Bitwise Adaptive Early Termination in Hyperdimensional Computing Inference

Wei-Chen Chen, H.-S. Philip Wong and Sara Achour
Stanford University
{weichen9,hspwong,sachour}@stanford.edu

ABSTRACT

Hyperdimensional computing (HDC), a powerful paradigm for cognitive tasks, often demands hypervectors of high dimensions (e.g., 10,000) to achieve competitive accuracy. However, processing such large-dimensional data poses challenges for performance and energy efficiency, particularly on resource-constrained devices. In this paper, We present a framework to terminate bit-serial HDC inference early when sufficient confidence is attained in the prediction. This approach integrates a Naive Bayes model to replace the conventional associative memory in HDC. This transformation allows for a probabilistic interpretation of the model outputs, steering away from mere similarity measures. We reduce more than 70% of bits that need to be processed while maintaining comparable accuracy across diverse benchmarks. In addition, We show the adaptability of our early termination algorithm during on-the-fly learning scenarios.

KEYWORDS

Hyperdimensional Computing, Bayesian Network, Early Termination, Online Learning

1 INTRODUCTION

Hyperdimensional Computing (HDC) is a computational framework rooted in high-dimensional spaces, leveraging the properties of vector spaces to perform cognitive tasks [8]. HDC presents several advantages that render it an intriguing paradigm in computational and cognitive domains [16, 21]. One of its primary strengths lies in its robustness to noise and errors inherent in data, owing to the distributed and redundant nature of high-dimensional hypervectors [11, 17]. Additionally, its ability to perform computations using simple and parallelizable operations (e.g., bitwise XOR) offers advantages in terms of computational speed and scalability [9, 13].

However, HDC encounters several challenges that affect its widespread adoption and effectiveness in practical applications so far. One primary challenge lies in the computational overhead associated with retrieving, processing and manipulating high-dimensional hypervectors. The need for handling and operating within spaces of thousands or even more dimensions demands substantial memory resources, counteracting the simple arithmetic operations of HDC algorithms, especially on resource-constrained devices [4, 10, 12].

Numerous techniques have emerged to mitigate the processing load in HDC. Hypervector pruning [7, 19] statically reduces the number of bits processed. However, its applicability remains limited to specific hypervector encodings. For example, dimensional-wise pruning isn't feasible when the encoding process involves permutations because there's no assurance that random inputs won't engage those pruned dimensions. On the other hand, model compression methods [14] focus on compressing learnt class hypervectors, offering computational savings mainly during the similarity check stage rather than the encoding phase—known to be the bottleneck in HDC [9]. Early termination strategies [1, 15, 20] involve processing hypervectors in chunks and comparing intermediate results with a predetermined similarity threshold. When this threshold is reached, fewer bits require processing, enabling early inference termination. However, the challenge lies in determining the threshold value, often relying on heuristics, varying across classes, and necessitating multiple retraining iterations to optimize the threshold for each class individually. Moreover, these methods lack support for newly learned classes in online learning settings which is one of the main advantages of HDC [5, 6].

In this paper, we present BAET, a bitwise adaptive early termination technique, for efficient HDC. The main contribution of the paper are listed below:

- We present a HDC + Naive Bayes hybrid model architecture. This hybrid model architecture shifts the output paradigm from measuring distances between hypervectors to generating probabilities. This transition to probability outputs not only eliminates the need for recurrent retraining but also

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3656532

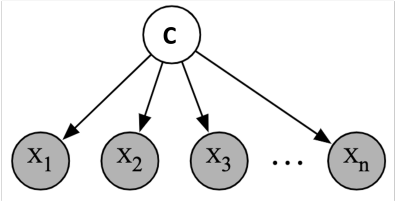

Figure 1: Naive Bayes Model with N independent feature nodes X_i and a class node Y.

offers a principled approach for determining the threshold value for early termination.

- We introduce an early termination algorithm derived from the hybrid model, which shows an reduction of over 70% in the average number of processed bits, maintaining consistent accuracy across diverse benchmark datasets.
- We showcase the adaptability of our early termination algorithm, proving its efficacy in accommodating new samples during on-the-fly learning scenarios.

2 HYPERDIMENSIONAL COMPUTING

Classification in HDC essentially involves three steps : encoding, training and similarity check.

Encoding is the foundation of HDC, wherein input data I_i (an image, audio, etc.), undergo a transformation into a binarized high-dimensional vectors, $H_i \in \{-1,+1\}^D$, where D ≈ 10,000. The transformation includes manipulation of base hypervectors $B_k \in \{-1,+1\}^D$ where each dimension in B $\overset{\text{iid}}{\sim} Bernoulli(0.5)$. The well-defined high-dimensional vector arithmetic ensures that the similarity, measured in Hamming distance, between H_i and H_j corresponds to the affiliation of I_i and I_j to the same class, and conversely, they exhibit dissimilarity if belonging to different classes.

Training class hypervectors H_c are constructed through the aggregation of encoded input data H_i belonging to the same class. These class hypervectors serve as repositories of collective class knowledge.

$$H_c = \sum_i H_i, \forall \text{ i s.t. } I_i \in \text{class c} \tag{1}$$

Similarity check calculate the distance (δ) between encoded input data H_i with the learnt class hypervectors H_c that are stored in the associative memory (AM). For real-valued H_c, $\delta(H_i, H_c) = cos(H_i, H_c)$. For binarized H_c, the distance calculation reduces to Hamming distance. Finally, the class with the closest distance will be picked as the prediction.

3 BAET CLASSIFIER

The BAET classifier is a probabilistic hypervector classifier that categorizes hypervector-encoded inputs into $C_0 ... C_k .. C_M$ classes. Provided an N-dimensional bipolar hypervector $H = [h_0, h_i, .. h_N]$ that encodes an input datum (e.g., image, audio), the classifier identifies the class C_{cls} that has the maximum conditional probability given the hypervector X:

$$C_j = argmax_{C_k}\ P(C_k|X=H)$$

$$where\ P(C_k|X=H) = \frac{P(C_k)\prod P(x_i|C_k)}{P(X=H)}$$

The BAET classifier returns both the highest likelihood C_j for the hypervector input and the probability the input is sampled from the selected class, $P(C_i|X=H)$. The returned probability captures the model's confidence that the input X belongs to the assigned class C_j. This confidence value is used by BAET's inference and training optimizations to reduce the amount of information processed without sacrificing accuracy. Section 3.1 overviews the derivation of the conditional probabilities, and Section 4 presents the inference optimization, and Section 5 presents the training optimizations employed by BAET.

3.1 Derivation of Classifier

The BAET classifier requires the evaluation of the conditional probability $P(C = C_k|X = H)$ to classify hypervector inputs. BAET uses a Naive-Bayes model (Figure 1) to compute the conditional probability, which imposes strong independence assumptions on the model inputs.

We next derive our usage of the Naive Bayes-based probabilistic model from Bayes' theorem. Using Bayes' theorem, the conditional probability of a class C_k given an input H can be expanded to the following relation:

$$P(C=C_k|X=H) = \frac{P(C=C_k)P(X=H|C=C_k)}{P(H)}$$

We next leverage HDC's independence properties to simplify the modeling of the conditional probability. Recall in hyperdimensional computing, any bits x_i in the base hypervectors B are drawn from independent and identically distributed (i.i.d.) Bernoulli distributions. Here we assumed the HDC encoding operations do not introduce dependencies between bits, thus any two bits x_i, x_j in hypervector X can still be assumed to be independent from one another. The hypervector conditional probability $P(X|C_k)$ can then be rewritten as the product of per-bit conditional probabilities $\prod P(x_i|C_k)$:

$$P(C_k|X) = \frac{P(C_k)\prod P(x_i=h_i|C_k)}{P(H)} \tag{2}$$

Therefore, the above conditional probability of a class C_k for a given hypervector H is computed from the probability of the class $P(C_k)$, the probability of a bit x_i being value h_i conditioned on C_k, $P(x_i = h_i|C_k)$. The probabilities and conditional probabilities in BAET's Naive Bayes model are populated with the training algorithm presented in Section 3.2.

3.2 Naive Training Method

Learning a BAET model is through maximum likelihood estimation (MLE). We present the basic formulations for computing the prior and conditional probabilities from training data.

3.2.1 Computing The Prior. The algorithm derives the class probability $P(C=C_k)$, or the prior, for each class k by computing the fraction of the dataset assigned to class C_k:

$$P(C_k)=\frac{|D_k|}{|D|}$$

3.2.2 Computing the Conditional Probabilities. The per-bit conditional probability $P(x_i=1|C_k)$ and $P(x_i=-1|C_k)$ for a given class C_k is computed by counting the fraction of training hypervectors belonging to class C_k that have the bit value 1 in bit position x_i respectively:

$$P(x_i=1|C_k)=\frac{|D_k(x_i=1)|}{|D_k|}$$

The $P(x_i=-1|C_k)$ probability can then be trivially derived from $P(x_i=1|C_k)$ since the hypervector H_i is bipolar and only supports two kinds of values:

$$P(x_i=-1|C_k)=1-P(x_i=1|C_k)$$

4 PROGRESSIVE INFERENCE

BAET uses conditional probability information to terminate inference once the best-matching class has exceeded a probability threshold P_{thr}. This optimization enables the BAET classifier to produce a classification after only processing a small number of hypervector bits. The structure of the conditional probability equation (Eqn 2) enables bit-serial processing and classification of input hypervectors.

4.1 Basic Algorithm

The algorithm maintains a running conditional probability $Q_i(H,C_k)$ for each class C_k, where Q_i is the conditional probability of C_k conditioned on the leading i bits of H.

4.1.1 Initialization. Initial conditional probability $Q_1(H,C_k)$ is computed from the class probability $P(C_k)$ and the probability of the first bit having value h_1 given C_k:

$$Q_1(H,C_k)=P(C_k)\cdot P(x_1=h_1|C_k)$$

4.1.2 Conditional Probability Update. At each step i of the algorithm, the conditional probabilities $Q_i(H,C_0)..Q_i(H,C_M)$ are computed from the previous conditional probabilities $Q_{i-1}(H,C_0)..Q_{i-1}(H,C_M)$ to incorporate bit h_i of H:

$$Q_i(H,C_k)=Q_{i-1}(H,C_k)\times P(x_i=h_i|C_k)$$

4.1.3 Early Termination and Classification. After the probabilities are updated at step i, the algorithm tests if any of the classes C_k have exceeded the conditional probability threshold P_{thr}. If there some class C_j with a probability $Q_i(H,C_k)$ that exceeds P_{thr} the inference algorithm *terminates early* and returns C_j. If the end of the input hypervector is reached ($i=N$), the class C_j with the highest conditional probability class is returned.

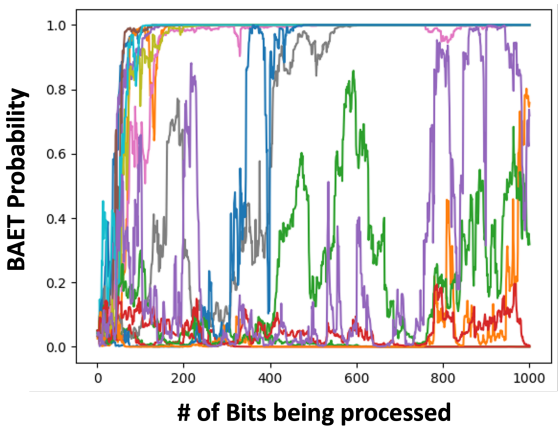

Figure 2: Probability trace of different inputs of a same class with BAET model on ISOLET dataset.

4.1.4 Example: Progressive Inference – ISOLET Dataset. An illustration of such a "conditional probability trace" involving streaming input dimensions is presented in Figure 2. Each trajectory is the running conditional probability of the labeled class for a distinct input hypervector. Notably, certain inputs yield high probabilities with minimal bits processed, while intriguingly, for some inputs, processing additional bits does not contribute to refining the final prediction and even deteriorate it. These observations serve as a motivation to implement an bit-serial early termination strategy where inference ceases once a predetermined confidence threshold is attained.

4.2 Inference as a MVM

In practical implementations, the conditional probability calculations within the progressive inference algorithm can be executed in larger granularities, such as k bits. This approach can equivalently be modeled as a matrix-vector multiplication problem. Initially, the logarithm of the conditional probability is taken. Consequently, the product of probabilities transforms into the summation of log-probabilities:

$$\log(\prod P(x_i=h_i|C_k))=\sum\log(P(x_i=h_i|C_k))$$

And the matrix vector multiply formulation is :

$$\begin{bmatrix} \log P_{i1}(1), & \log P_{i1}(-1), & ..., & \log P_{(i+k)1}(-1) \\ \log P_{i2}(1), & \log P_{i2}(-1), & ..., & \log P_{(i+k)2}(-1) \\ \vdots & \vdots & \vdots & \vdots \\ \log P_{im}(1), & \log P_{im}(-1), & ..., & \log P_{(i+k)m}(-1) \end{bmatrix} \begin{bmatrix} \mathbb{1}(h_i=1) \\ \mathbb{1}(h_i=-1) \\ \vdots \\ \mathbb{1}(h_{i+k}=-1) \end{bmatrix}$$

where $\log P_{im}(\pm1)=\log(P(x_i=\pm1|C_m))$ and $\mathbb{1}$ is the indicator function. Recall in HDC, the similarity check is also a matrix-vector multiplication problem, computed using cosine distance. This reformulation aligns the inference process in BAET to be as straightforward as inference in HDC, thereby simplifying the computational complexity and facilitating a comparable ease of execution between the two models.

Algorithm 1 Progressive BAET learning (p-BAET)

1: Dataset D, threshold P_{thr}
2: **for** $D_k = [H_0,...,]$ in $D = \{D_0...D_M\}$ **do**
3: **for** $H = [h_0,...,h_N]$ in D_k **do**
4: $P(C_k)$ = PriorUpdate(C_k)
5: $Q_1(H,C_k)$ = ProgInfer($P(x=h_1|C_k)$,$P(C_k)$)
6: **for** i in 1...N **do**
7: **if** $Q_i(H,C_k) \geq P_{thr}$ **then**
8: next input
9: **end if**
10: $P(x=1|C_k)$ = CondProbUpdate($P(x=1|C_k)$, h_i)
11: $Q_{i+1}(H,C_k)$ = ProgInfer($P(x=h_{i+1}|C_k)$, $Q_i(H,C_k)$)
12: **end for**
13: **end for**
14: **end for**

5 PROGRESSIVE TRAINING ALGORITHM

Beyond the naive training method, BAET can use a novel training algorithm that uses conditional probability information to reduce the amount of training data needed to populate the BAET classifier.

Algorithm 1 presents the progressive training algorithm, which we termed p-BAET. The training algorithm accepts as input a training dataset $D = D_0,..D_k,..D_m$ and derives the class probability ($P(C=C_k)$) and the $2\times N$ per-bit conditional probabilities ($P(x_i=1|C=C_k)$ and $P(x_i=-1|C=C_k)$ for each class $C_0..C_k..C_m$. While typically, the conditional probabilities are derived by processing every bit of each training input, p-BAET uses confidence information so that only a subset of input hypervector bits need to be processed. Each dataset D_k is comprised of a sequence of N-bit hypervector-encoded training inputs H_i with a label C_k. The C_k model for each class is trained separately.

5.1 Incremental Updates

p-BAET naturally supports online learning capability to incorporate inputs on the fly easily. Given an input $H \in D_k$ belonging to class C_k, the algorithm incrementally updates the prior $P(C_k)$ and per-bit conditional probabilities $P(x_j=h_j|C_k)$ to incorporate the input information. Because the model parameters can be incrementally updated, the training algorithm can be deployed as an online algorithm. This streamlined approach avoids the necessity of revisiting the entire training set or employing gradient-based learning techniques.

5.1.1 Incrementally Updating the Prior. Line 4 incrementally updates the prior $P(C_k)$ to incorporate new training inputs. Incremental updates on the prior are accomplished by maintaining a running count of $|D_k|$ and $|D|$.

$$P(C_k) = \frac{|D_k|+1}{|D|+1}$$

$$P(\neg C_k) = \frac{|D_k|}{|D|+1}$$

5.1.2 Incrementally Updating the Conditional Probability. Line 10 incrementally updates the condition probability of bit h_j of H given class C_k. Incremental updates on the conditional probabilities are accomplished by maintaining a running count of $|D_k|$ and $|D_k(x_i=1)|$:

$$P(x_i=1|C_k) = \frac{|D_k(x_i=1)|+1}{|D_k|+1} \; if h_i = 1$$

$$P(x_i=1|C_k) = \frac{|D_k(x_i=1)|}{|D_k|+1} \; if h_i = -1$$

5.2 Partial Processing of Training Inputs

Typically, the conditional probability is derived by processing each bit in a hypervector input $h_0...h_i...h_N$. Lines 6-12 of the algorithm terminate training a hypervector H with a label C_k early if the class C_k conditional probability of H exceeds a probability threshold P_{thr}.

5.2.1 Progressive Inference. The early truncation optimization uses the conditional probability, obtained with progressive inference, to determine how many bits to process. Line 5 instantiates the running conditional probability, and line 11 computes the running conditional probability $Q_i(H,C_k)$ from $Q_{i-1}(H,C_k)$ and the conditional probability $P(h_i|C=C_k)$.

5.2.2 Early Truncation. If the conditional probability $Q_i(H,C_k)$ exceeds P_{thr}, then the training algorithm stops training on H early without processing bits $i+1..N$. If the running conditional probability $Q_i(H_i,C_k)$ is less than P_{thr}, BAET updates the conditional probability $P(x_i=h_i|C_k)$ using the ith bit value h_i from the hypervector H. The number of processed bits i is incremented, and the process is repeated from the inference step.

6 EXPERIMENT RESULT AND DISCUSSION

Our experimentation involves three diverse datasets: MNIST[3] for image classification, ISOLET[2] for audio signal classification, and EMG[18] for biosignal classification. In Table 1, we present the baseline accuracies achieved by both HDC (in black) and BAET models trained in the naive way (in red) across various hypervector dimensions (D), excluding the application of early termination technique. The result shows that Vanilla BAET model, without fine tuning and early termination, improves accuracy about 2% across different benchmarks at all dimension. We attribute this improvement to MLE learning that minimize the Kullback-Leibler (KL) divergence.

6.1 Early Termination

In Table 2, we present the performance metrics of BAET, including accuracy and the average percentage of processed

Dimension	200	500	2000	5000
ISOLET	62.9 / 64.7	74.7 / 77.4	82.8 / 84.4	84.2 / 84.7
MNIST	65.4 / 67.4	75.5 / 77.5	81.5 / 83.2	81.5 / 82.9
EMG	96.2 / 99.4	97.5 / 100	98.7 / 100	99.4 / 100

Table 1: Baseline accuracy for HDC and BAET model.

bits, employing the early termination technique. Comparing HDC and BAET models both trained with a dimension of 2000, BAET consistently surpasses HDC, exhibiting better accuracy and processing fewer bits at high confidence thresholds. Notably, at a confidence threshold of 0.999, BAET processes only 4.7%, 30.5%, and 37.0% of bits for EMG, MNIST, and ISOLET datasets, respectively. Additionally, BAET enables facile exploration of the accuracy-efficiency trade-off by tuning the confidence threshold. Lowering the confidence threshold compromises accuracy but also reduces the number of processed bits. This feature proves advantageous, particularly on resource-constrained devices, offering flexibility when optimizing for metrics beyond sheer accuracy.

Dataset	EMG	MNIST	ISOLET
HDC	98.7	81.5	82.8
0.9	96.2 / 3.2%	73.6 / 9.9%	72.1 / 13.4%
0.95	96.2 / 3.5%	76.4 / 12.9%	76.6 / 17.4%
0.99	98.7 / 3.9%	80.0 / 20.2%	81.7 / 26.1%
0.995	98.7 / 4.1%	80.6 / 23.9%	82.4 / 29.9%
0.999	99.4 / 4.7%	81.7 / 30.5%	83.4 / 37.0%

Table 2: BAET model performance (black : accuracy, red : average % of bits processed) at different confidence threshold.

6.2 Online Learning Adaptation

Figure 3 illustrates the online learning capabilities of p-BAET using the ISOLET dataset. Two scenarios are depicted: starting with a p-BAET model trained initially with 20% (left) and 5% (right) of the training data, both at a dimension of 2000. The model is incrementally updated with Algorithm 1 as additional training data streams in, with different lines representing various confidence thresholds. The accuracy measurements on the test set are recorded. Interestingly, models initialized with 5% and 20% of the data converge to similar accuracies when subjected to more training data, showcasing the potential for incremental learning even from minimal initial data. In addition, observations indicate that higher thresholds yield better model learning due to the algorithm's propensity to halt updates when the model exhibits increased confidence in its predictions. However, this corresponds to an increase in the average number of processed bits, as demonstrated in Figure 4. The distribution plot in Figure 4 illustrates the number of bits processed upon reaching the threshold. A distinct rightward shift in the distribution is evident as the threshold value increases.

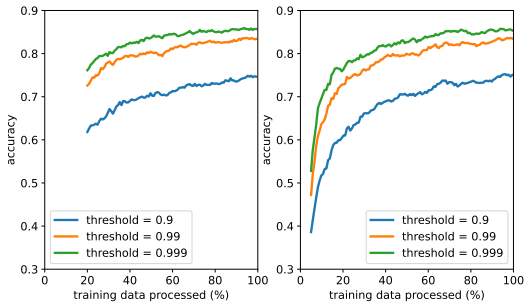

Figure 3: Progressive learning through p-BAET inference starts from model trained with 20% (left) and 5% (right) of training data.

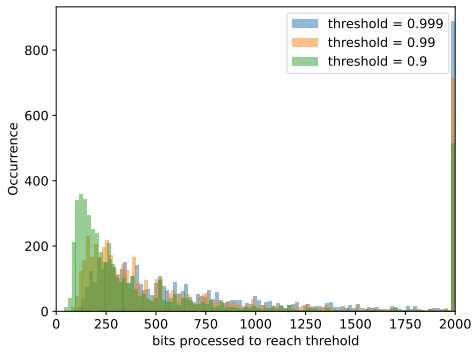

Figure 4: Distribution of bits processed with p-BAET.

In Table 3, a comparison among HDC, BAET, and p-BAET in terms of accuracy, average bits for training, and inference on the ISOLET dataset is presented. p-BAET only uses 43% of bits during training while deliver higher accuracy at the same time. This can be attributed to the distillation ability inherent in p-BAET learning, wherein the model learns until it attains sufficient confidence; beyond this point, additional bits do not enhance but may even diminish accuracy. Regarding inference, p-BAET exhibits a slight increase in the required bits. This discrepancy might arise from the fact that in naive BAET training, every dimension undergoes updates with all training data, implying that each bit carries more information than in p-BAET, where some dimensions are infrequently updated.

Model	HDC	BAET	p-BAET
Accuracy	82.8	84.4	85.6
Avg. bits for Training	100%	100%	43%
Avg. bits for Inference	100%	37%	38.9%

Table 3: Comparison of HDC, BAET and p-BAET.

7 FUTURE WORK

The symbiotic relationship between HDC and NB discussed in this paper paves the way for investigating hybrid model architectures. Beyond NB, the integration of probabilistic graphical

models (PGM), such as tree-augmented naive bayes, holds the potential to augment the capabilities of these hybrid models even further. Simultaneous exploration of PGM back-end classifier with HDC's front-end encoding strategies needs thorough investigation. These directions represent promising path for future research, offering opportunities to harness the combined strengths of HDC and PGM for enhanced performance and versatility in various applications.

8 CONCLUSION

This paper introduces BAET, a hybrid model architecture merging hyperdimensional computing with probabilistic graphical models. BAET facilitates a shift from similarity-based measures to a probabilistic interpretation, enabling confidence threshold-based early termination without relying on heuristics or requiring retraining. Leveraging this probabilistic output for inference significantly reduces the requisite number of processed bits. Additionally, an online learning algorithm p-BAET, utilizing BAET inference, is proposed, offering the potential for lightweight and efficient learning at the edge.

ACKNOWLEDGMENTS

This work is supported in part by SRC JUMP 2.0 PRISM and CHIMES Centers and member companies of the Stanford SystemX Alliance.

REFERENCES

[1] Yu-Chuan Chuang, Cheng-Yang Chang, and An-Yeu Andy Wu. 2020. Dynamic Hyperdimensional Computing for Improving Accuracy-Energy Efficiency Trade-Offs. In *2020 IEEE Workshop on Signal Processing Systems (SiPS)*. 1–5. https://doi.org/10.1109/SiPS50750.2020.9195216

[2] Ron Cole and Mark Fanty. 1994. ISOLET. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C51G69.

[3] Li Deng. 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142. https://doi.org/10.1109/MSP.2012.2211477

[4] Manuel Eggimann, Abbas Rahimi, and Luca Benini. 2021. A 5 W Standard Cell Memory-Based Configurable Hyperdimensional Computing Accelerator for Always-on Smart Sensing. *IEEE Transactions on Circuits and Systems I: Regular Papers* 68, 10 (2021), 4116–4128. https://doi.org/10.1109/TCSI.2021.3100266

[5] Alejandro Hernández-Cano, Namiko Matsumoto, Eric Ping, and Mohsen Imani. 2021. OnlineHD: Robust, Efficient, and Single-Pass Online Learning Using Hyperdimensional System. In *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*. 56–61. https://doi.org/10.23919/DATE51398.2021.9474107

[6] Michael Hersche, Edoardo Mello Rella, Alfio Di Mauro, Luca Benini, and Abbas Rahimi. 2020. Integrating Event-Based Dynamic Vision Sensors with Sparse Hyperdimensional Computing: A Low-Power Accelerator with Online Learning Capability. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design* (Boston, Massachusetts) *(ISLPED '20)*. Association for Computing Machinery, New York, NY, USA, 169–174. https://doi.org/10.1145/3370748.3406560

[7] Mohsen Imani, Sahand Salamat, Behnam Khaleghi, Mohammad Samragh, Farinaz Koushanfar, and Tajana Rosing. 2019. SparseHD: Algorithm-Hardware Co-optimization for Efficient High-Dimensional Computing. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 190–198. https://doi.org/10.1109/FCCM.2019.00034

[8] Pentti Kanerva. 2009. Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors. *Cogn Comput* 1 (2009), 139–159. https://doi.org/10.1007/s12559-009-9009-8

[9] Geethan Karunaratne, Manuel Le Gallo, Giovanni Cherubini, Luca Benini, Abbas Rahimi, and Abu Sebastian. 2020. In-memory hyperdimensional computing. *Nat Electron* 3 (2020), 327–337. https://doi.org/10.1038/s41928-020-0410-3

[10] Dehua Liang, Hiromitsu Awano, Noriyuki Miura, and Jun Shiomi. 2023. A Robust and Energy Efficient Hyperdimensional Computing System for Voltage-Scaled Circuits. *ACM Trans. Embed. Comput. Syst.* (sep 2023). https://doi.org/10.1145/3620671

[11] Dongning Ma, Tajana Šimunić Rosing, and Xun Jiao. 2023. Testing and Enhancing Adversarial Robustness of Hyperdimensional Computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42, 11 (2023), 4052–4064. https://doi.org/10.1109/TCAD.2023.3263120

[12] Alisha Menon, Daniel Sun, Melvin Aristio, Harrison Liew, Kyoungtae Lee, and Jan M. Rabaey. 2021. A Highly Energy-Efficient Hyperdimensional Computing Processor for Wearable Multi-Modal Classification. In *2021 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. 1–4. https://doi.org/10.1109/BioCAS49922.2021.9645008

[13] Fabio Montagna, Abbas Rahimi, Simone Benatti, Davide Rossi, and Luca Benini. 2018. PULP-HD: Accelerating Brain-Inspired High-Dimensional Computing on a Parallel Ultra-Low Power Platform. , Article 111 (2018), 6 pages. https://doi.org/10.1145/3195970.3196096

[14] Justin Morris, Mohsen Imani, Samuel Bosch, Anthony Thomas, Helen Shu, and Tajana Rosing. 2019. CompHD: Efficient Hyperdimensional Computing Using Model Compression. In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. 1–6. https://doi.org/10.1109/ISLPED.2019.8824908

[15] Justin Morris, Si Thu Kaung Set, Gadi Rosen, Mohsen Imani, Baris Aksanli, and Tajana Rosing. 2021. AdaptBit-HD: Adaptive Model Bitwidth for Hyperdimensional Computing. In *2021 IEEE 39th International Conference on Computer Design (ICCD)*. 93–100. https://doi.org/10.1109/ICCD53106.2021.00026

[16] Yang Ni, Mariam Issa, Danny Abraham, Mahdi Imani, Xunzhao Yin, and Mohsen Imani. 2022. HDPG: Hyperdimensional Policy-Based Reinforcement Learning for Continuous Control. (2022), 1141–1146. https://doi.org/10.1145/3489517.3530668

[17] Prathyush Poduval, Yang Ni, Yeseong Kim, Kai Ni, Raghavan Kumar, Rossario Cammarota, and Mohsen Imani. 2022. Adaptive Neural Recovery for Highly Robust Brain-like Representation. (2022), 367–372. https://doi.org/10.1145/3489517.3530659

[18] Abbas Rahimi, Simone Benatti, Pentti Kanerva, Luca Benini, and Jan M. Rabaey. 2016. Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*. 1–8. https://doi.org/10.1109/ICRC.2016.7738683

[19] Farhad Taheri, Siavash Bayat-Sarmadi, and Shahriar Hadayeghparast. 2022. RISC-HD: Lightweight RISC-V Processor for Efficient Hyperdimensional Computing Inference. *IEEE Internet of Things Journal* 9, 23 (2022), 24030–24037. https://doi.org/10.1109/JIOT.2022.3191717

[20] Farhad Taheri, Siavash Bayat-Sarmadi, and Hamed Rastaghi. 2023. PartialHD: Toward Efficient Hyperdimensional Computing by Partial Processing. *IEEE Internet of Things Journal* (2023), 1–1. https://doi.org/10.1109/JIOT.2023.3287316

[21] Ruixuan Wang, Xun Jiao, and X. Sharon Hu. 2022. ODHD: One-Class Brain-Inspired Hyperdimensional Computing for Outlier Detection. (2022), 43–48. https://doi.org/10.1145/3489517.3530395