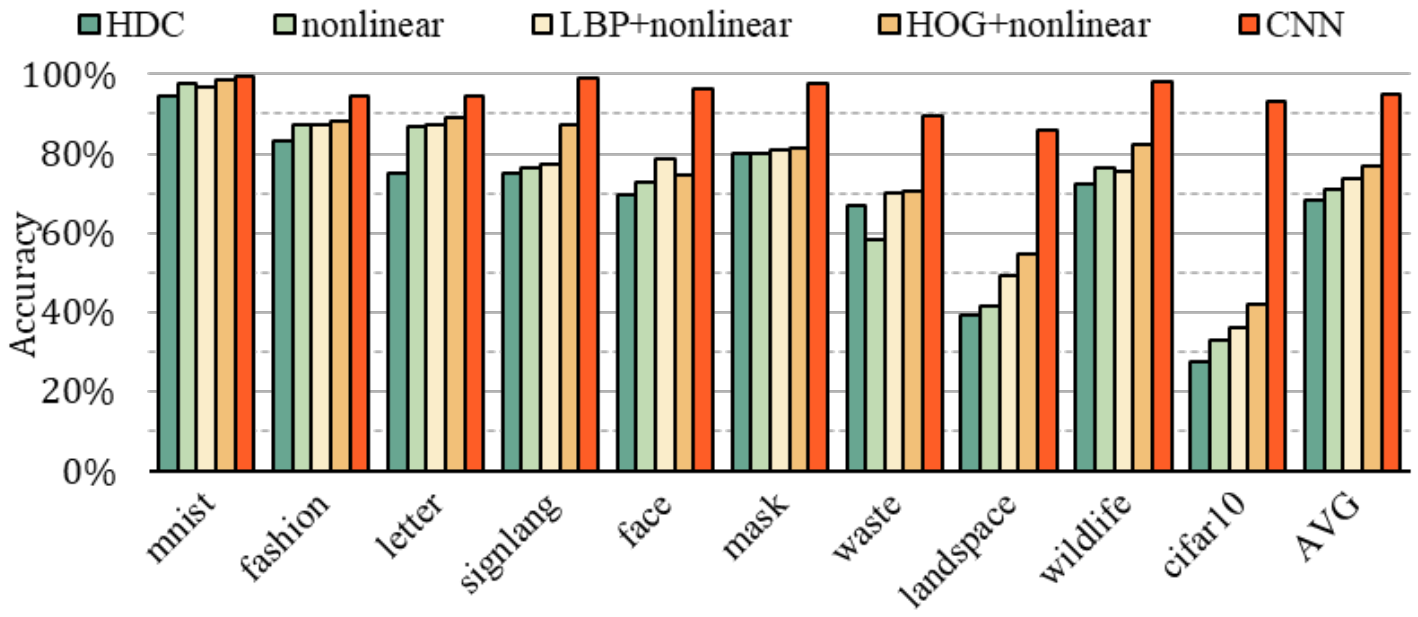


Images **Privacy**

DAC 2024 (WIP)

ISLPED 2024 (submitted)

HDC Challenge on Image Data: Accuracy

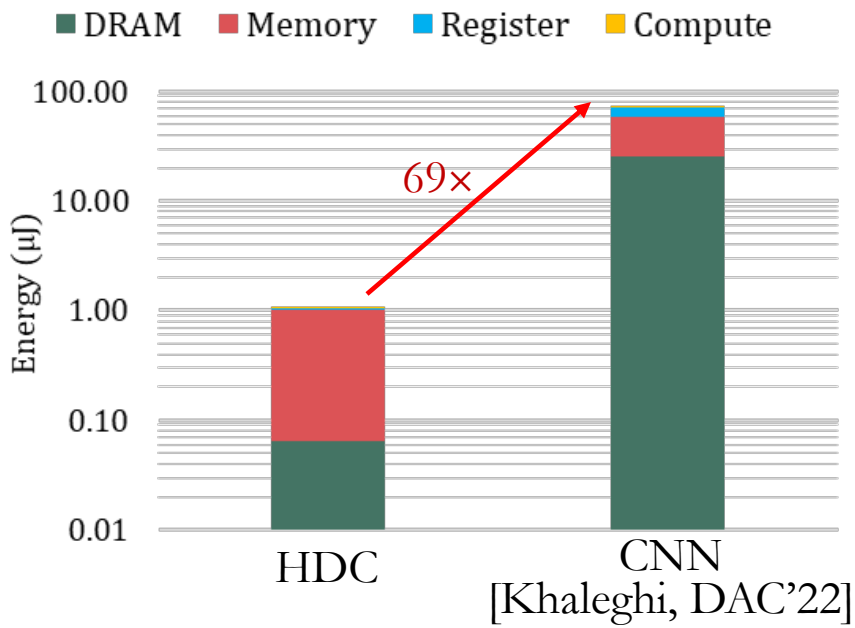


Approach	Avg. Accuracy
Baseline HDC	68.37%
Nonlinear	70.99%
LBP+nonlinear	73.89%
HOG nonlinear	76.89%
CNN (MobileNetV2)	94.88%

- Observation: Current HDC-based models exhibit notably lower accuracy (at least 18%) on image data compared to CNN models

HDC Challenge on Image Data: Efficiency

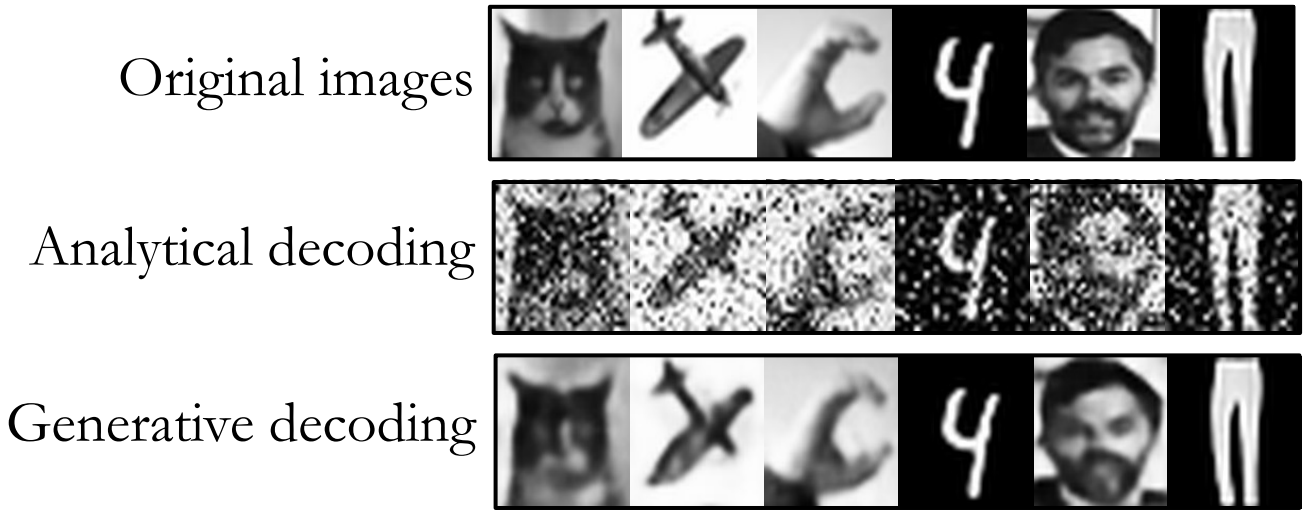
- Recent studies use CNN for feature extraction [Dutta, GLS-VLSI'22; Chandrasekaran, DAC'22]



- CNN feature extractor consumes 69× more energy than HDC
- Observation: Relying on existing CNN-based feature extraction shrinks HDC's energy gains

HDC Challenge on Image Data: Privacy

- Encoded hypervectors are decodable [Khaleghi, DAC'20; Cano, DAC'21, Cortes, DATE'24]
  - Data reconstruction attacks
  - Model inversion
  - etc.



- Observation: Privacy of HDC-based methods is challenged due to susceptibility to decoding.

Previous Work

Accuracy-Efficiency Trade-off

Novel Encoding & Training

- Spatial encoding [Miranda, HST'22]
- Random Fourier features [Yu, NeurIPS'22]
- Kernel-based multi-layer binary encoder [Yan, Springer'23]

- Limited accuracy: 92.2% for MNIST, 51.1% for CIFAR10

Previous Work

Accuracy-Efficiency Trade-off

Novel Encoding & Training

Classical Feature Extraction

- Hash networks e.g. deep quantization networks and deep Cauchy hashing [Mitrokhin, Frontiers'20]
- Histogram of Oriented Gradients (HOG) features [Duan, NanoArch'21; Imani, DAC'22]

- Significantly (18%) lower accuracy compared to CNNs

Previous Work

Accuracy-Efficiency Trade-off

Novel Encoding & Training | **Classical Feature Extraction** | **CNN Feature Extractor**

- CNN feature extractor + HDC head
  - In-field training [Dutta, GLSVLSI'22]
  - Few-shot learning [Hersche, CVPR'22; Xu, DATE'23]
  - Reduced communication [Chandrasekaran, DAC'22]
- These work leverage HDC's ancillary benefits
- Overall system efficiency is governed by the CNN component
  - Orders of magnitude more energy than HDC

Previous Work

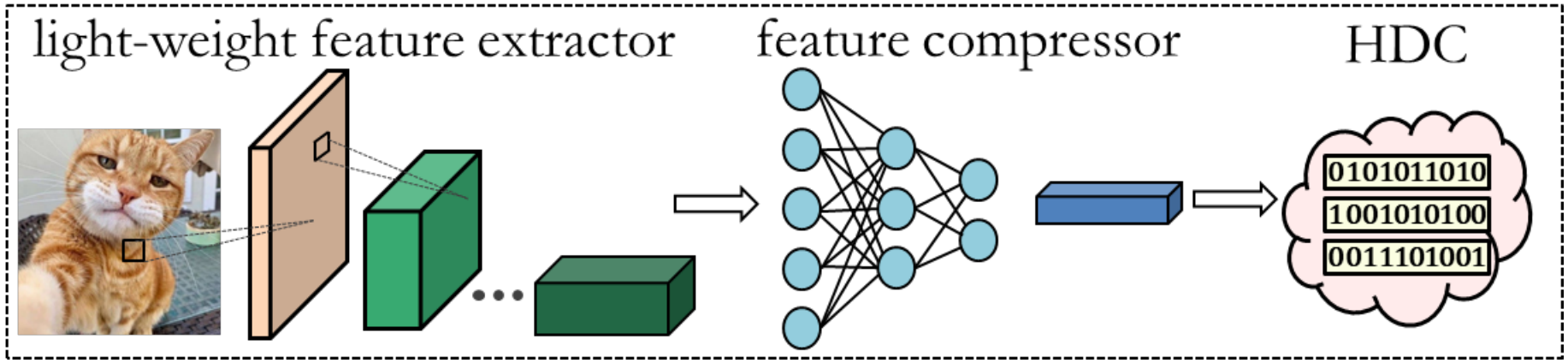
Accuracy-Privacy Trade-off

Investigating HDC Decodability

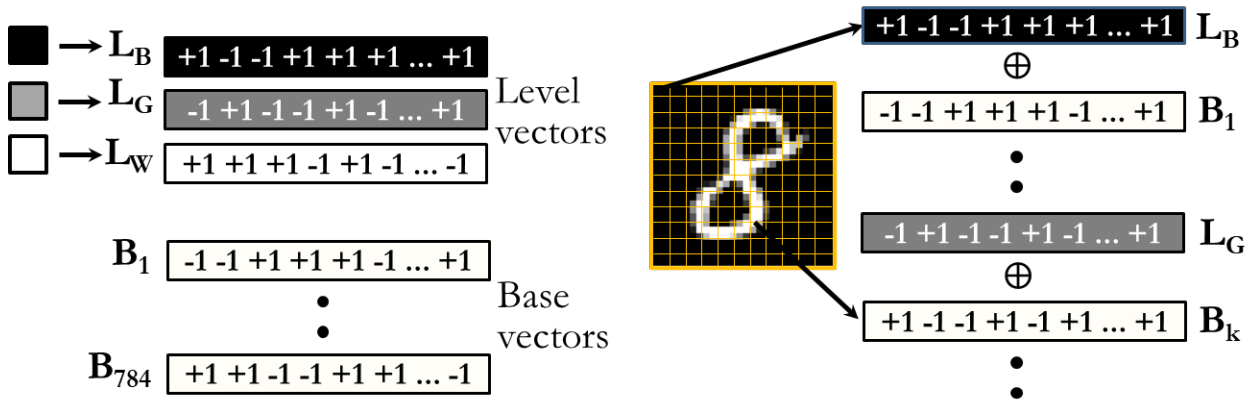
- Data reconstruction using analytical approach[Khaleghi, DAC'20]
- Data reconstruction using regression [Cano, DAC'21]
  - Need the model' parameters to be able to decode
  - Not effective on quantized/sparse vectors

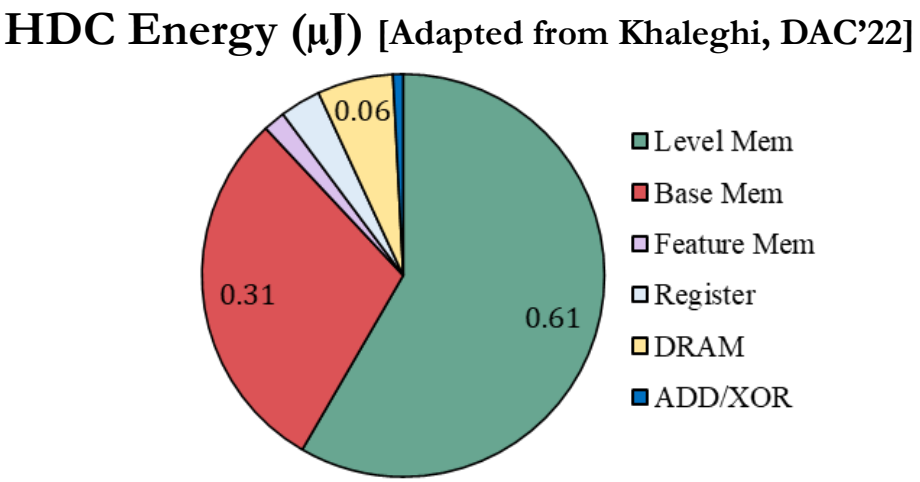
Proposed Method: VisionHD

- Algorithm-hardware co-design for efficient and privacy preserved image classification
  - Alleviate HDC's memory and energy bottleneck
  - Repurpose those resources for a light-weight feature extractor



HDC Energy Breakdown

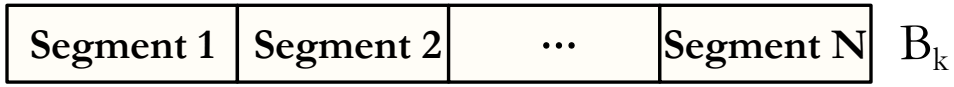




- Level vectors occupy 64KB memory and consume 58% of HDC energy
- Base vectors can be generated on-the-fly by shifting a seed vector [Khaleghi, DAC'22]
  - Base needs small memory
  - However, reading seed still consumes 30% of total energy
- Need a more efficient approach to generate level and base vectors

Vector-Free Encoding: On-the-fly Base Generation

- Jenkin's hash to generate segment **m** of base **k**

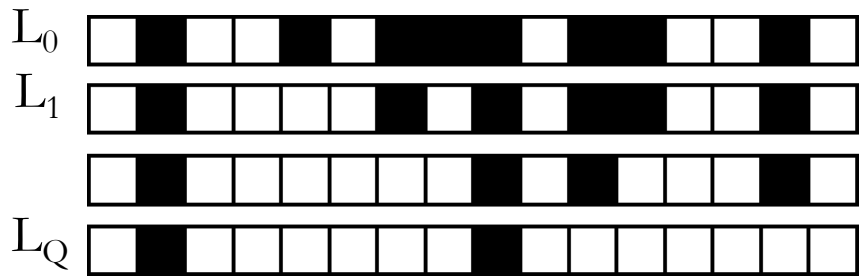



```
uint32_t jenkins_one_at_a_time_hash(const uint8_t* key, size_t length) {
  size_t i = 0;
  uint32_t hash = 0;
  while (i != length) {
    hash += key[i++];
    hash += hash << 10;
    hash ^= hash >> 6;
  }
  hash += hash << 3;
  hash ^= hash >> 11;
  hash += hash << 15;
  return hash;
}
```

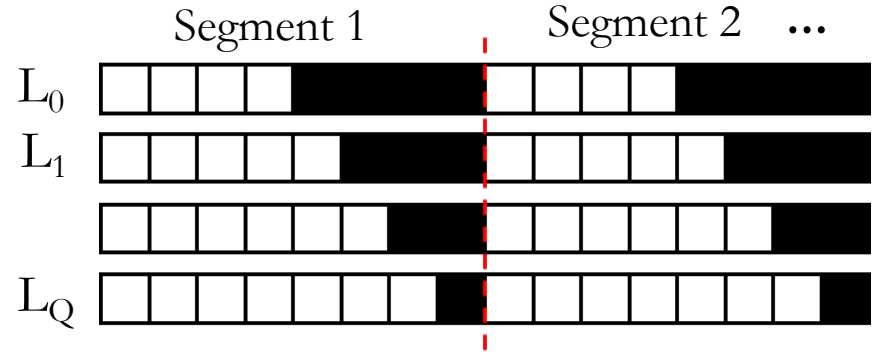
- segment(m, k) = Jenkins({m, k}, 4)
- Consumes only 0.014pJ (87× smaller than reading a 32b seed)

Vector-Free Encoding: On-the-fly Level Generation

- Levels need to preserve proximity
  - $L_0 \cdot L_Q \simeq 0$

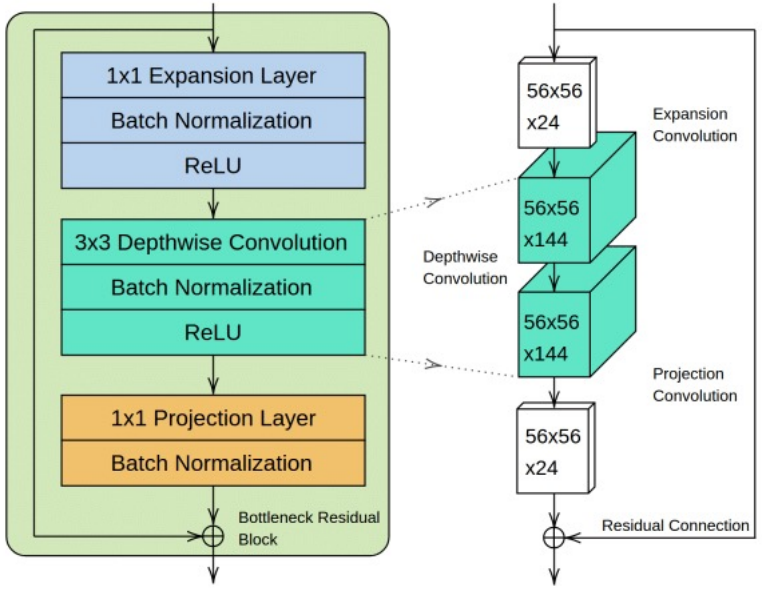


- Proposed method: structured level generation
- For each segment of level L_k
  - First half: all 0
  - Second half: first k bits are 0



- Efficient hardware: Q-to-2^Q decoder
- Consumes only 0.008pJ (303× smaller than 2.32pJ/access for reading level)

VisionHD Feature Extractor



	small	tiny	nano
C0	16	16	8
X1, C1, S1	1, 16, 1	1, 16, 1	1, 8, 1
X2, C2, S2	1, 20, 1	1, 20, 1	1, 10, 1
X3, C3, S3	1, 32, 2	1, 32, 2	1, 16, 2
X4, C4, S4	3, 32, 2	1, 32, 2	1, 16, 2
X5, C5, S5	3, 32, 1	1, 32, 1	1, 16, 1
X6, C6, S6	3, 32, 2	1, 32, 2	1, 16, 2
X7, C7, S7	3, 96, 1	1, 64, 1	1, 32, 1
C8	96	64	32

- MobileNetV2 backbone with tunable parameters
  - Expansion (X), Channel (C), Stride (S)
- Small (40K params), tiny (20K params), nano (5.5K params) configurations
  - 57−420× smaller than baseline MobileNetV2
  - 4.3−5.7M MACs (vs 98M of MobileNetV2)

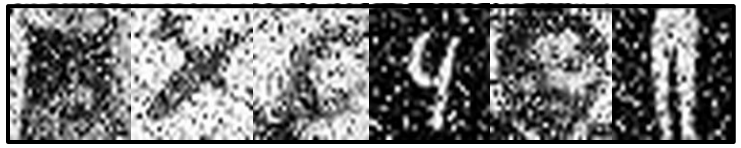
Backup Generative Decoder

```
def generator(D, size):
  return nn.Sequential(
    nn.Linear(D, size ** 2),
    nn.Linear(size ** 2, size ** 2),
    nn.Unflatten(1, (1, size, size)),
    nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3, stride=1, padding=1),
    nn.ReLU(),
    nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=1),
    nn.ReLU(),
    nn.Conv2d(in_channels=64, out_channels=1, kernel_size=3, stride=1, padding=1),
    nn.Sigmoid()
  )
```

- Generative decoder
  - Better reconstruction quality
  - Can map encoded "features" back to original images

Original images

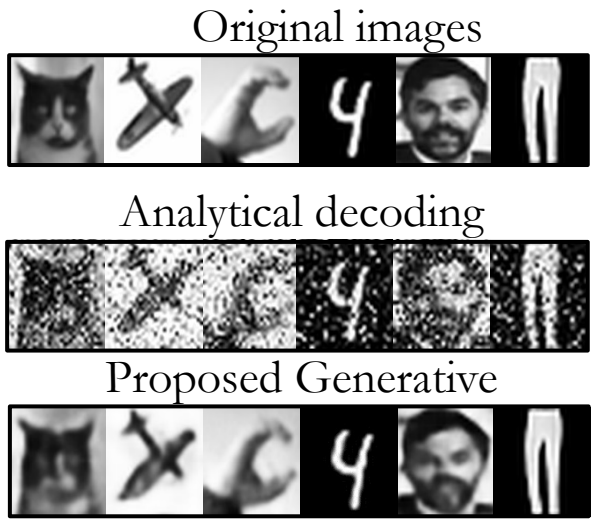
Analytical decoding



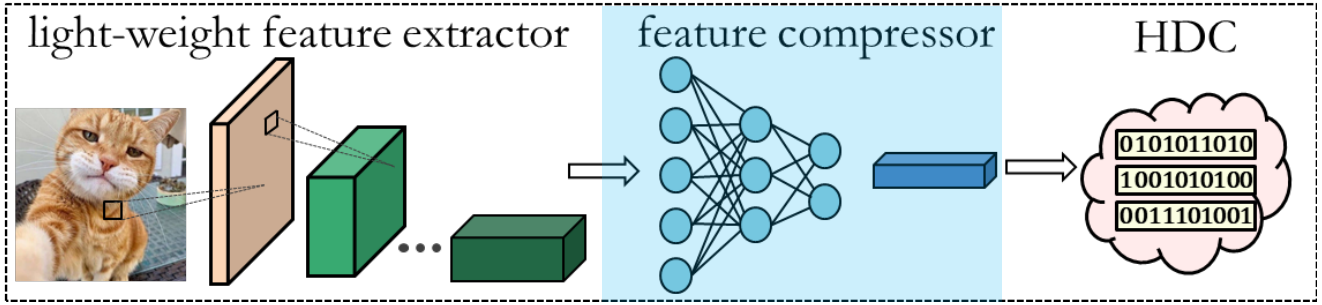
Proposed Generative

Feature Compressor

- Generative decoder
  - Better reconstruction quality
  - Can map encoded "features" back to original images

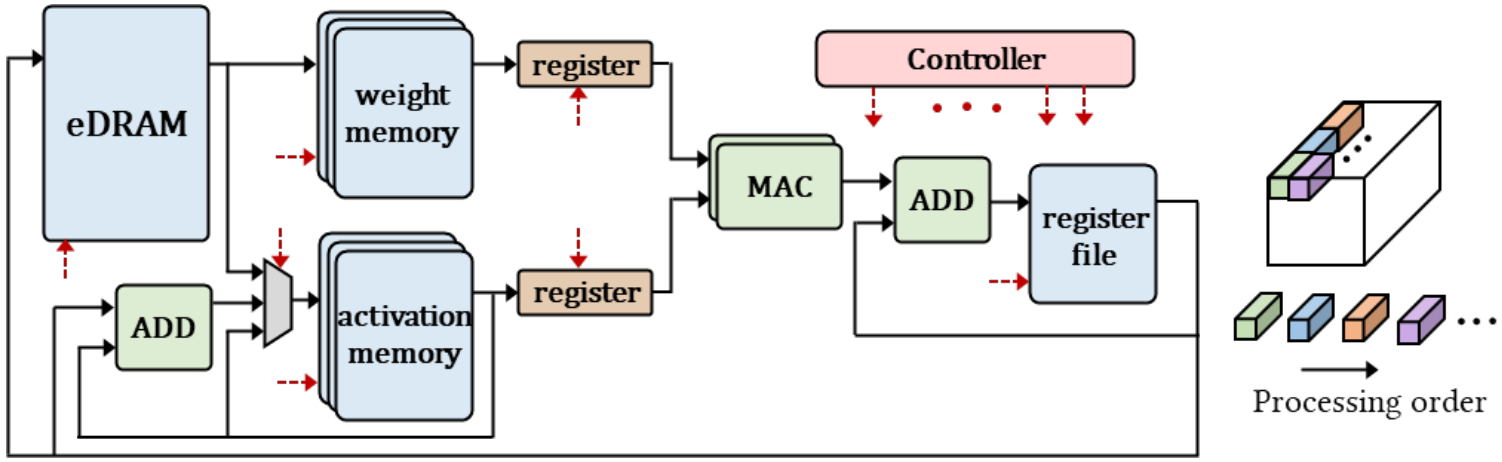


- (1) Train CNN feature extractor independently
- (2) Train MLP compressor + HDC on extracted features using SGD
- (3) With MLP fixed, train HDC using conventional accumulation



VisionHD Hardware Architecture

- All weights and activations fit in on-chip memories
- Once an input brick is read, all the corresponding weights are applied
- All partial outputs fit in register-file
- Each input is consumed only once



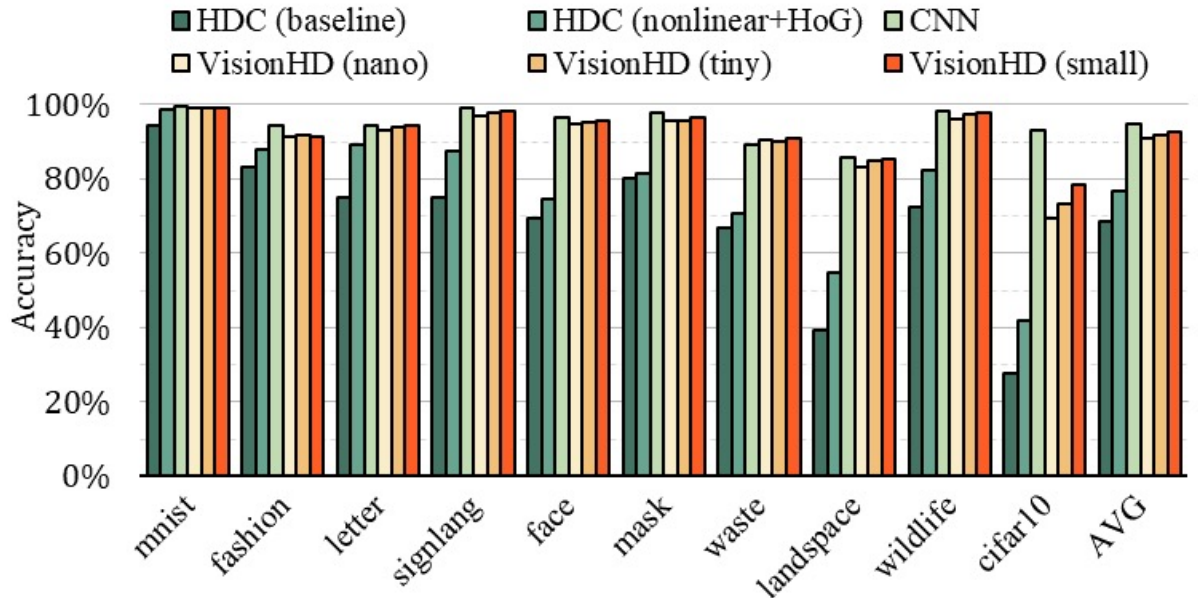
Experimental Setup

- Datasets

Dataset	Classes	Train size	Test size	Description
MNIST	10	60,000	10,000	Handwritten digits
Fashion	10	60,000	10,000	Clothing
Letter	26	60,000	10,000	Handwritten English letters
Sign Lang	24	27,455	7,172	American sign language
Face	2	10,638	3,546	Human face vs non-face
Mask	2	5,664	1,889	Detecting mask wearing
Waste	2	22,564	2,513	Organic vs recyclable
Landscape	6	14,034	3,000	Buildings, forest, etc.
Wildlife	3	14,630	1,500	Cats vs dogs vs other
CIFAR-10	10	50,000	10,000	Airplanes, cars, birds, etc.

- All algorithms implemented in Python (PyTorch for NNs; Ray-Tune for hyper-parameter tuning; AIMET for quantization)
  - VisionHD and baseline MobileNetV2: 100 epochs, batch size 64, lr 1e-2 to 1e-4
- Hardware setup: implemented in Verilog, synthesized in 14nm

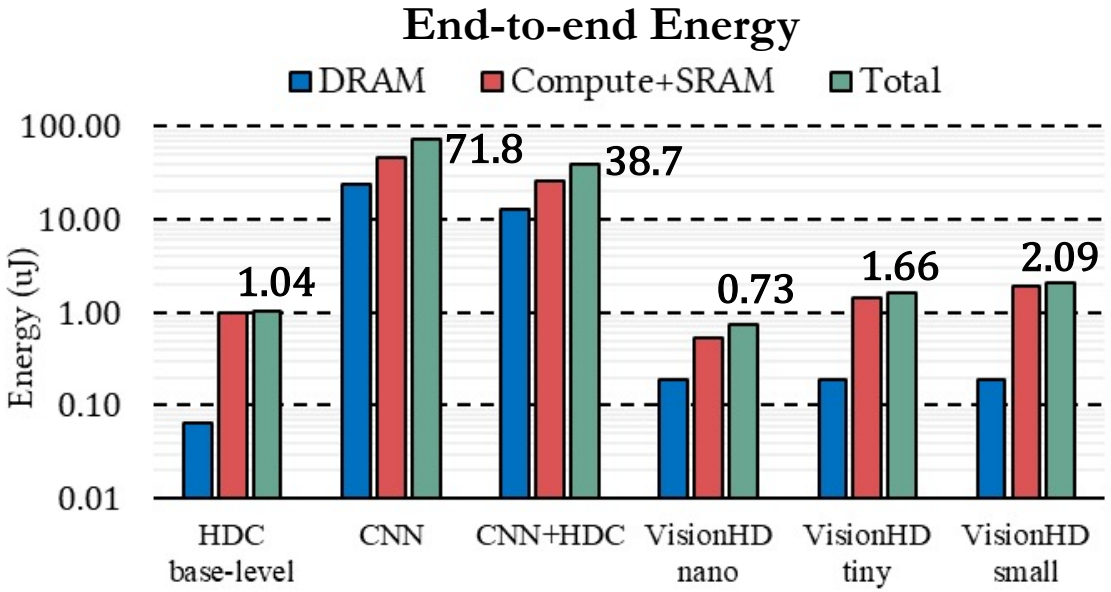
Results: Accuracy

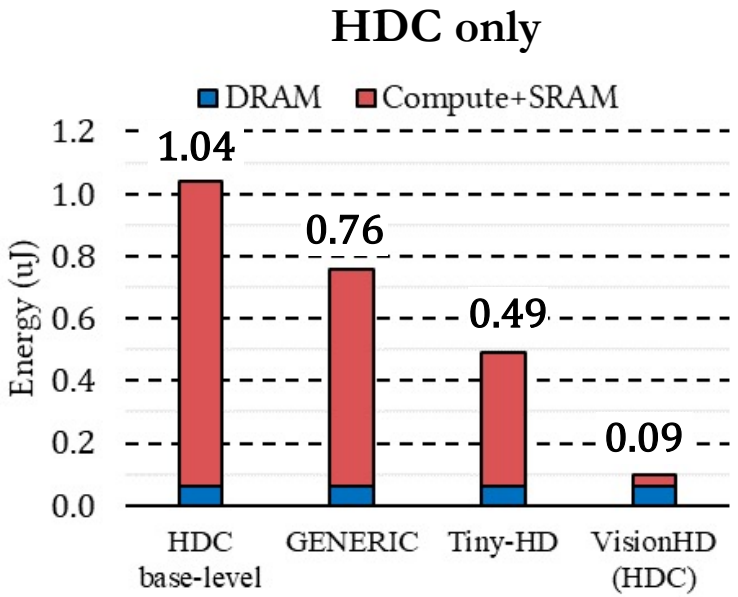


Approach	Avg. Accuracy
Baseline HDC	68.37%
Nonlinear + HOG	76.89%
CNN (MobileNetV2)	94.88%
VisionHD nano	91.01%
VisionHD tiny	91.94%
VisionHD small	92.76%

- 22.6−24.4% higher accuracy compared to baseline HDC
- 14.1−15.9% better than best HDC baseline (nonlinear encoding with HOG features)
- 2.1−3.9% lower accuracy compared to CNN
  - 0.7−1.6% excluding CIFAR10

Results: Energy Efficiency





- 34−98× more efficient than CNN [Khaleghi, DAC'22] (2.1−3.9% lower accuracy cost)
- 18−53× more efficient than HDnn (CNN+HDC, [Dutta, GLSVLSI'22])

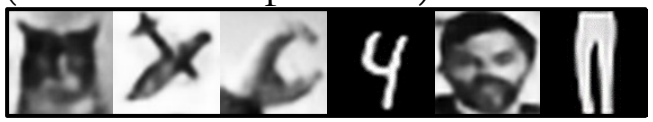
- 10.8× more efficient than baseline, 7.8× vs GENERIC [Khaleghi, DAC'22], 5.1× vs tiny-HD [Khaleghi, DATE'21]

Results: Privacy

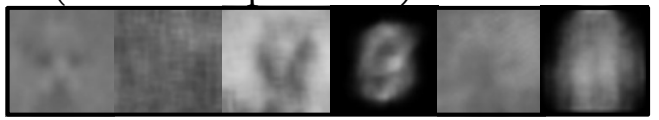
- Proposed generative decoding could reconstruct both conventional and CNN-based HDC (including VisionHD)
- VisionHD leverages compression to obfuscate reconstruction

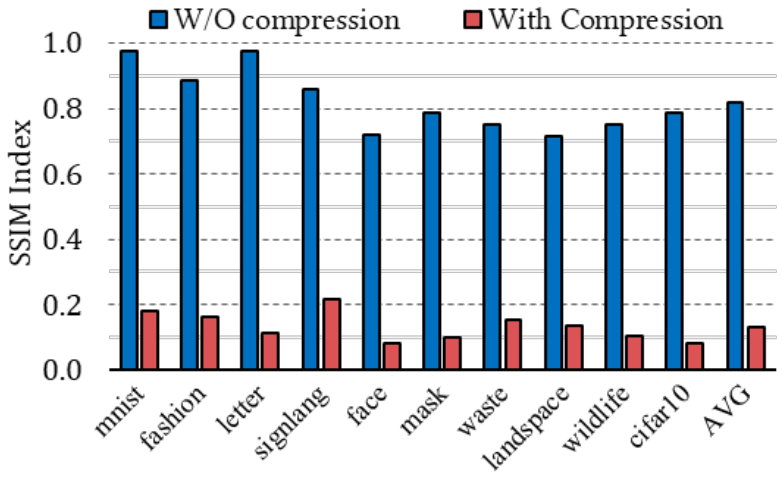
Original images

Decoded VisionHD (**without** compression) vectors

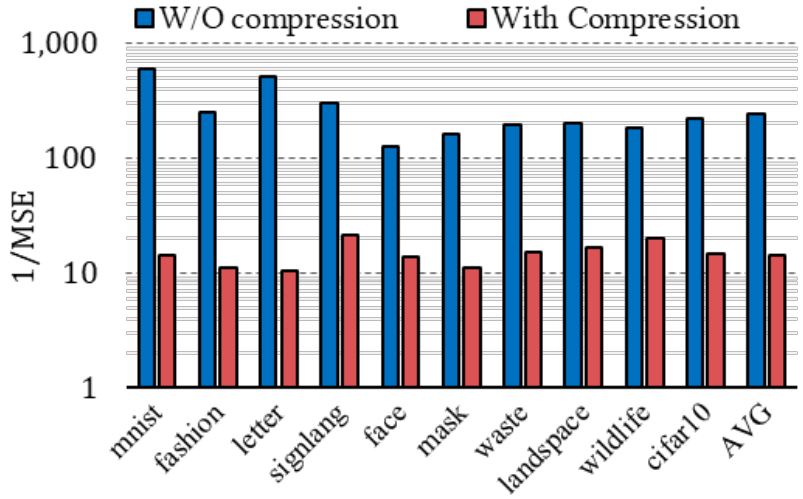


Decoded VisionHD (**with** compression) vectors





SSIM from 0.82 to 0.13



1/MSE from 243 to 14.4

Conclusion

- We adopt a NN-based mechanism to learn the projection
  - PINEER improves accuracy by 18.3% over state-of-the-art at D=50.
  - In multi-pass training with D=10k, PIONEER beats best single-pass by 1.92% and is 179× faster and 165× more energy efficient than state-of-the-art HDC

Efficiency **Accuracy**

Privacy **Images**

- We propose on-the-fly vector generation and repurpose the released resources for opting a light feature extractor
  - VisionHD is **34-98×** more efficient than CNN and outperforms best HDC baseline by **15.9%**
  - The average SSIM of compressed features decoding reduces to 0.13, with an inverse MSE (1/MSE) of 14.4 compared to 243 of the baseline

Future works

Security

- We propose a novel and effective HDC-based adversarial attack generation
- target: ICCAD2024, May (just submitted)

Efficiency **Privacy** **Federated Learning**

- We propose an efficient and robust HDC-based Federated Learning framework
- target: DATE2025, September

Efficiency **Accuracy**

- We propose an extension of PIONNER that integrate learned projection with lightweight feature extraction for further improving accuracy
- target: IEEE TCAD, November

Security

- We propose a defense mechanism to tackle adversarial attacks in HDC
- target: IEEE IoT, February 2025

Defense: June 2025

Publications

Published papers

- **F. Asgarinejad**, X. Yu, D. Jiang, J. Morris, T. Rosing, B. Aksanli. Enhanced Noise-Resilient Pressure Mat System Based on Hyperdimensional Computing, Sensors, 2024.
- **F. Asgarinejad**, J. Morris, T. Rosing, B. Aksanli. PIONEER: Highly Efficient and Accurate Hyperdimensional Computing using Learned Projection, ASP-DAC, 2024.
- R. Chandrasekaran, **F. Asgarinejad**, J Morris, T. Rosing. Multi-label classification with Hyperdimensional Representations, IEEE ACCESS, 2023.
- X. Yu, M. Zhou, **F. Asgarinejad**, O. Gungor, B. Aksanli, T. Rosing. Lightning Talk: Private and Secure Edge AI with Hyperdimensional Computing. DAC, 2023
- R. Garcia, **F. Asgarinejad**, B. Khaleghi, T. Rosing, M Imani. TruLook: A framework for configurable GPU approximation, DATE, 2021.
- Z. Zhang, R. Hildebrant, **F. Asgarinejad**, N. Venkatasubramanian, S. Ren. Improving process discovery results by filtering out outliers from event logs with hidden Markov models, CBI, 2021.
- **F. Asgarinejad**, A. Thomas, T. Rosing. Detection of Epileptic Seizures from Surface EEG using Hyperdimensional Computing, EMBC, 2020.
- B. Khaleghi, S. Salamat, A. Thomas, **F. Asgarinejad**, Y. Kim, T. Rosing. SHEARer highly-efficient hyperdimensional computing by software-hardware enabled multifold approximation, ACM, 2020.

Submitted Papers

- **F. Asgarinejad**, F. Pozina, O. Gungor, T. Rosing, B. Aksanli. Understanding to Deceive: Harnessing Hyperdimensional Computing's Explainability for Adversarial Attacks, ICCAD, 2024.
- F. Asgarinejad, J. Morris, T. Rosing, B. Aksanli. VisionHD: Revisiting Hyperdimensional Computing for Improved Image Classification, ISLPED, 2024.

Under submission

- **F. Asgarinejad**, A. Thomas, R. Hildebrant, Z. Zhang, S. Ren, T. Rosing, B. Aksanli, "Predicting the outcome of an ongoing business process based on event logs using hyperdimensional computing", *To be submitted to MDPI,* 2024.
- M. Gaddi, **F. Asgarinejad**, F. Pozina, B. Aksanli , T. Rosing, RadarHD: Resilient Radar Signal Inference with Hyperdimensional Computing, to be submitted to Biosys, 2024.

Backup: Accuracy of VisionHD on Cifar100

Approach	Avg. Accuracy
Baseline Base-level HDC	7.99%
Nonlinear	12.25%
Nonlinear + HOG	17.68%
CNN (Mobilenet)	72.66%
VisionHD nano	41.77%
VisionHD tiny	52.57%
VisionHD small	61.33%

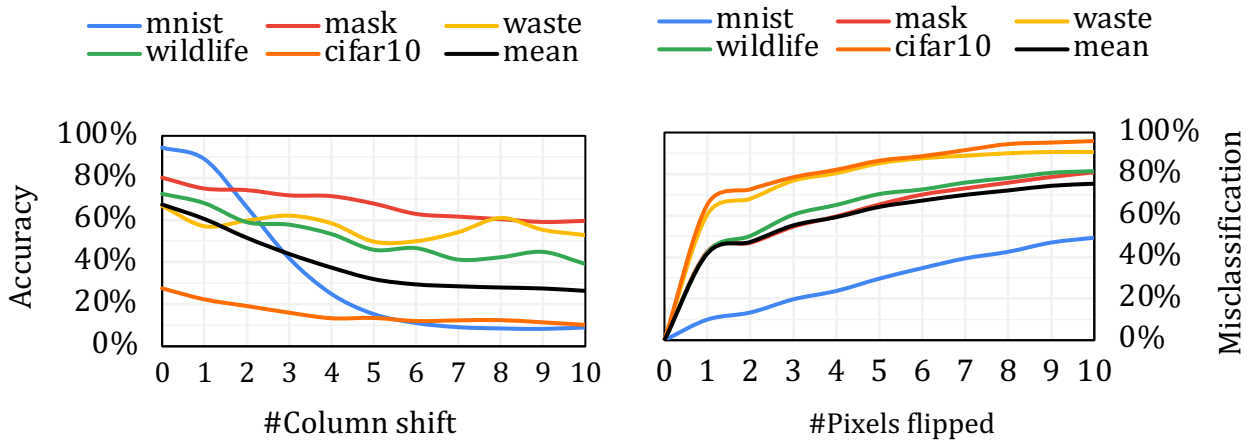
- 24.9−43.63% higher accuracy compared to best HDC-based model
- 11.33% lower accuracy compared to CNN

Backup: Future works

Security

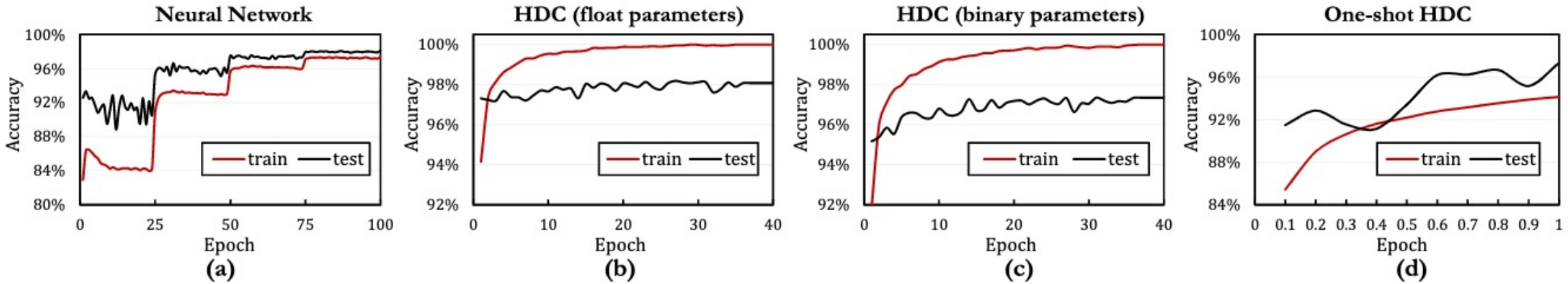
Submitted to ICCAD 2024

HDC is susceptible to adversarial attacks



- Simple noise in input data can perturb HDC
- We propose an HDC-based adversarial attack and use the perturbed samples as a defense mechanism for improving model's tolerance against noise

Backup (Accuracy of PIONEER vs NN on MNIST)

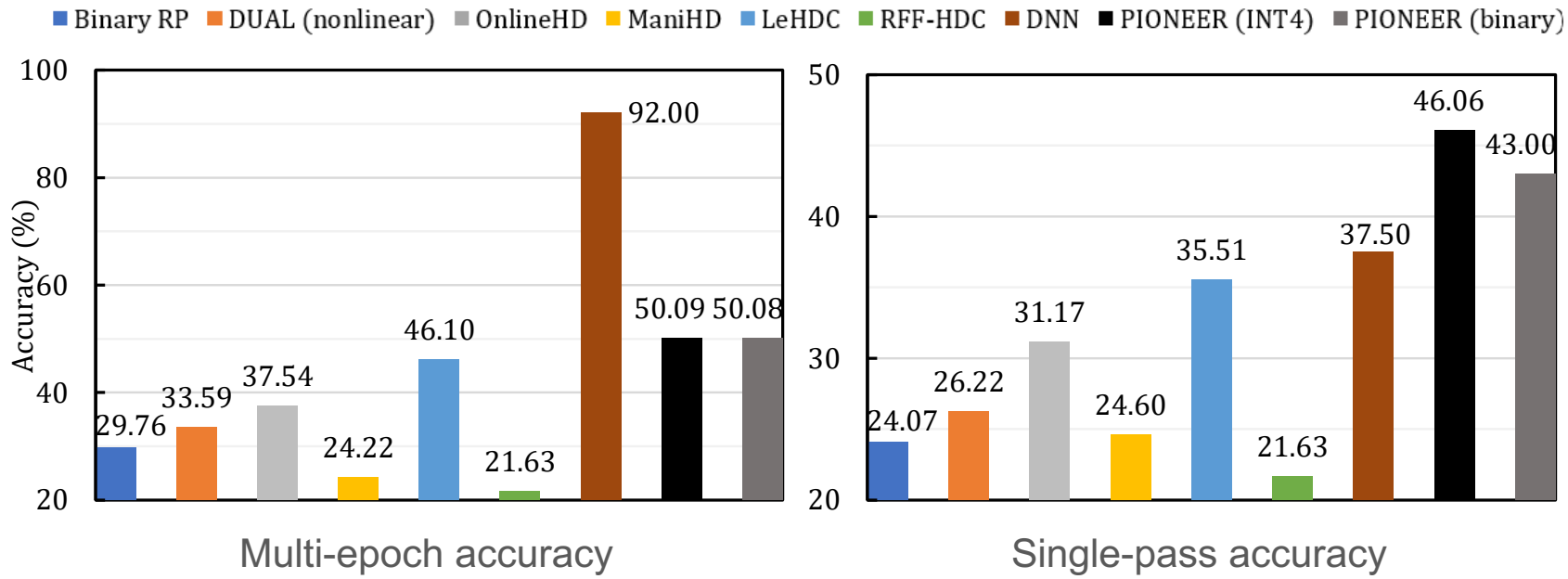


MNIST train and test accuracy of **(a)** HDC modeled as a neural network, **(b)** HDC with (float) learned parameters (PIONEER), **(c)** HDC with (binarized) learned parameters (PIONEER), and **(d)** one-shot HDC with learned parameters (PIONEER).

- NN model achieves a 98.09% accuracy after 100 epochs, while PIONEER using the projection constants learned by the same NN model achieves an accuracy of **98.08%** after **37 epochs**
- The **one-shot** (single-epoch) accuracy of PIONEER with the learned projection is **97.31%**

Backup (Accuracy of CIFAR-10 in Single-Pass and Multi-Epoch Training)

- For CIFAR10 dataset, PIONEER achieves **50.10%** accuracy in multi-epoch training, which is **4.0%** better than the best HDC baseline

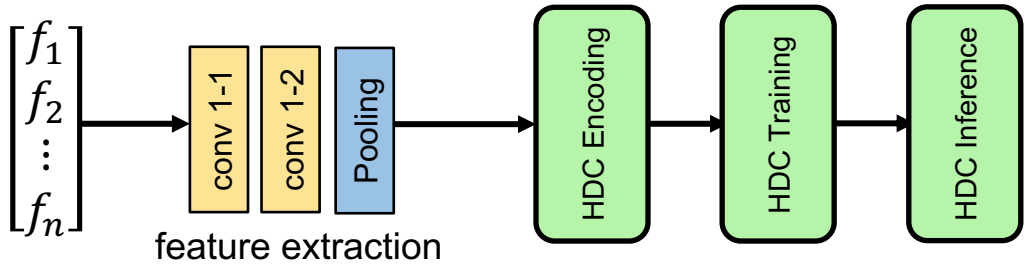


Backup: Related Works for PIONEER

There exists an accuracy gap between NN-based models and HD Computing

Studies that try to fill the accuracy gap between NN and HDC:

- Extract the features using a NN and then use a HDC head for learning and classification [Dutta'22, Nazemi'22, Poduval' 21, Imani'22, Duan'21]



Dutta et al., 2021

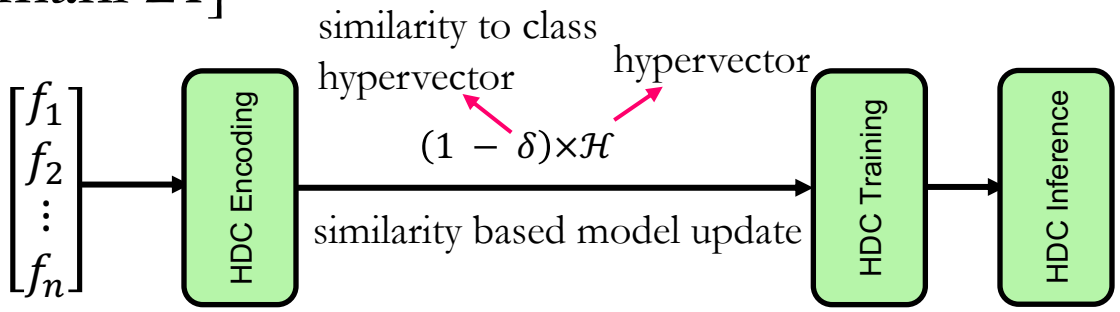
- Jeopardize the robustness of HDC and create performance bottleneck with computation of features

Backup: Related Works for PIONEER

There exists an accuracy gap between NN-based models and HD Computing

Studies that try to fill the accuracy gap between NN and HDC:

- Propose novel encoding or training [Cano'21, Zuo'21, Imani'20, Khaleghi'22, Imani'21]



Cano et al., 2021

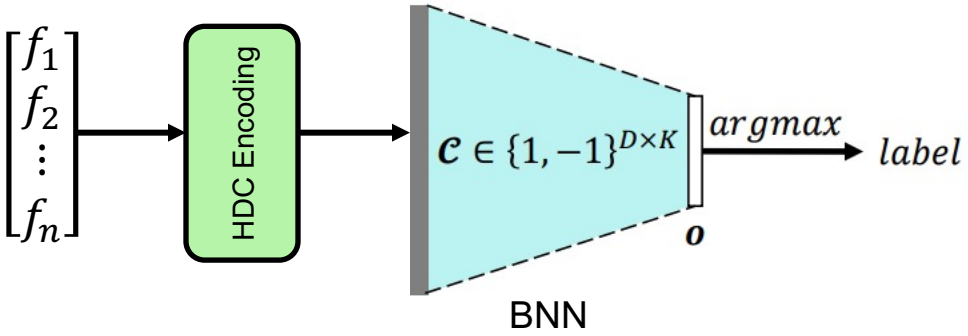
  - Exhibit increased operational complexity and demonstrate inferior accuracy under quantization or with small vector dimensions

Backup: Related Works for PIONEER

There exists an accuracy gap between NN-based models and HD Computing

Studies that try to fill the accuracy gap between NN and HDC:

- Employ neural networks to train HDC models [Duan'21, Yu'22]



Duan et al., 21

Class vectors are modeled as a hidden layer with weights corresponding to class elements

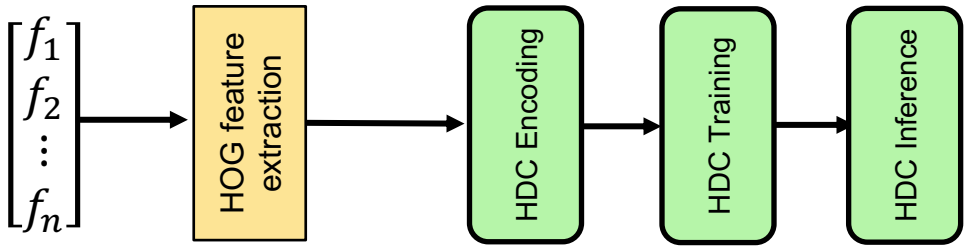
- Require complex operations and update all parameters for each sample
- Focuses on training and deviates from straightforward simple HDC training

Backup: Related Works for VisionHD

There exists challenges in terms of HDC's performance on image data

Studies that try to enhance the performance of HDC on image data:

- Implement traditional feature extractors [Miranda'22, Duan'22, Yu'2022]



Imani et al., 2022

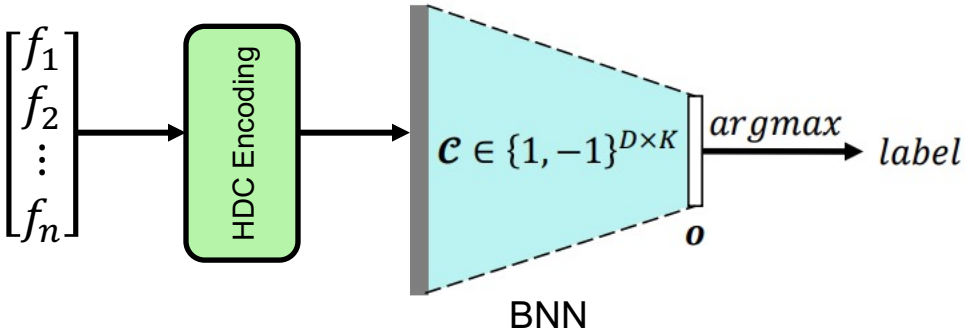
- Create performance bottleneck with computation of features and perform poorly on complex images

Backup: Related Works for VisionHD

There exists challenges in terms of HDC's performance on image data

Studies that try to enhance the performance of HDC on image data:

- Propose novel encoding or training [Mitrokhin'20, Duan'21, Imani'22]



Duan et al., 21

Class vectors are modeled as a hidden layer with weights corresponding to class elements

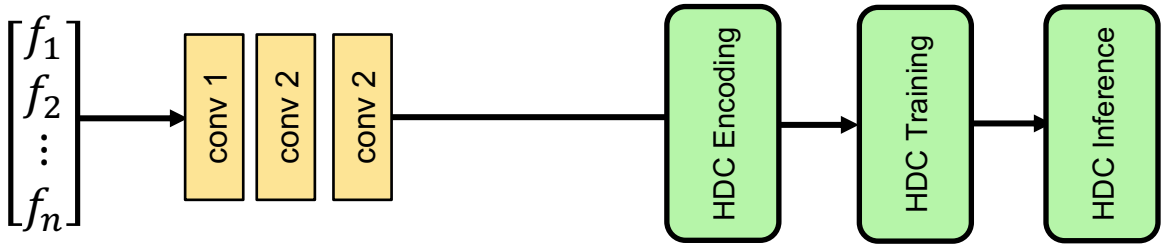
- Perform poorly on relatively complex images

Backup: Related Works for VisionHD

There exists challenges in terms of HDC's performance on image data

Studies that try to enhance the performance of HDC on image data:

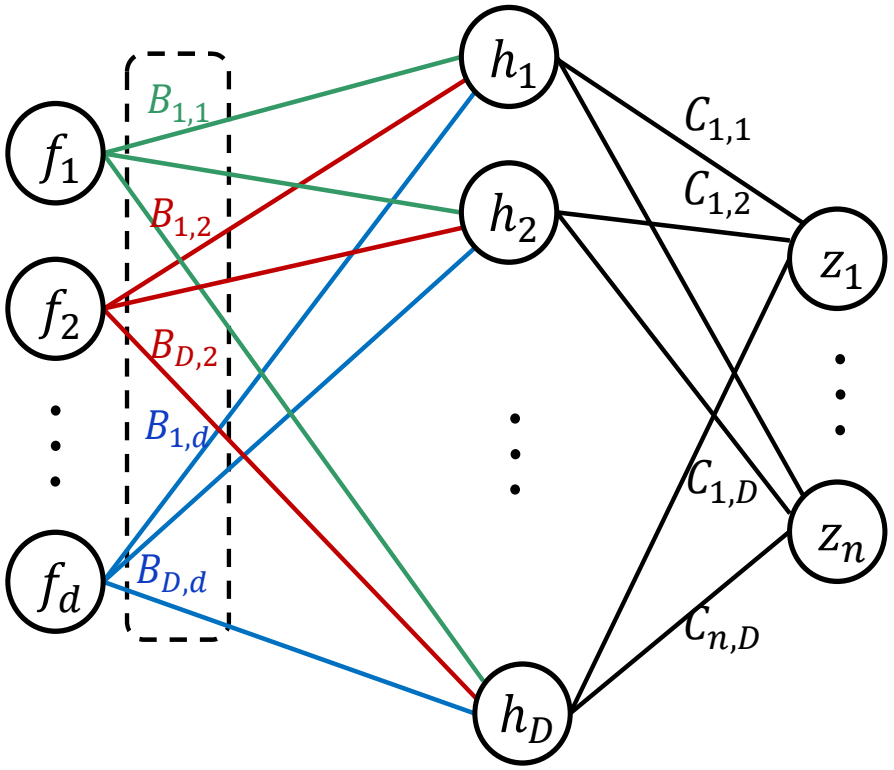
- Employ neural networks to train HDC models [Duan'22, Chandrasekaran'22]



Chandrasekaran et al., 2022

- In these methods the overall system's efficiency is primarily governed by the CNN component that consumes orders of magnitude more energy than HDC

Learning the Projection Matrix Mathematics



Update in class hypervectors in above NN aligns with HDC class updates

$$(1)\ L = -\sum_{i=1}^{n} y_i \log \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}} = -\sum_{i}^{n} y_i \log s_i \quad \text{loss function}$$

$$(2)\ C_{1,1}^{(t+1)} = C_{1,1}^{(t)} - \lambda \frac{\partial L}{\partial C_{1,1}} \qquad (3)\ \frac{\partial L}{\partial C_{1,1}} = \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial C_{1,1}}$$

$$(4)\ \frac{\partial L}{\partial z_i} = s_i - y_i \qquad (5)\ \frac{\partial z_1}{\partial C_{1,1}} = \frac{\partial (h_1 . C_{1,1} + \cdots + h_D . C_{1,D})}{\partial C_{1,1}} = h_1$$

$$(6) \rightarrow \frac{\partial L}{\partial C_{1,1}} = (s_1 - y_1)h_1 = \left(\frac{e^{z_1}}{\sum_{k=1}^{n} e^{z_k}} - y_1\right) h_1 = \alpha_1\ h_1$$

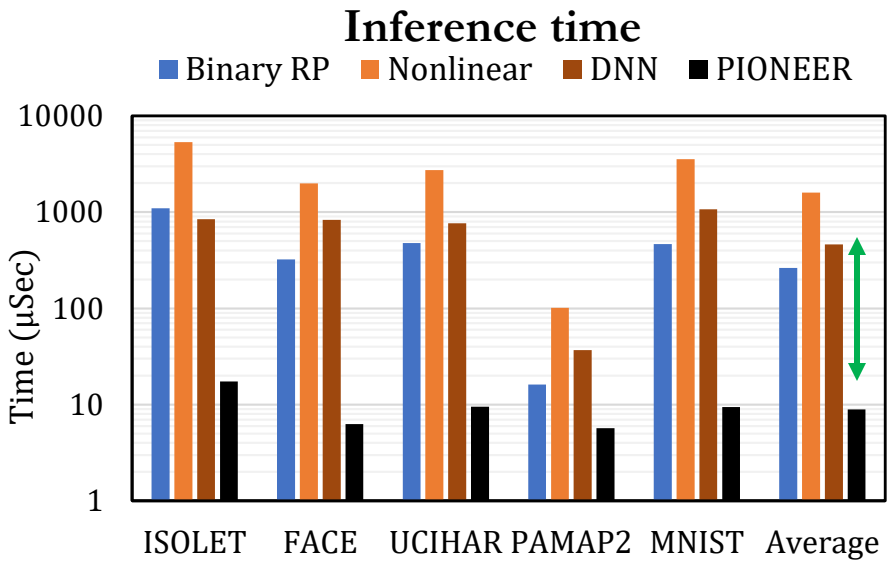
When label is, e.g., C1 ($y = [1,0,\cdots,0]$):

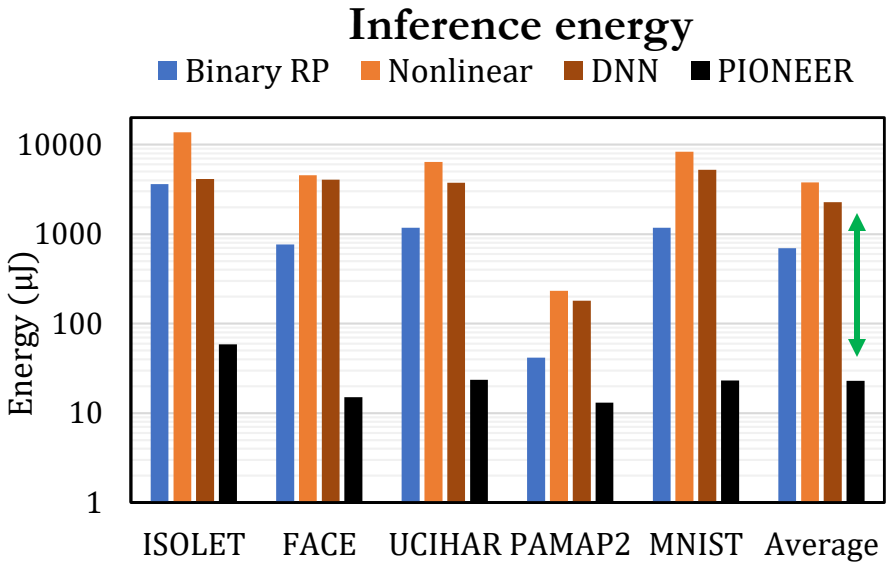
$$y_1 = 1\ , \frac{e^{z_1}}{\sum_{k=1}^{n} e^{z_k}} = \alpha_1 \rightarrow \quad C_{1,1}^{(t+1)} = C_{1,1}^{(t)} + \lambda(1-\alpha_1)h_1 = C_{1,1}^{(t)} + \lambda\alpha_2 h_1$$

When label is not C1 ($y = [0,\cdots]$):

$$y_1 = 0\ , \frac{e^{z_1}}{\sum_{k=1}^{n} e^{z_k}} = \alpha_3 \rightarrow \quad C_{1,1}^{(t+1)} = C_{1,1}^{(t)} - \lambda\alpha_3\ h_1$$

Results: Efficiency





- PIONEER achieves faster inference time: 29× vs baseline binary RP, 179× vs nonlinear, and 52× vs DNN
  - PIONEER skips non-zero elements (98.4% sparsity rate, i.e., 63 out of 64 matrix elements are 0)

- PIONEER saves energy by 30× over binary RP, 165× over nonlinear HDC, and 98× over DNN
  - Designs consume similar power, so the energy saving is mainly affected by run-time
  - PIONEER: 2.6W Baseline RP: 2.7W Nonlinear: 2.4 W

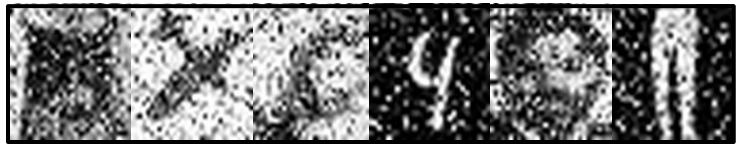
Backup Generative Decoder

```
def generator(D, size):
  return nn.Sequential(
    nn.Linear(D, size ** 2),
    nn.Linear(size ** 2, size ** 2),
    nn.Unflatten(1, (1, size, size)),
    nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3, stride=1, padding=1),
    nn.ReLU(),
    nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=1),
    nn.ReLU(),
    nn.Conv2d(in_channels=64, out_channels=1, kernel_size=3, stride=1, padding=1),
    nn.Sigmoid()
  )
```

- Generative decoder
  - Better reconstruction quality
  - Can map encoded "features" back to original images

Original images

Analytical decoding



Proposed Generative