

# PatBiNN: A 65 nm Processing-in-CAM Based BNN Implementation for Pathogen Genome Classification

Yuval Harary\*, Almog Sharoni\*, Esteban Garzón<sup>§\*</sup>, and Leonid Yavits\*

\*EnICS Labs Institute, Faculty of Engineering, Bar Ilan University, Ramat Gan 5290002, Israel

<sup>§</sup>Dep. of Comp. Eng., Modeling, Electronics, and Systems Eng. (DIMES), University of Calabria (UNICAL), Rende, Italy

**Abstract**—Binary Neural Networks (BNNs) are a cost-effective and highly efficient alternative to traditional neural networks. Genome classification is a frequent component of genome analysis pipelines, with a variety of applications spanning pandemic preparedness, AMR resistance control, drinking water and food safety. PatBiNN is a BNN based pathogen genome classifier optimized for edge and field use. It employs a binary multilayer perceptron (MLP) implemented using in-Hamming distance tolerant (similarity search) content addressable memory processing. PatBiNN was designed and manufactured in a commercial 65nm process. It achieves  $F_1$  score of 88%, ROC AUC of 0.986, throughput of 0.8M inferences/s, power consumption of 4.8mW and energy efficiency of 237TOPs/s/W with silicon area of 0.87 mm<sup>2</sup>.

**Index Terms**—Pathogen Classification, Binary Neural Network, Processing-using-Memory, PuM, In-Memory Computing, Content-Addressable Memory

## I. INTRODUCTION

Content-addressable memory (CAM) represents a specialized storage architecture enabling data retrieval based on information content instead of memory addresses. CAM systems perform comprehensive searches across stored data to identify locations where queries achieve matches, whether precise or within specified tolerance levels such as Hamming distance thresholds [1], [2]. Their native parallel search characteristics have found extensive application across diverse domains, particularly in neural computing architectures [3]–[5] and genome classification [6].

A fundamental relationship exists between artificial neural network operations and associative memory mechanisms. In a classification neural network, a neuron exhibiting maximum correlation between input signals (features or activations) and stored weight parameters identifies the appropriate target category, as elevated correlation typically produces maximum output response. Similarly, in associative memories, the storage location containing weight data with optimal (identical or sufficiently similar) correspondence to the input query vector indicates the desired class. This relationship becomes particularly evident in binary neural networks (BNNs), where input activations and weight parameters are constrained to +1 or −1 values, commonly encoded as logical ‘1’ and ‘0’ states. Under these conditions, weight-input multiplication reduces to elementary XNOR operations, which also constitute the fundamental computation in associative memory systems.

BNNs represent efficient network architectures that achieve exceptional energy performance by substituting full-precision

arithmetic with single-bit XNOR operations and replacing accumulation with POPCOUNT functions [7]. Nevertheless, practical implementations of BNN typically necessitate maintaining several network components at full precision to prevent substantial accuracy degradation. In binary convolutional classification network architectures, an input layer frequently utilizes full-precision implementation. Additionally, the last (fully connected) BNN layer must employ high-precision arithmetic to enable reliable class discrimination.

Rapidly evolving pathogens such as SARS-CoV-2, along with environments characterized by high microbial diversity or exposure to novel pathogenic agents, requires computing and storage resources infrequent in genome analysis pipelines available to small clinic labs, point of care or rural hospitals. Nevertheless, analysis time often proves most critical precisely under these demanding conditions. Critical application domains encompass viral pandemics including SARS-CoV-2, dengue fever, and Zika virus, in addition to emerging areas such as drinking water and food safety surveillance, identification of antibiotic-resistant pathogenic strains in water and soil, and point-of-care examination of patient DNA samples potentially affected by unknown or insufficiently researched pathogens.

The primary time-critical bottleneck emerges from the computational analysis of sequenced DNA, such as genome classification. Conventional heuristics based on DNA fragment matching exhibit substantial data-intensive characteristics [6]. When DNA originates from heterogeneous sources such as diverse environments or rapidly mutating pathogens, or when subjected to other confounding factors, genome classification becomes computationally resource-intensive. Neural network based genome classifiers were proposed earlier [8]–[10]. However such solutions typically implement full-precision DNN models which define their size and energy consumption and limit their practicality (for example in resource-constrained field environments).

This research aims to realize a genuine *end-to-end-binary* BNN in-content addressable memory (CAM) implementation of pathogen genome classifier, where the entire network operates in binary mode and executes in-memory, eliminating auxiliary digital processing and software-based full-precision calculations. We introduce PatBiNN, a CAM-based classification BNN accelerator developed and manufactured in silicon using commercial 65 nm technology. Specifically, we implement a binary multilayer perceptron based classifier, whose accuracy,

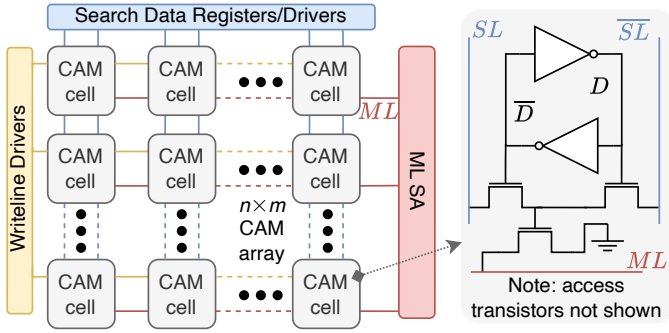


Fig. 1. Top-level view of a content-addressable memory (CAM) array. At the right: a conventional SRAM-based NOR-type CAM cell (access transistors are not shown)

throughput and power consumption were evaluated using a diverse pathogen genome datasets. PatBiNN demonstrates competitive performance metrics (sensitivity, precision,  $F_1$  score and specificity) while delivering exceptional inference speed and energy efficiency. This true end-to-end binary in-CAM implementation is facilitated through a novel Level/Pace/Time-to-digital conversion methodology developed within this work.

This work makes the following contributions:

- To our knowledge, PatBiNN is the first genome classifier by a true end-to-end binary multilayer perceptron implemented in-Content Addressable Memory;
- PatBiNN eliminates the need for full precision operations by implementing a novel a Level/Pace/Time-to-digital conversion scheme;
- PatBiNN was implemented in a 65 nm process and experimentally evaluated by means of silicon measurements.

## II. BACKGROUND

### A. Content-Addressable Memory (CAM)

Fig. 1 shows a  $n$  columns by  $m$  rows NOR-type CAM array. Each row presents a matchline ( $ML$ ) shared across the  $n$  bitcells of a row, and connected to a matchline sense amplifier (MLSA). Searchlines ( $SLs$ ) span all rows of the CAM array. Read and write operations resemble those of conventional six-transistor static random access memory (6T-SRAM). During search operations, the query data pattern, sent from search data registers (SDRs), is driven onto the  $SLs$  and compared against the contents of the bitcells ( $D$  and  $\bar{D}$ ). This comparison is carried out in parallel across the array within a single clock cycle, consisting of two steps: (1) precharging the  $ML$  to  $V_{DD}$  and, (2) driving the query data to the  $SLs$ . The MLSA then detects the  $ML$  state to determine a match or mismatch. Beyond exact-match implementations leveraging CMOS and emerging memory technologies [11]–[13], several CAM architectures support approximate or similarity search functions, including Hamming distance-based approaches [11], [14], [15].

### B. Binary Neural Networks (BNNs)

Fig. 2 illustrates a family of deep learning models in which both weights and neuron outputs are restricted to binary states,

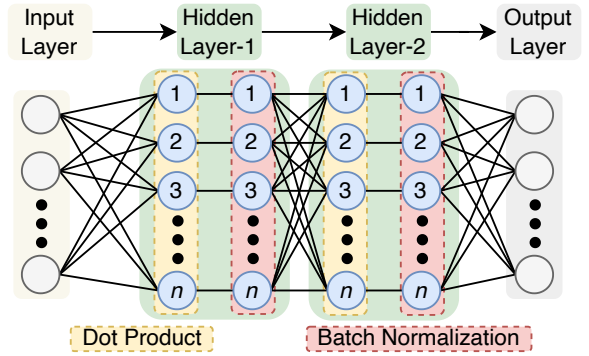


Fig. 2. Multilayer perceptron (MLP) network featuring two hidden layers, each comprising a dot product (XNOR and POPCOUNT) and batch normalization layers.

usually within  $\{-1, +1\}$ . This constraint reduces hardware complexity and memory usage, making BNNs ideal for low-power, compact devices such as edge processors or embedded platforms. By replacing standard floating-point arithmetic with bitwise operations, BNNs deliver substantial speedups while still achieving competitive accuracy on many benchmarks [16].

Within multilayer perceptron (MLP) frameworks, both connection matrices and intermediate activations are binarized. The forward propagation for such layers adopts the following form, using a generic binarization operator  $\mathcal{B}(\cdot)$ :

$$Z_k^{(m+1)} = \psi \left( \mathcal{B} \left( \sum_r V_{kr} Z_r^{(m)} + d^{(m)} \right) \right), \quad (1)$$

where  $Z_r^{(m)}$  represents the output from neuron  $r$  in layer  $m$ ,  $V_{kr}$  is the binary weight linking neuron  $r$  in layer  $m$  to neuron  $k$  in layer  $m+1$ ,  $d^{(m)}$  denotes the bias, and  $\psi(\cdot)$  is a chosen activation function. The binarization process,  $\mathcal{B}(\cdot)$ , typically projects the input onto one of two possible values, effectively inducing a strong nonlinearity.

To ensure stable training and effective use of binary states, batch normalization is introduced after each linear transformation [17]:

$$s = \sum_r (V_{kr} Z_r^{(m)}), \quad (2)$$

$$Z_k^{(m+1)} = \psi \left( \frac{s - \nu}{\sqrt{\tau^2 + \delta}} \cdot \alpha + \beta \right),$$

in which  $\nu$  and  $\tau$  stand for the mean and standard deviation across the batch, while  $\alpha$  and  $\beta$  are learnable scaling and offset parameters. Batch normalization is crucial, as it mitigates representational collapse and encourages activations to populate both  $-1$  and  $+1$  values.

During deployment, batch normalization parameters can be collapsed into a layer-specific constant  $Q_k$  (see Equation (3)) [18]. The practical realization of the binary MLP is thus:

$$Z_k^{(m+1)} = \mathcal{B} \left( \text{POPCOUNT}(\text{XNOR}(V_{kr}, Z_r^{(m)})) + Q_k \right). \quad (3)$$

### C. Related Work: Hardware-accelerated genome classifiers

A DNA classifier aims to identify the organism associated with a given DNA sequence (or individual read), considering that the sequence may be part of a metagenomic sample comprising DNA from diverse species. Hardware-accelerated genome classification uses traditional (for example GPU) as well as processing-in-memory based hardware platforms and implement several classification techniques, including exact and approximate (similarity search-based)  $k$ -mer matching heuristics and classification neural networks (applied to DNA reads or raw nanopore signals in the case of 3rd generation-sequenced DNA).

**Exact  $k$ -mer matching.** Sieve [19] is a DRAM based exact  $k$ -mer-matching accelerator. While its throughput is very high, reliance on exact  $k$ -mer matching may degrade sensitivity in the case of sequencing errors and base ambiguity.

**Approximate search using in-memory processing.** EDAM implements edit-distance-tolerant matching in CMOS CAM arrays, improving robustness but incurring cell/interconnect complexity that limits density and scaling [20]. DIPER explores resistive-memory CAMs for edit-distance-aware matching, trading footprint gains for the well-known write energy/latency/endurance costs of emerging NVMs [21]. Domain-specific SoCs have also emerged: GCOC integrates a similarity (Hamming-tolerant) SAS-CAM with an embedded controller and reports 769.2k reads/s at 1.27 mW in 3.12 mm<sup>2</sup> in 65 nm, targeting portable use. DASH-CAM improves CAM density for viral surveillance and reports large speedups and F1 gains versus software baselines [6], [22]. MajorK [23] implements approximate  $k$ -mer matching based genome classification in commodity off the shelf DRAM.

**Direct raw-signal (squiggle) analysis.** Basecalling which is an integral part of Oxford Nanopore Technologies' (ONT) 3rd generation sequencing pipelines, is the heaviest computing component of such pipelines [24]. SquiggleFilter accelerates virus classification by processing raw ONT's MinION [25] signals directly in Read-Until workflows [24]. RawHash and RawHash2 perform hash-based seeding directly on squiggles for real-time mapping at scale [26], [27]. SquiggleNet classifies nanopore signals with compact CNNs, achieving > 90% host-microbe discrimination using ~1 s of signal [28]. These approaches demonstrate early-stage decisions that cut basecalling/alignment cost on portable ONT platforms).

### III. PATBINN DESIGN

PatBiNN is designed around an approximate search capable (HD tolerant) CAM with processing-in-memory capabilities. Its CAM core was custom-designed using commercial circuit design, evaluation and layout tools. PatBiNN was integrated within a System-on-Chip (SoC) platform, and fabricated in commercial 65nm process. PatBiNN CAM core comprises four 32-kbit banks.

Fig. 3(a) presents the high-level architecture of a single PatBiNN bank, highlighting its peripherals. Fig. 3(b) shows the physical layout of a single PatBiNN bank, with an estimated footprint of 0.21 mm<sup>2</sup>. Fig. 3(c) shows the PatBiNN

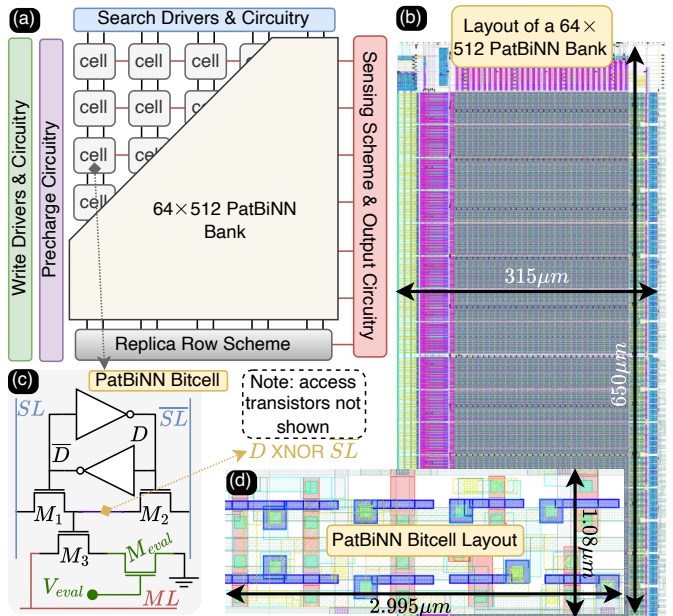


Fig. 3. (a) Top-level view of a PatBiNN memory bank. (b) Layout of a 64×512 PatBiNN bank. (c) PatBiNN bitcell and (d) layout.

bitcell, made up of a conventional 9T NOR CAM cell (access transistors are not shown) [1] with an extra nMOS transistor (refer to  $M_{eval}$ ). Fig. 3(d) shows the PatBiNN bitcell layout featuring an area footprint of about 3.24  $\mu\text{m}^2$ .

To accommodate a broad spectrum of Hamming distance (HD) tolerances in PatBiNN search, three distinct, user-adjustable voltage controls are engaged (depicted in yellow within Fig. 3):

- **MLSA reference voltage ( $V_{ref}$ ):** Decreasing  $V_{ref}$  elevates the permissible HD threshold in PatBiNN, whereas increasing it reduces HD tolerance.
- **$V_{eval}$ :** Modulates the discharge rate of the PatBiNN match line ( $ML$ ) through the conductance of the  $M_{eval}$  transistor. Lowering  $V_{eval}$  results in slower  $ML$  discharge, thereby raising the HD acceptance range; conversely, increasing  $V_{eval}$  quickens discharge and restricts HD tolerance.
- **$V_{st}$ :** Alters the moment at which the MLSA output is sampled. Advancing the sampling with  $V_{st}$  amplifies the HD threshold, while delaying it diminishes the tolerance achievable by PatBiNN.

The use of all three user-tunable voltages is required to enable large HD tolerance and support a wide range of POPCOUNT values, as further explained in Section IV. Table I details some ( $V_{ref}$ ,  $V_{eval}$ ,  $V_{st}$ ) combinations and the HD threshold levels enabled by them.

### IV. PRINCIPLE OF OPERATION

While SOTA solutions use ADC or TDC in the output (fully connected) layer to convert analog POPCOUNT to digital representation pointing to a class, PatBiNN applies a different strategy. We utilize the fact that ( $V_{ref}$ ,  $V_{eval}$ ,  $V_{st}$ ) voltages are

TABLE I  
 $V_{REF}$ ,  $V_{EVAL}$ ,  $V_{ST}$  COMBINATIONS AND THE HD TOLERANCE THRESHOLD LEVELS ENABLED BY THEM.

| $V_{ref}$ (mV) | $V_{eval}$ (mV) | $V_{st}$ (mV) | HD Tolerance |
|----------------|-----------------|---------------|--------------|
| 1200           | 1200            | 1200          | 0            |
| 750            | 950             | 1200          | 4            |
| 775            | 600             | 1200          | 8            |
| 1175           | 350             | 1150          | 12           |
| 950            | 525             | 1100          | 16           |
| 1025           | 475             | 1000          | 20           |
| 950            | 500             | 1025          | 24           |
| 775            | 600             | 1100          | 28           |
| 1175           | 400             | 1150          | 32           |
| 1000           | 475             | 725           | 36           |

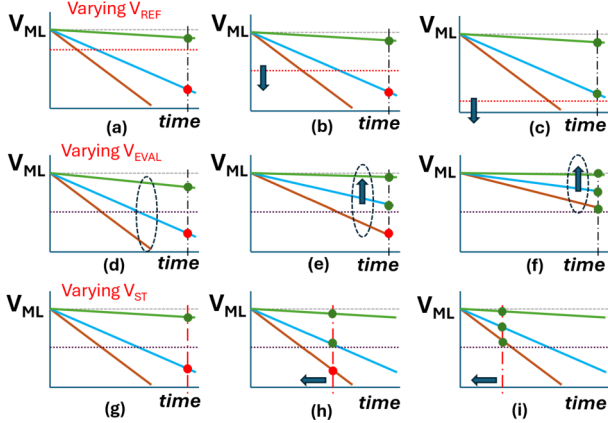


Fig. 4. Popcount Level/Pace/Time-to-digital conversion by configuration voltage sweep: (a)-(c)  $V_{ref}$ , (d)-(f)  $V_{eval}$ , (g)-(i)  $V_{st}$

user-configurable and can be tuned by software from outside the SoC. We refer to this approach as Level/Pace/Time-to-digital conversion (Fig. 4).

We are able to sample the ML at multiple voltage levels (by sweeping  $V_{ref}$ , Fig. 4(a)-(c)), with different pace of ML discharge (by sweeping  $V_{eval}$ , Fig. 4(d)-(f)), and at different time points (by sweeping  $V_{st}$ , Fig. 4(g)-(i)).

Specifically, we sweep the levels ( $V_{ref}$ ,  $V_{eval}$ ,  $V_{st}$ ) of Table I, bottom to top, which effectively applies different HD tolerance threshold levels, ranging from a certain maximum level  $HD_L$  to zero.

The higher the analog POPCOUNT, the lower the HD tolerance threshold at which MLSA outputs ‘1’. Zero HD tolerance threshold (exact match) indicates the highest POPCOUNT value.  $HD_L$  is associated with a relatively low POPCOUNT level. The former only occurs (or rather is supposed to occur) in the correct class, while the latter may accidentally register in unrelated (wrong) class(es).

Logically, each PatBiNN row (Fig. 5) performs the following operation, whose components are detailed below:

$$\mathbf{X}_j^{l+1} = \text{MAJ} \left( \sum_i \text{XNOR}(W_{ji}, \mathbf{X}_i^l) + \sum_i c_{ji} \right), \quad (4)$$

where MAJ is a logic majority operation, and  $c_{ji}$  are the bits of a fused batch normalization constant ( $C_j = \sum_i c_{ji}$ ).

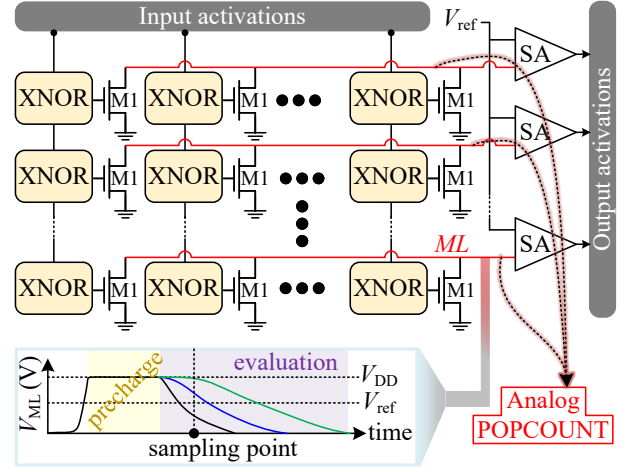


Fig. 5. PatBiNN concept

The following details the components of (4) and their implementation:

- XNOR is implemented by a per-bit matching in a PatBiNN cell that stores a binary weight  $W_{ji}$ . The binary input activation  $X_i$  bit is asserted on  $SL$  ( $\overline{SL}$ ). If  $X_i \neq W_{ji}$  (representing  $-1$ ), the  $ML$  discharge path through the cell. Otherwise, if  $X_i = W_{ji}$  (representing  $+1$ ), the  $ML$  does not discharge.
- Batch normalization parameters are either trainable or calculated during inference, and thereby are known in advance. Therefore, it is possible to replace batch normalization by adding to (fusing into) the output of a network layer (a dot product) a batch normalization constant  $C_j$ . In PatBiNN we implement batch normalization by offsetting  $V_{ref}$ ,  $V_{eval}$  and  $V_{st}$ , which when carefully executed creates an effect identical to adding ( $C_j = \sum_i c_{ji}$ ) term to the POPCOUNT before MAJ.
- Majority is implemented as follows:  $ML$  is precharged (refer to  $ML$  inset in Fig. 5) and the input activations  $X^l$  are asserted. Suppose the MLSA reference voltage  $V_{ref}$  is calibrated such that the  $ML$  voltage ( $V_{ML}$ ) crosses it at the sampling time when the number of matching and mismatching bitcells in a row is the same. This way, if the number of matches exceeds the number of mismatches (refer to green line at the sampling point in Fig. 5), the  $ML$  discharges slowly and so  $V_{ML}$  crosses  $V_{ref}$  after the MLSA sampling, generating  $X_i^{l+1} = 1$  (representing  $+1$ ). Otherwise, if the number of matching cells is below the number of mismatching ones (refer to black line at the sampling point in Fig. 5), the  $ML$  discharges faster, and  $V_{ML}$  crosses the  $V_{ref}$  before the MLSA sampling time, signaling  $X_i^{l+1} = 0$  (representing  $-1$ ).

## V. EVALUATION

### A. Evaluation Setup and Figures of Merit

We evaluate the accuracy of the binary MLP implemented in PatBiNN using a set of viral genomes from NCBI site [29],

as follows: SARS-CoV-2, H1N1, Hep. B, HSV, Lassa, Variola, Measles, Norovirus, Rabies and Zika.

PatBiNN was trained using Illumina-style DNA reads generated by ART simulator [30] from fully assembled NCBI genomes. The model was trained on 100,000 reads for each virus. This viral dataset was additionally evaluated using Kraken-2 [31] to make sure all DNA reads tested by PatBiNN are tagged correctly.

PatBiNN implements a genome classifier that classifies viral pathogens into 10 classes, where the model topology is as follows:  $1024 \rightarrow 128 \rightarrow 10$ . Input features are generated by extracting the 1024 most frequent  $k$ -mers ( $k = 5$ ) from the sequences, resulting in a  $1024 \times 1$  binary vector for each 150bp read. During inference, the preprocessing of the data was performed using exact search mode, which generated the activation vector fed into the first layer of the MLP.

The figures of merits in PatBiNN classifier performance evaluation are:

- Sensitivity:  $\frac{TP}{TP+FN}$
- Precision:  $\frac{TP}{TP+FP}$
- $F_1$  score:  $\frac{2 \cdot \text{Precision} \cdot \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}}$
- Specificity:  $\frac{TN}{TN+FP}$

We also present the ROC diagram, plotting true positive rate vs. false positive rate, and the area under the curve (AUC) figure.

## B. Experiments and interpretation of results

We conduct two experiments, as follows:

- 1) The input MLP layer is implemented completely binary, with the output activations generated as majority of XNOR results and fused batch normalization. The output layer which is expected to output a target class, implements the level/pace/time-to-digital conversion by varying the HD tolerance threshold by tuning  $V_{ref}$ ,  $V_{val}$ ,  $V_{st}$ . In this experiment, we terminate the procedure once a single PatBiNN output becomes '1'. We then interpret the result as follows:

- If a DNA read hits in the right class, its TP counter is incremented. TN counters of the rest of (correctly unclassified) classes are incremented as well.
- If such DNA read hits in a wrong class, the target (correct) class's FN counter is incremented. The accidentally hit class's FP is also incremented. TN counters of the rest of (correctly missed) classes are incremented as well.

- 2) The second experiment is an extension of the first one. We continue increasing the HD tolerance threshold until it reaches the maximum level  $HD_L$ . During this process, more than a single PatBiNN output may become one. We then interpret this experiment's result as follows:

- If a target class hits, its TP is incremented.
- If a target class misses, its FN is incremented.
- TP counters of all accidentally hit classes are incremented.

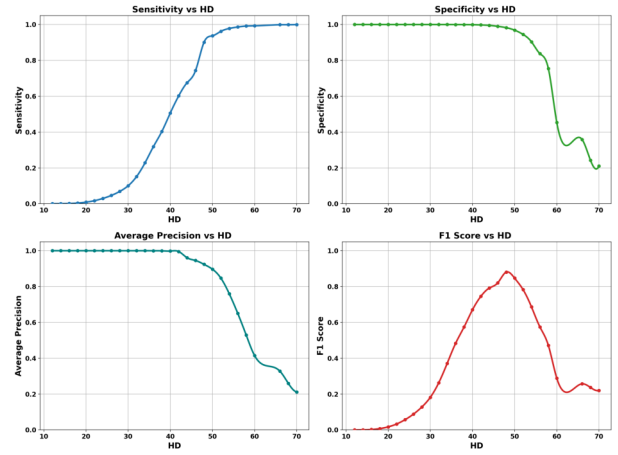


Fig. 6. Average Sensitivity, Precision,  $F_1$  Score and Specificity as a function of HD tolerance threshold

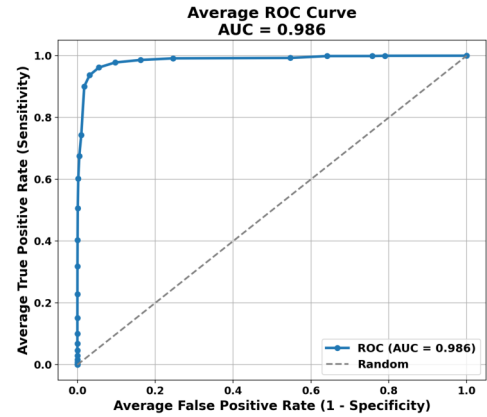


Fig. 7. ROC diagram (HD tolerance threshold serves as a tuning knob)

- TN counters of the classes that correctly miss are incremented.

## C. Evaluation Results

Sensitivity, precision,  $F_1$  score and specificity averaged across all evaluated genomes as functions of HD tolerance threshold are presented in Fig. 6. All diagrams begin around HD tolerance threshold of 12 because below this threshold, no '1' popcount results are obtained (i.e. the tolerance threshold is too low for a DNA read to be classified using BNN classifier). With growing HD tolerance threshold, the sensitivity improves due to reduction in false negative rate (a growing number of DNA reads yielding lower popcount value in the output layer are classified correctly). At the same time, the precision falls, due to increase in the false positive rate (for the same reason - an elevated HD threshold level lets more classes to falsely output '1'). The average  $F_1$  score maxes out at 88% at HD tolerance threshold of 48.

High specificity is maintained almost till the HD tolerance threshold of 50 (where it drops from 100% to 97%). Above that HD tolerance, specificity falls rather quickly, due to high

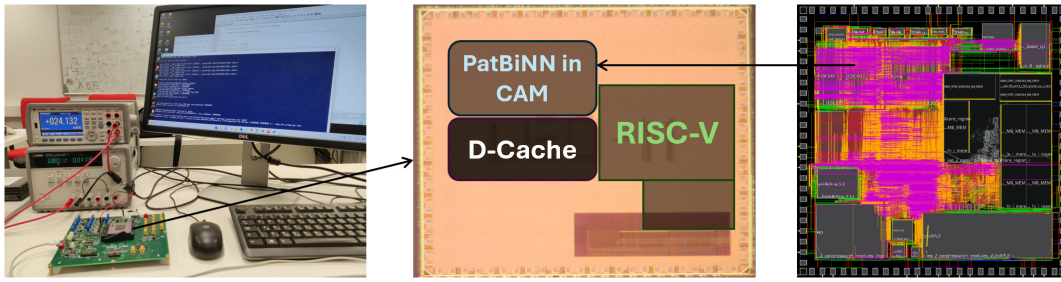


Fig. 8. PatBiNN evaluation setup, research system-on-chip (SoC) micrograph with PatBiNN marked, SoC layout.

TABLE II  
SUMMARY OF PATBINN HARDWARE PARAMETERS.

|                                  |                      |
|----------------------------------|----------------------|
| <b>Technology</b>                | 65 nm CMOS           |
| <b>Supply Voltage</b>            | 1.2 V                |
| <b>SoC Area</b>                  | 2.38 mm <sup>2</sup> |
| <b>PatBiNN capacity</b>          | 128 kbit             |
| <b>PatBiNN area</b>              | 0.87 mm <sup>2</sup> |
| <b>PatBiNN power consumption</b> | 4.8 mW               |
| <b>PatBiNN energy efficiency</b> | 237TOPs/s/W          |
| <b>Operating Frequency</b>       | 150 MHz              |

false positive rate caused by multiple classes yielding popcount values within such threshold.

The ROC diagram, plotting true positive rate vs. false positive rate, averaged across all evaluated genomes, is shown in Fig. 7. The average AUC is 0.986.

#### D. Performance, power consumption, and silicon area

Fig. 8 (left) depicts the PatBiNN evaluation setup with its test (evaluation) board. Fig. 8 (middle, right) presents the SoC micrograph with PatBiNN marked and SoC layout, respectively. Silicon-based experimental measurements were utilized to characterize the performance, silicon footprint, and power consumption of PatBiNN. Evaluations were carried out at 150 MHz. Given that adjusting the voltage regulators is not instantaneous, a batching strategy is employed to amortize the voltage tuning overhead across the processing of multiple DNA sequences. In particular, each chosen set of  $V_{ref}$ ,  $V_{eval}$ ,  $V_{st}$  is maintained for the fully connected output layer while several DNA reads are processed consecutively, prior to performing any additional voltage adjustment. PatBiNN provides a measured throughput of 0.8M inferences (i.e. DNA reads) per second for a  $1024 \rightarrow 128 \rightarrow 10$  binary MLP classifier configuration. At a supply voltage of 25 °C, the power consumption is 4.8 mW, corresponding to 237TOPs (Binary operations)/s/W of effective energy efficiency, with all BNN computation performed natively and without delegation of different network components to external digital processing or software implementation. Combined, PatBiNN and the controlling RISC-V processor dissipate around 0.3 mW in total. The PatBiNN alone occupies 0.87 mm<sup>2</sup> of silicon area, while the total SoC footprint (including the RISC-V) is 2.38 mm<sup>2</sup>. A consolidated summary of the system’s area, power, and performance attributes is available in Table II.

## VI. CONCLUSION

In this work, we presented PatBiNN, an in-CAM binary neural network based pathogen genome classifier. PatBiNN implements a true end-to-end genome classification binary neural network, replacing full precision operations, typically outsourced to host CPU or auxiliary digital units in state-of-the-art BNN solutions, through application of a novel level/pace/time-to-digital conversion technique. The latter is implemented by applying user-configurable control voltages to adjust the Hamming distance tolerance threshold of approximate matching in PatBiNN CAM arrays. PatBiNN is designed and fabricated in a commercial 65 nm process, and evaluated (accuracy, inference throughput, power consumption and silicon area) on a variety of viral pathogen classes using silicon measurements.

## ACKNOWLEDGEMENTS

This work was supported by the Sweden-Israel Lise Meitner research collaboration under grant number 1001569396, by the Israeli Ministry of Science, Innovation and Technology, under grants number 1001818838 and 1001702600, by the Pazy Foundation under grant number 5100089682, and by the Italian Ministry of University and Research (MUR) under grant number SOE\_20240000022.

## REFERENCES

- [1] K. Pagiamtzis and A. Sheikholeslami, “Content-addressable memory (CAM) circuits and architectures: a tutorial and survey,” *IEEE J. of Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, 2006.
- [2] E. Garzón, E. Rechef, R. Golman, O. Harel, Y. Harary, P. Snapir, M. Lanuzza, A. Teman, and L. Yavits, “A 128-kbit Approximate Search-Capable Content-Addressable Memory (CAM) With Tunable Hamming Distance,” *IEEE Journal of Solid-State Circuits*, vol. 60, no. 8, pp. 3009–3019, 2025.
- [3] A. F. Laguna, M. M. Sharifi, A. Kazemi, X. Yin, M. Niemier, and X. S. Hu, “Hardware-software co-design of an in-memory transformer network accelerator,” *Frontiers in Electronics*, vol. 3, p. 847069, 2022.
- [4] Y. Halawani, B. Mohammad, M. A. Lebdeh, M. Al-Qutayri, and S. F. Al-Sarawi, “ReRAM-based in-memory computing for search engine and neural network applications,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 388–397, 2019.
- [5] W. Choi, K. Jeong, K. Choi, K. Lee, and J. Park, “Content addressable memory based binarized neural network accelerator using time-domain signal processing,” in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.
- [6] Y. Harary, P. Snapir, S. Siman-Tov, C. Kruphman, E. Rechef, Z. Jahshan, E. Garzón, and L. Yavits, “GCOC: A Genome Classifier-On-Chip based on Similarity Search Content Addressable Memory,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 19, no. 3, pp. 484–495, 2025.

- [7] M. Zahedi, T. Shahroodi, C. Escuin, G. Gaydadjiev, S. Wong, and S. Hamdioui, "BCIM: Efficient implementation of binary neural network based on computation in memory," *IEEE Transactions on Emerging Topics in Computing*, 2024.
- [8] C. M. Dasari *et al.*, "Explainable deep neural networks for novel viral genome classification," *Applied Intelligence*, vol. 52, pp. 9785–9800, 2022. [Online]. Available: <https://dl.acm.org/doi/abs/10.1007/s10489-021-02572-3>
- [9] F. Mock *et al.*, "Taxonomic classification of dna sequences beyond sequence similarity using deep neural networks," *PNAS*, vol. 119, no. 35, p. e212636119, 2022. [Online]. Available: <https://www.pnas.org/doi/10.1073/pnas.2122636119>
- [10] F. Mock, F. Kretschmer, A. Kriese, S. Böcker, and M. Marz, "Bertax: taxonomic classification of dna sequences with deep neural networks," *BioRxiv*, pp. 2021–07, 2021.
- [11] A. Shaban, T.-H. Hou, and M. Suri, "SOT-MRAM-based approximate content addressable memory for DNA classification," *IEEE Transactions on Electron Devices*, 2024.
- [12] Y. Pan, A. Wheeldon, M. Mughal, S. Agwa, T. Prodromakis, and A. Serb, "An energy-efficient capacitive-rram content addressable memory," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 11, pp. 5285–5295, 2024.
- [13] W. Xu, J. Luo, Z. Chen, B. Fu, Z. Fu, K. Wang, Q. Huang, and R. Huang, "A Novel Ferroelectric FET based Universal Content Addressable Memory with Reconfigurability for Area-and Energy-Efficient In-Memory-Searching System," *IEEE Electron Device Letters*, 2024.
- [14] M. M. Taha and C. Teuscher, "Approximate memristive in-memory Hamming distance circuit," *ACM J. on Emerging Technologies in Computing Systems (JETC)*, vol. 16, no. 2, pp. 1–14, 2020.
- [15] C. Ni, S. Chen, C.-K. Liu, L. Liu, M. Imani, T. Kämpfe, K. Ni, M. Niemier, X. S. Hu, C. Zhuo *et al.*, "TAP-CAM: A Tunable Approximate Matching Engine based on Ferroelectric Content Addressable Memory," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, 2024, pp. 1–9.
- [16] C. Yuan and S. S. Agaian, "A comprehensive review of binary neural network," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 12949–13013, 2023.
- [17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456.
- [18] T. Hirtzlin, M. Bocquet, B. Penkovsky, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz, "Digital biologically plausible implementation of binarized neural networks with differential hafnium oxide resistive memory arrays," *Frontiers in neuroscience*, vol. 13, p. 1383, 2020.
- [19] L. Wu, R. Sharifi, M. Lenjani, K. Skadron, and A. Venkat, "Sieve: Scalable in-situ dram-based accelerator designs for massively parallel k-mer matching," in *2021 ACM/IEEE 48th annual international symposium on computer architecture (ISCA)*. IEEE, 2021, pp. 251–264.
- [20] R. Hanhan, E. Garzón, Z. Jahshan, A. Teman, M. Lanuzza, and L. Yavits, "EDAM: edit distance tolerant approximate matching content addressable memory," in *Proc. of the 49th Annual Int. Symp. on Comp. Arch. (ISCA)*, 2022, pp. 495–507.
- [21] I. Merlin, E. Garzón, A. Fish, and L. Yavits, "DIPER: Detection and Identification of Pathogens using Edit distance-tolerant Resistive CAM," *IEEE Transactions on Computers*, vol. 73, no. 10, pp. 2463–2473, 2024.
- [22] Z. Jahshan, I. Merlin, E. Garzón, and L. Yavits, "DASH-CAM: Dynamic Approximate Search Content Addressable Memory for genome classification," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23, New York, NY, USA, 2023, p. 1453–1465.
- [23] Z. Jahshan and L. Yavits, "Majork: Majority based kmer matching in commodity dram," *IEEE Computer Architecture Letters*, vol. 23, no. 1, pp. 83–86, 2024.
- [24] T. Dunn, H. Sadasivan, J. Wadden, K. Goliya, K.-Y. Chen, R. Das, D. Blaauw, and S. Narayanasamy, "Squiggfilter: An accelerator for portable virus detection," in *arXiv:2108.06610*, 2021. [Online]. Available: <https://arxiv.org/abs/2108.06610>
- [25] Oxford Nanopore Technologies, "Minion portable nanopore sequencing device," 2025. [Online]. Available: <https://nanoporetech.com/products/sequence/minion>
- [26] C. Firtina, M. Alser, J. Park, J. Lindegger, and O. Mutlu, "Rawhash: enabling fast and accurate real-time analysis of raw nanopore signals using hash-based seeding," *Bioinformatics*, vol. 39, no. Supp\_1, pp. i297–i306, 2023.
- [27] C. Firtina, M. Soysal, J. Lindegger, and O. Mutlu, "Rawhash2: mapping raw nanopore signals using hash-based seeding and adaptive quantization," *Bioinformatics*, vol. 40, no. 8, p. btae478, 2024.
- [28] Y. Bao, J. Wadden, J. R. Erb-Downward, P. Ranjan, W. Zhou, T. L. McDonald, R. E. Mills, A. P. Boyle, R. P. Dickson, D. Blaauw, and J. D. Welch, "Squigglenet: real-time, direct classification of nanopore signals," *Genome Biology*, vol. 22, no. 1, p. 298, 2021.
- [29] NCBI, "Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information," 2021. [Online]. Available: <https://www.ncbi.nlm.nih.gov/>
- [30] W. Huang, L. Li, J. R. Myers, and G. T. Marth, "ART: a next-generation sequencing read simulator," *Bioinformatics*, vol. 28, no. 4, pp. 593–594, 2012.
- [31] D. E. Wood, J. Lu, and B. Langmead, "Improved metagenomic analysis with Kraken 2," *Genome Biology*, vol. 20, no. 1, p. 257, 2019.