PIONEER: Highly Efficient and Accurate Hyperdimensional Computing using Learned Projection

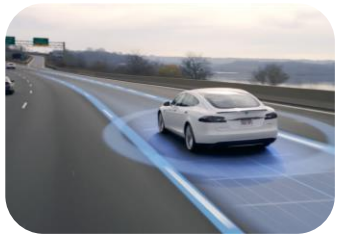
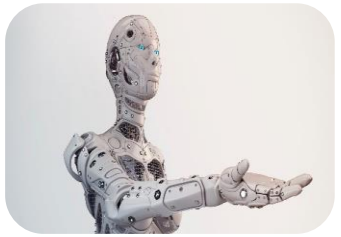
Fatemeh Asgarinejad, Justin Morris, Tajana Rosing, Baris Aksanli
University of California San Diego and San Diego State University

SAN DIEGO STATE UNIVERSITY
College of Engineering

UC San Diego

Challenges of Machine Learning

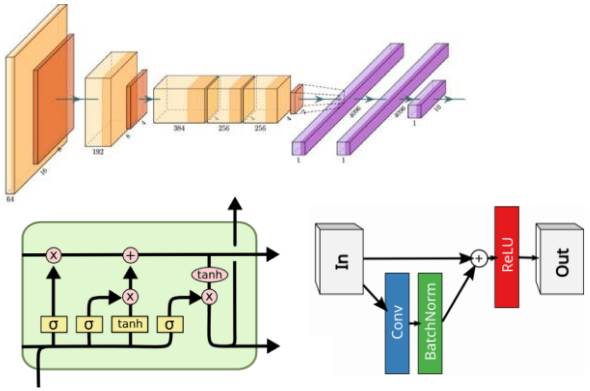
Machine learning is everywhere

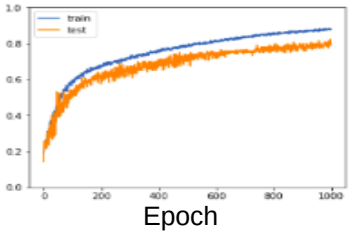



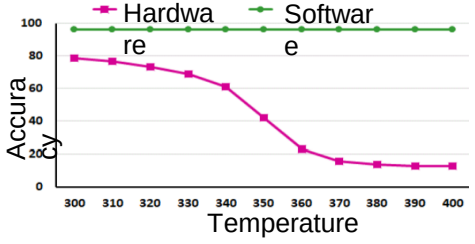


But there are challenges associated with ML:

- Large models
  - Millions of parameters, billions of operations
- Complex to understand and tune
  - Even more complex in case of heterogeneous data
- Needs a lot of data and iterations to train
- Highly vulnerable to error and noise

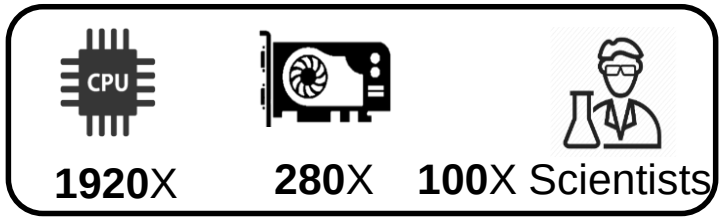


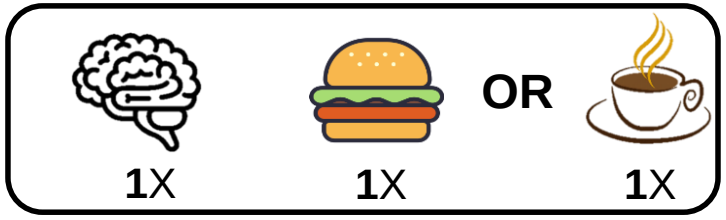




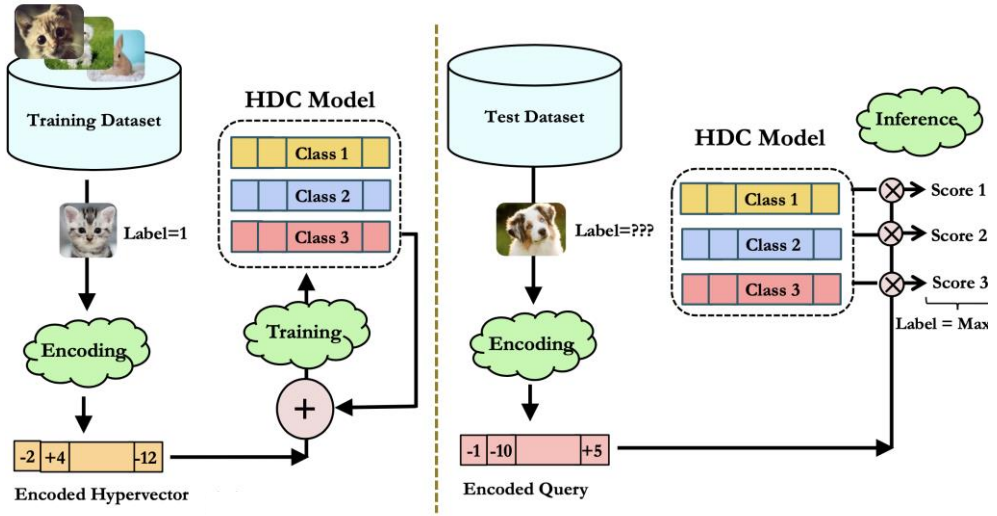
Hyperdimensional Computing (HDC)

- Biological brains, to this day, stand as the most superior platforms for cognitive abilities.





- Similar to brain, HDC is based on neural (distributed) representation of data
- HD Computing uses simple operations, is parallelizable, and robust to noise
- Data is mapped from its original representation $x \in n$ to an *HD representation* $\mathcal{H}$ under some encoding function φ ($\varphi: x \rightarrow \mathcal{H}$)



HDC Challenges and Related Works

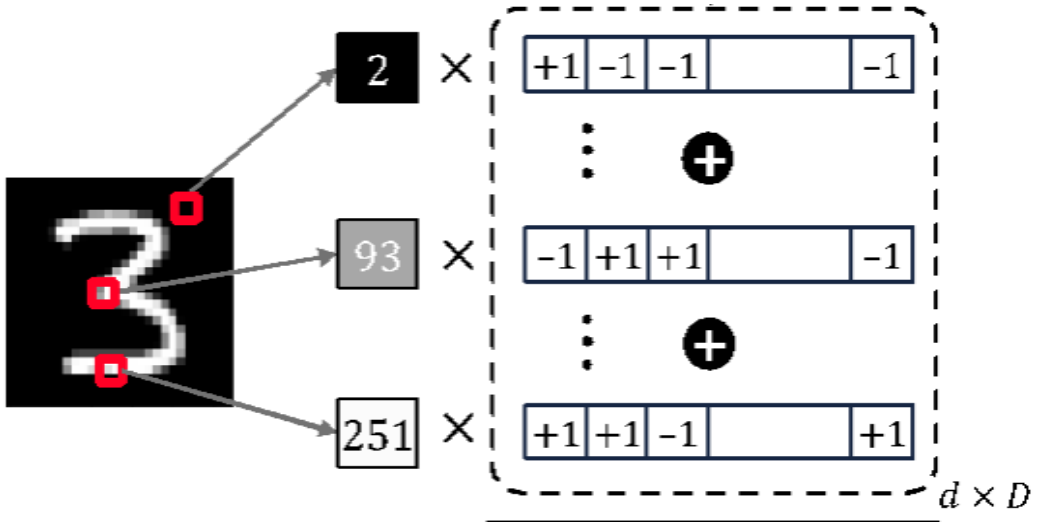
Challenge: There exists an accuracy gap between NN-based models and HDC

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

- Extract the features using a NN and then a HDC head for learning and classification [Dutta'22, Nazemi'22, Poduval' 21, Imani'22, Duan'21]
  - Jeopardize the robustness of HDC and create performance bottleneck with computation of features
- Propose novel encoding or training [Cano'21, Zuo'21, Imani'20, Khaleghi'22, Imani'21]
  - Exhibit increased operational complexity and demonstrate inferior accuracy under quantization or with small vector dimensions
- Employ neural networks to train HDC models [Duan'22, Yu'22]
  - Require complex operations and update all parameters for each sample

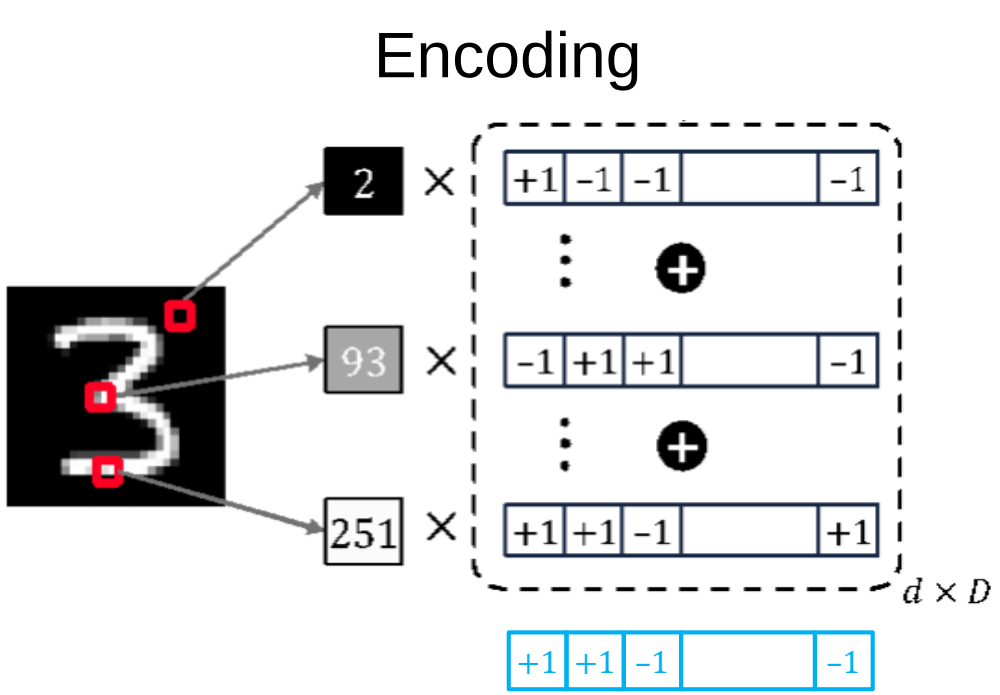
Projection-Based Encoding

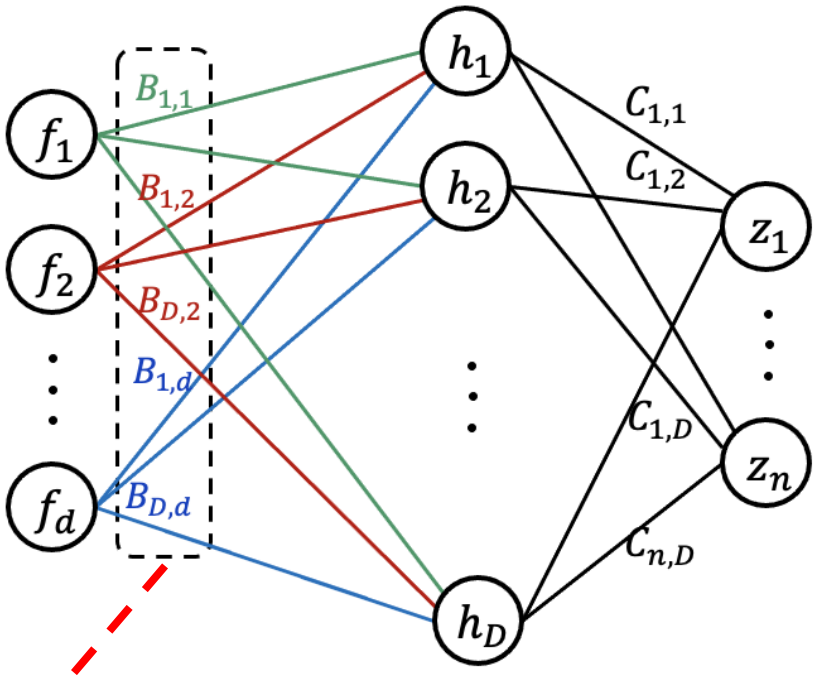
Encoding



$$[f_1 \quad f_2 \quad f_3 \quad \cdots \quad f_n]_{1\times n} \times \begin{bmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,D} \\ B_{2,1} & B_{2,2} & \cdots & B_{2,D} \\ B_{3,1} & B_{3,2} & \cdots & B_{3,D} \\ \vdots & \vdots & \vdots & \vdots \\ B_{n,1} & B_{n,2} & \cdots & B_{n,D} \end{bmatrix}_{n\times D} = [H_1 \quad H_2 \quad H_3 \quad \cdots \quad H_n]_{1\times D}$$

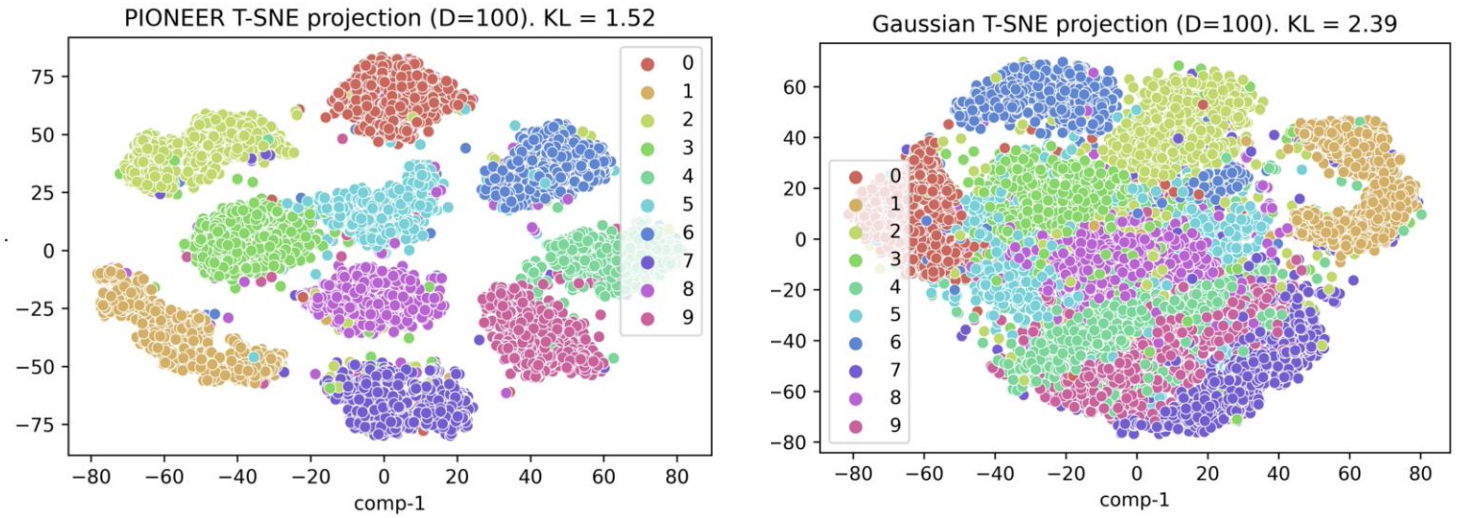
Proposed Method: Learning the Projection Matrix



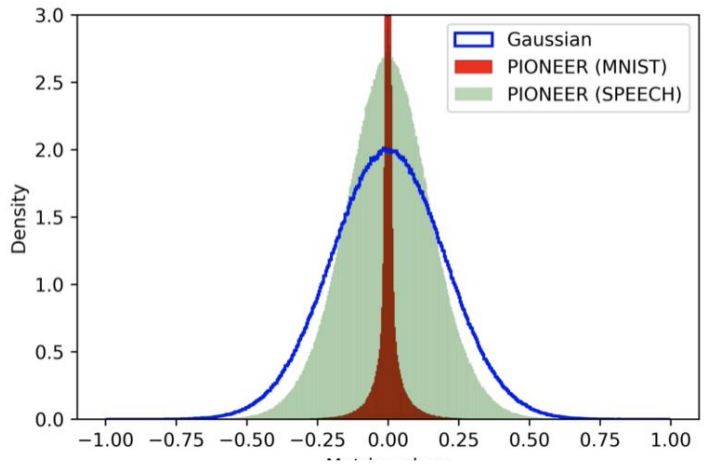


$$[f_1 \quad f_2 \quad f_3 \quad \cdots \quad f_n]_{1\times n} \times \begin{bmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,n} \\ B_{2,1} & B_{2,2} & \cdots & B_{2,n} \\ B_{3,1} & B_{3,2} & \cdots & B_{3,n} \\ \vdots & \vdots & \vdots & \vdots \\ B_{D,1} & B_{D,2} & \cdots & B_{D,n} \end{bmatrix}^T_{n\times D} = [H_1 \quad H_2 \quad H_3 \quad \cdots \quad H_n]_{1\times D}$$

Discriminative Quality of PIONEER between Classes



t-SNE plots of MNIST (D = 100)



Distribution of the learned matrix elements

- PIONEER better learns to distinguish between classes
- PIONEER calibrates the projection matrix based on the underlying data characteristics in contrast to random matrix values in Nonlinear method (based on Gaussian matrix)

Row-sparse Binary Projection

Floating Point Projection Matrix

$$\text{sign}\left(\begin{bmatrix} 0.53 & -0.82 & -0.04 & 0.12 & -0.44 & -0.77 & 0.22 & 0.85 \\ -0.90 & -0.15 & -0.62 & 0.48 & -0.43 & -0.25 & -0.17 & -0.58 \\ 0.17 & 0.99 & 0.97 & 0.36 & -0.22 & 0.54 & 0.75 & 0.21 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.86 & -0.58 & 0.50 & 0.52 & -0.33 & -0.71 & 0.58 & 0.06 \end{bmatrix} \times \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_d \end{bmatrix}\right) = \begin{bmatrix} H_1 \\ H_2 \\ H_3 \\ \vdots \\ H_D \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & 0 & 0 & 0 & -1 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & +1 & 0 & 0 & 0 & +1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

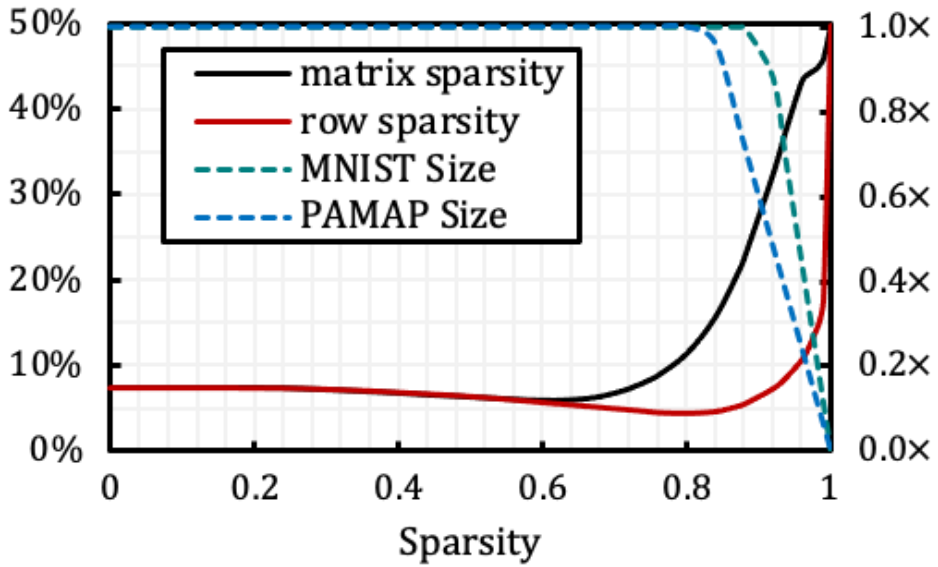
Sparse binary Matrix

$$\begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 & 0 & +1 \\ -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & +1 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ +1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

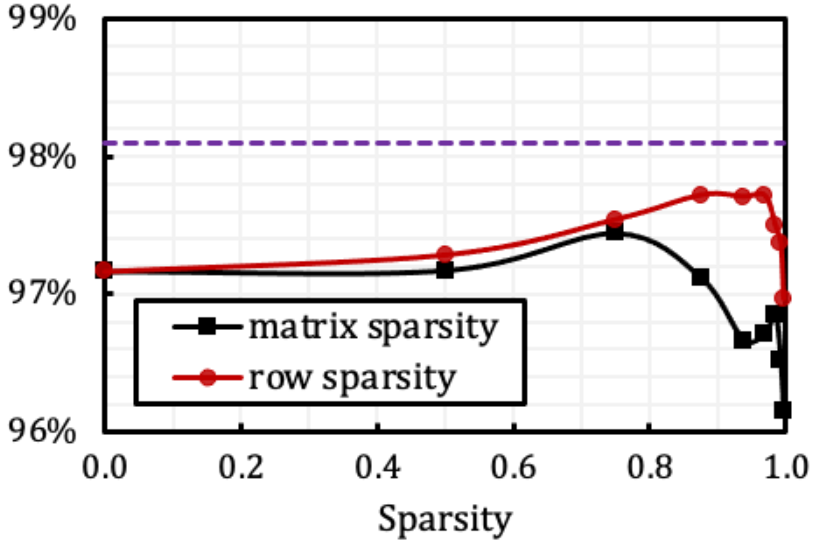
Row sparse matrix

- Keeping the maximum elements of each row, we incur minimal discrepancy to the encoding output (with respect to the dense, floating point matrix)
- Same number of non-zero elements in rows results in better compression

Row-sparse Binary Projection



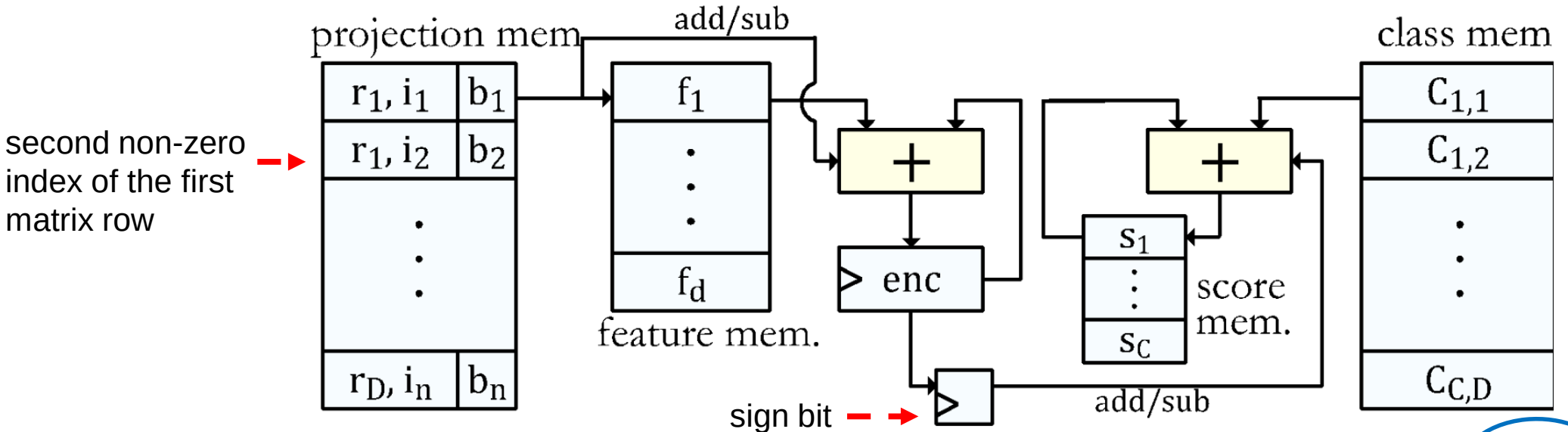
- Row-level sparsification leads to significantly lower encoding bit-change compared to matrix-level sparsification
- MNIST at 91% and PAMAP at 83% sparsity compensate the index overhead and start to save memory compared to dense binary matrix.



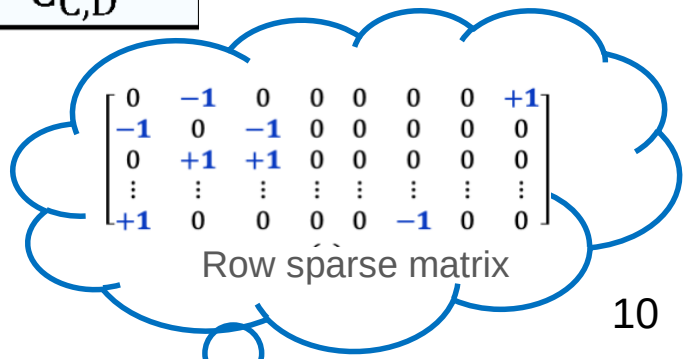
- The accuracy of MNIST with sparse binary projection drops less than the matrix sparsity

FPGA Implementation

- The simplicity of the projection encoding along with the binarization and row-level sparsification allows us to implement the encoding and inference using the below compact datapath



Datapath for generating one encoding element



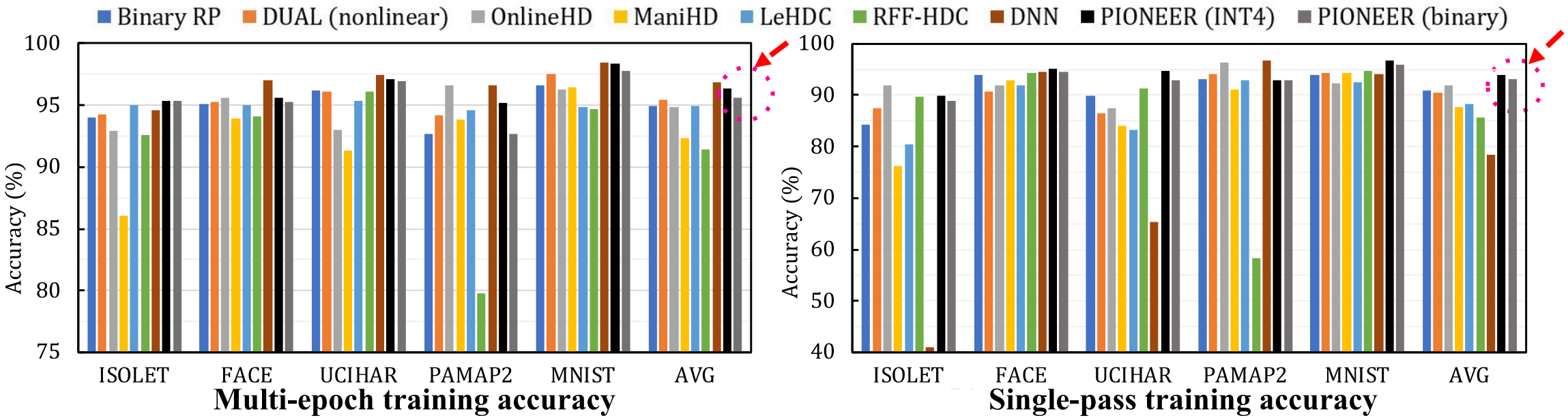
Experimental Setup

- Datasets

Dataset	# features	# classes	Train size	Test size	Description
ISOLET	617	26	6,238	1,559	Voice recognition
MNIST	784	10	60,000	10,000	Handwritten digits recognition
CIFAR-10	1024	10	50,000	10,000	Vision dataset
UCIHAR	561	12	6,213	1,554	Human activity recognition
PAMAP	75	5	611,142	101,582	Activity recognition
FACE	608	2	522,441	2,494	Face recognition

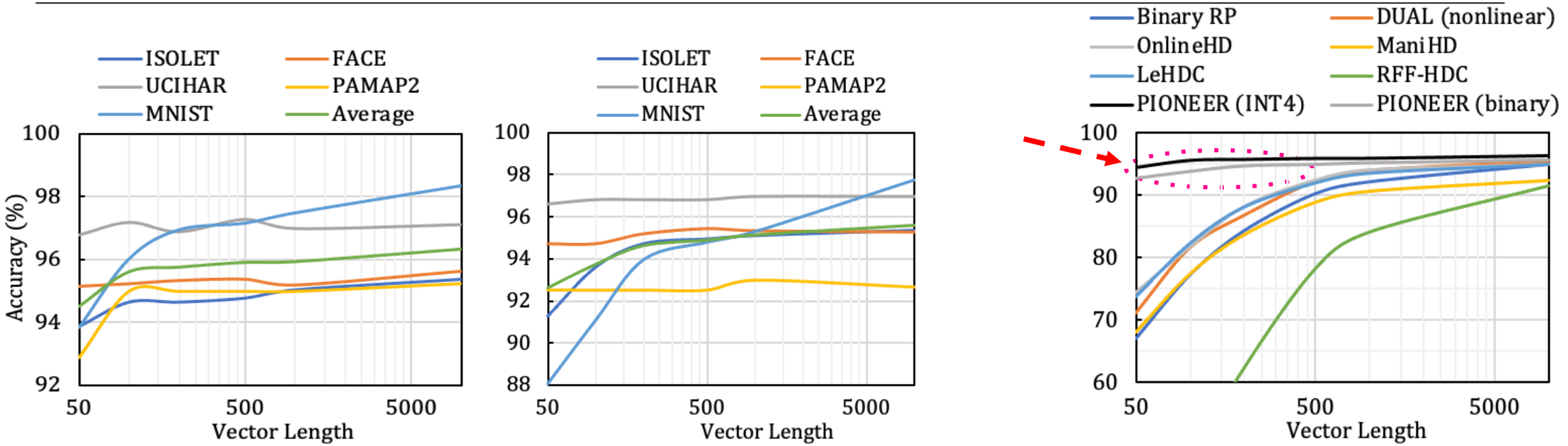
- Baseline Comparison: Binary RP [Kanerva'09], Nonlinear encoding [Imani'20], OnlineHD [Imani' 21], ManiHD [Zou'21], LeHDC [Duan'22], RFF-HDC [Yu'22], DNN [Liaw'18]
- Hardware Setup:
  - **DNN Baseline Implementation**: we used DNNWeaver to generate the Verilog code based on the optimized parameters
  - **Hardware Implementation**: The designs were implemented using Vivado HLS on a Xilinx Kintex-7 Kit featuring an XC7K325T FPGA, targeted at a frequency of 100 MHz
  - **Power Estimation**: To acquire power estimates, we used Xilinx Power Estimator (XPE)

Results: Accuracy



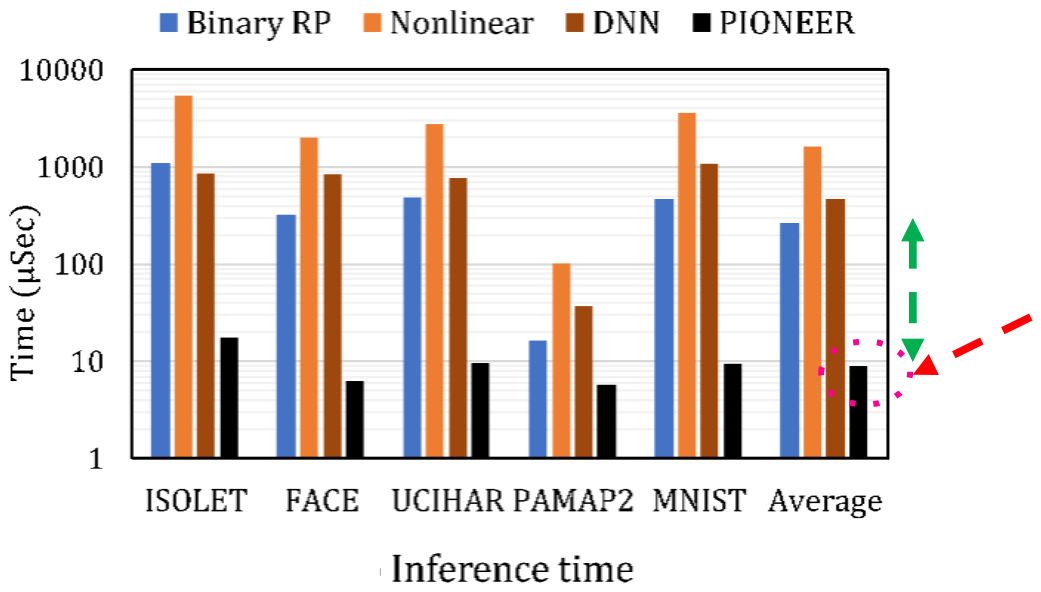
- In multi-pass training: PIONEER with INT4 (binary) matrix obtains an average accuracy of **96.33%** (**95.61%**)
- In single-pass training: PIONEER outperforms best baseline (OnlineHD[10] which is specifically optimized for single-pass training) by **1.92%**

Results: DownScalability

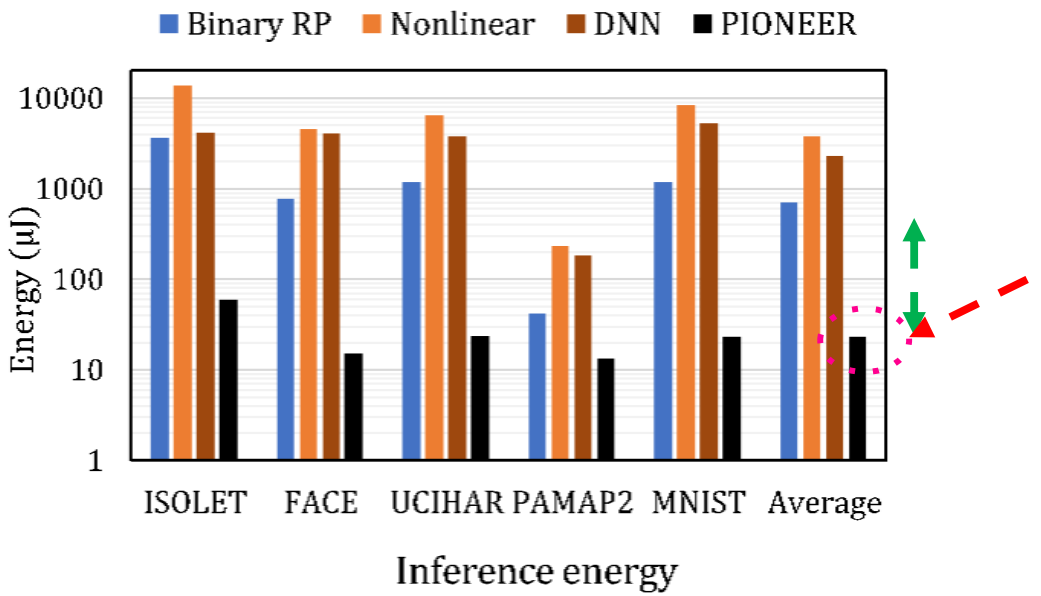


- At vector length of **100** (**50**), INT4 PIONEER achieves an average accuracy of **95.61%** (**94.50%**) among all the benchmarks (only 0.72% (1.8%) compared to D = 10k)
- In binary PIONEER vs. random binary matrix projection at $D = 50$, PIONEER shows a **25.7%** higher accuracy, emphasizing the importance of learned projection

Performance



Inference time



Inference energy

- PIONEER achieves **29×**, **179×** and **52×** faster inference time than the baseline random projection, nonlinear encoding and DNN respectively
- PIONEER saves **30×** , **165×** and **98×** energy over binary random projection, nonlinear and DNN

Conclusion

- We present PIONEER to close the accuracy gap between HD and NN-based methods while retaining the simplicity of HD learning and inference
- PINEER yields **18.3%** improvement in accuracy compared to state-of-the-art when $\boldsymbol{D = 50}$
- In single-pass (multi-pass) training with $\boldsymbol{D = 10k}$, PIONEER outperforms best baseline by **1.92%** (**0.86%**) compared to best HD-based method while yielding comparable accuracy to NN
- With vector lengths of as small as $\boldsymbol{D = 100}$, PIONEER achieves an average accuracy of **94.8%** (1.2% drop compared to when $D = 10k$), while previous works drop below 82%
- With $10k$ vectors, our FPGA implementation achieves a **179×** performance improvement and **165×** energy efficiency gain over state-of-the-art HDC encoding with comparable accuracy

References

[1] P. Kanerva, “Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors,” Cognitive computation, 2009

[2] P. Poduval, Z. Zou, H. Najafi, H. Homayoun, and M. Imani, “Stochd: Stochastic hyperdimensional system for efficient and robust learning from raw data,” in ACM/IEEE Design Automation Conference (DAC), pp. 1195–1200, IEEE, 2021.

[3] Ge et al, “Classification using hyperdimensional computing: A review,” IEEE Circuits and Systems Magazine, 2020.

[4] Cano et al, “Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system,” in DATE, 2021

[5] Duan et al, “Lehdc: Learning-based hyperdimensional computing classifier,” arXiv, 2022.

[6] Zou et al, “Manihd: Efficient hyperdimensional learning using manifold trainable encoder,” in DATE, 2021.

[7] Imani et al, “Bric:Locality-based encoding for energy-efficient brain-inspired hyperdimensional computing,” in DAC, 2019.

[8] Imani et al, “Dual: Acceleration of clustering algorithms using digital-based processing in-memory,” in International Symposium on Microarchitecture, 2020.

[9] Rahimi et al, “Random features for large-scale kernel machines,” Advances in neural information processing systems, 2007.

[10] Buluc et al, “Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks,” in Annual symposium on Parallelism in algorithms and architectures,, 2009.

[11] B. Khaleghi, J. Kang, H. Xu, J. Morris, and T. Rosing, “Generic: highly efficient learning engine on edge using hyperdimensional computing,” in IEEE Design Automation Conference, pp. 1117–1122, 2022.

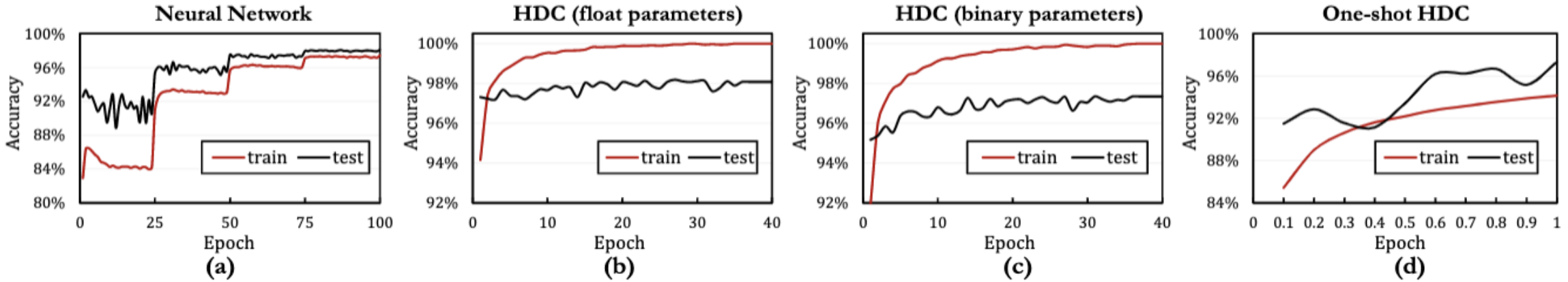
[12] A. Dutta, S. Gupta, B. Khaleghi, R. Chandrasekaran, W. Xu, and T. Rosing, “Hdnn-pim: Efficient in memory design of hyperdimensional computing with feature extraction,” in Great Lakes Symposium on VLSI, pp. 281–286, 2022.

[13] M. Nazemi, A. Fayyazi, A. Esmaili, and M. Pedram, “Synergiclearning: Neural network-based feature extraction for highly-accurate hyperdimensional learning,” in International Conference on Computer-Aided Design, pp. 1–9, 2020.

[14] M. Imani, Z. Zou, S. Bosch, S. A. Rao, S. Salamat, V. Kumar, Y. Kim, and T. Rosing, “Revisiting hyperdimensional learning for fpga and low-power architectures,” in 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 221–234, IEEE, 2021.

[15]] T. Yu, Y. Zhang, Z. Zhang, and C. De Sa, “Understanding hyperdimensional computing for parallel single-pass learning,” in Advances in Neural Information Processing Systems, 2022.

Backup (Accuracy of PIONEER vs NN on MNIST)

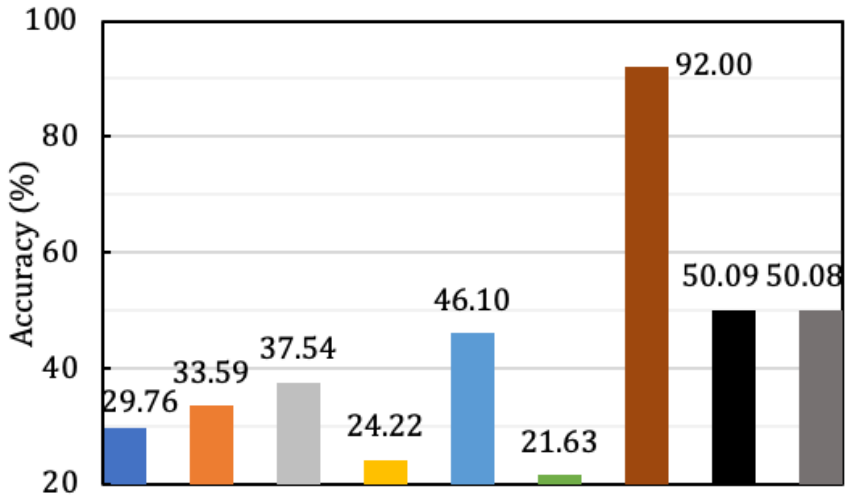


MNIST train and test accuracy of **(a)** HDC modeled as a neural network, **(b)** HDC with (float) learned parameters (PIONEER), **(c)** HDC with (binarized) learned parameters (PIONEER), and **(d)** one-shot HDC with learned parameters (PIONEER).

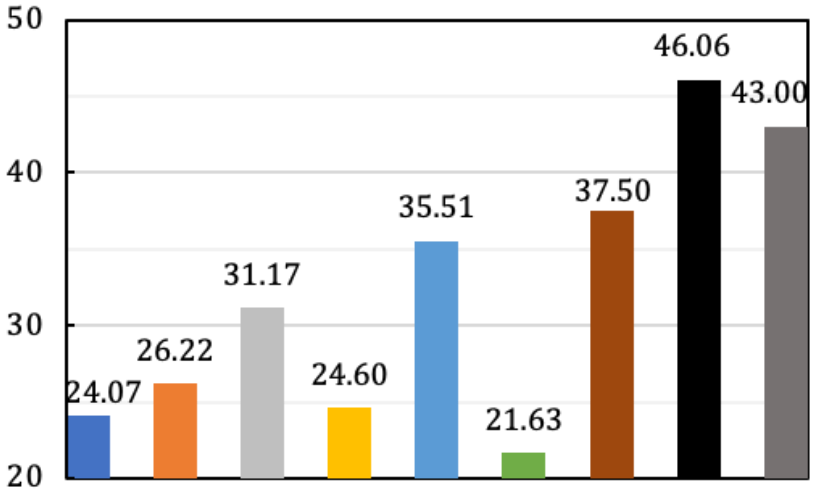
- NN model achieves a 98.09% accuracy after 100 epochs, while HDC using the projection constants learned by the same NN model achieves an accuracy of **98.08%** after **37 epochs**
- The **one-shot** (single-epoch) accuracy of HDC with the learned projection is **97.31%**

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

- For CIFAR10 dataset, PIONEER achieves 50.10% accuracy, which is 4.0% better than the best HDC baseline



Multi-epoch accuracy



Single-pass accuracy

Backup (Proposed Method: Learning the Projection Matrix)

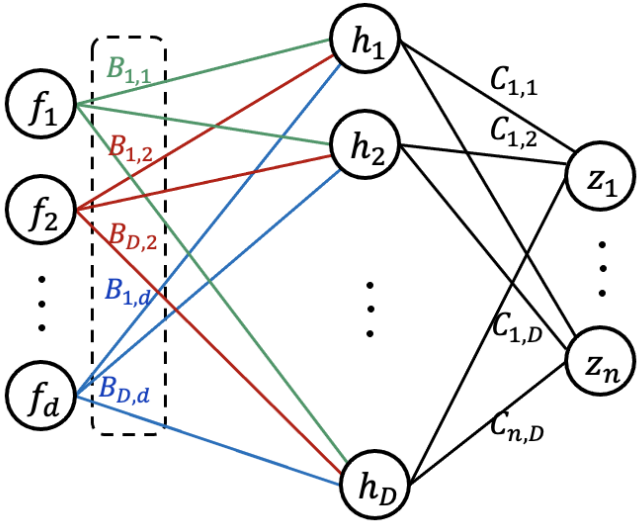
$$(1)\ C_{1,1}^{(t+1)} = C_{1,1}^{(t)} - \lambda \frac{\partial L}{\partial C_{1,1}}$$

$$(2)\ L = -\sum_{i=1}^{n} y_i \log \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}} = -\sum_{i}^{n} y_i \log s_i$$

$$(3)\ \frac{\partial L}{\partial C_{1,1}} = \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial C_{1,1}} \quad \text{and} \quad \frac{\partial L}{\partial z_i} = s_i - y_i \rightarrow \frac{\partial L}{\partial z_1} = s_1 - y_1$$

$$(4)\ \frac{\partial z_1}{\partial C_{1,1}} = \frac{\partial (h_1 . C_{1,1} + h_2 . C_{1,2} + \cdots + h_D . C_{1,D})}{\partial C_{1,1}} = h_1$$

$$\rightarrow (5)\ \frac{\partial L}{\partial C_{1,1}} = (s_1 - y_1)h_1 = \left(\frac{e^{z_1}}{\sum_{k=1}^{n} e^{z_k}} - y_1\right) h_1 = \alpha_1\ h_1$$



$$= B_{1,1}.f_1 + B_{1,2}.f_2 + \ldots + B_{1,n}.f_n = B_1.\mathcal{F}$$

$$z_1 = h_1.C_{1,1} + h_2.C_{1,2} + \ldots + h_D.C_{1,D} = h.C_1$$

$$z_n = h_1.C_{n,1} + h_2.C_{n,,2} + \ldots + h_n.C_{n,D} = h.C_d$$

$$h_D = B_{D,d}.f_1 + B_{D,2}.f_2 + \ldots + B_{D,d}.f_d = B_D.\mathcal{F}$$

19

Label is C1 ($y = [1,0,\cdots,0]$):

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

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

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