DNA Sequencing using Brain-inspired Hyperdimensional Computing

Mohsen Imani, Tarek Nassar, Justin Morris, and Tajana Rosing
Computer Science and Engineering, UC San Diego, La Jolla, CA 92093, USA

***Abstract*—DNA sequencing has a vast number of applications in a multitude of applied fields including, but not limited to, medical diagnosis and biotechnology. In this paper, we propose HDNA to apply the concepts of hyperdimensional (HD) computing (computing with hypervectors) to DNA sequencing. HDNA first assigns holographic and (pseudo)random hypervectors to DNA bases. Using an encoder, it then exploits the orthogonality of these hypervectors to represent a DNA sequence by generating a *class* hypervector. The class hypervector keeps the information of combined individual hypervectors (i.e., the DNA bases) with high probability. HDNA uses the same encoding to map a DNA sequence with unknown labels to a *query hypervectors* and performs the classification task by checking the similarity of the query hypervector against all class hypervectors. Our experimental evaluation shows that HDNA can achieve 99.7% classification accuracy for Empirical dataset which is 5.2% higher than state-of-the-art techniques for the same dataset. Moreover, our HDNA can improve the execution time and energy consumption of classification by 4.32× and 2.05× respectively, when compared against prior techniques.**

I. Introduction

The process of determining the order of nucleotides present in a DNA molecule is called DNA sequencing; there are four bases in strand DNA: adenine (A), guanine (G), cytosine (C), and thymine (T). The goal of DNA sequencing is to determine the physical order of these bases in a molecule of DNA. On the application level, DNA sequencing can be used to determine the sequences of individual genes, clusters of genes, and entire genomes of any organism [1]. In molecular biology sequencing allows researcher to study genomes and proteins and use this information to detect and identify any possible changes within genes [2]. In medicine, sequencing can help extract and identify the sequence of genes from patients to determine if there may be a risk of any number of genetic diseases [3].

In this paper, we propose the idea of Hyperdimensional (HD) DNA sequencing, called HDNA, which significantly improves the accuracy and efficiency of DNA classification [4]. Brain-inspired HDNA algorithm emulates cognitive tasks by computing with hypervectors as opposed to computing with numbers [5], [6], [7], [8], [9]. Instead of the traditional use of numerical representations, HD computations are defined by patterns that mimick the activity of neurons. HDNA assigns holographic and (pseudo)random hypervectors with i.i.d. components to DNA bases, then exploits the orthogonality of these hypervectors in order to generate hypervectors corresponding to DNA sequences, while keeping the information of the combined individual vectors with high probability. After training class hypervectors, our design uses the same encoding to map an unknown DNA sequence to a new hypervector, called a *query hypervector*. The inference is then made by checking for the similarity of these *query hypervectors* against all available class hypervectors, and returning the class with the highest Hamming distance similarity. Our experimental evaluations over well-known datasets show that HDNA can achieve 99.7% accuracy classifying Empirical dataset which is 5.2% higher than state-of-the-art techniques classifying the same task. Moreover, our HDNA can improve the execution time and energy consumption of classification by 4.32× and 2.05× respectively, when compared against prior techniques.

A. Hyperdimensional Computing

Hyperdimensional (HD) computing captures and imitates the idea of pattern recognition implemented with massive circuits in the form of hypervectors, which are vectors with dimensionality in the thousands. HD computing is built on a well-defined set of operations and offers a complete computational paradigm that can be applied to a vast number of learning problems. Examples include analogy-based reasoning, sequence memory, language recognition, biosignal processing, and predictions from multimodal sensor fusion [10], [7], [9]. These applications use HD computing to encode temporal analog signals. In contrast, in this paper we focus on mapping DNA sequences into HD space for classification/recognition task.

II. Hyperdimensional DNA Sequencing

In this paper we propose a hyperdimensional DNA classifier, which encodes DNA sequences to hypervectors, and applies the inference task over incoming query hypervectors. On the higher level, HDNA consists of two main blocks: encoder and associative memory. The encoder maps DNA sequences to hypervectors and combines them together in order to generate a single model representing each output class. These class models are then stored in the associative memory. In test mode, unknown input data is mapped onto high-dimensional space using the same encoding, and associative memory performs the classification task by searching for a class model which has the largest similarity to the input hypervector.

For simplicity, we will explain the functionality of the proposed design using an implementation of classification over an Empirical dataset [11]. This dataset consists of eight classes of species within the animal, fungi and plant kingdoms, each contains several DNA sequences corresponding to their respective class. The goal of DNA sequencing is to learn the patterns of the DNAs in each class, such that if a new DNA sequence was introduced, our design can recognize the class which it belongs to. Traditionally, researchers use supervised machine learning algorithms for classification tasks such as *K*-NN and SVM, however, these algorithms do not provide good enough accuracy for classifying longer sequences of DNA.

A. DNA in High-Dimensional Space

In this work, we propose a novel hyperdimensional DNA sequencing technique, called HDNA, consisting of encoder and associative memory. The encoder module learns the patterns of all DNA sequences that exist within a class and encodes them into a single hypervector with D dimensions. Each class is then associated with a hypevector which is encoded using all the information from that class. When considering a single sequence of DNA with length m, our goal is to map this sequence to a hypervector which not only allows us to save the bases stored on the sequence, but also allows us to store some information about the position of each base in the sequence. To this end, HDNA assigns holographic and (pseudo)random hypervector with i.i.d. components and D dimensions to DNA bases ($L_A,\ L_C,\ L_G,\ L_T$). Each element within a hypervector is assigned a 0 or 1 value randomly. This along with long dimensionality makes these hypervectors semi-orthogonal such that:

$$\delta(L_i,\ L_j) < D/2, \qquad i,j \in \{A,\ C,\ G,\ T\} \ \ \& \ \ i \neq j$$

where δ measures the similarity between the hypervectors.

We will propose two encoding schemes for HDNA to map and classify data to high-dimensional spaces: (i) Encoder I, a Ngram-based encoding which uses permutation and addition to encode the DNA sequences to hypervectors and (ii) Encoder II, a record-based encoding which maps DNA sequences to high-dimensional space using multiplication and addition. Through this section, we will first explain the functionality of these two encoding schemes. In section IV, we explore the accuracy, efficiency and robustness which these two encoding schemes can provide.

B. Encoder I: Ngram-based encoding

Encoding Module: HDNA combines base hypervectors in order to generate a hypervector representing a DNA sequence. The goal of DNA sequencing is to find the sequence patterns by determining the exact position of bases in a sequence. HDNA considers the impact of positions in generating the sequence hypervector by applying a unique number of permutations for bases in each position. Each permutation generates a hypervector which is unrelated to the given hypervector $\delta(\rho(L_A),\ L_A) \approx D/2$. This operation is commonly used for storing a sequence of tokens in a single hypervector. In the geometrical sense, the permutation rotates the hypervector in the space. To encode DNA sequences of length m, HDNA looks at the the sequence in an n-gram windows ($n = 2, 3, \ldots$). The hypervectors in an n-gram is combined as follows:

$$S_1 = [L_1 + \rho(L_2) + \rho\rho(L_3) + \cdots + \rho ... \rho(L_N)]$$

$$\{L_1,\ L_2, \ldots, L_N\} \in \{L_A,\ L_C,\ L_G,\ L_T\}$$

Using this encoding, the first element in n-gram takes no permutation. The second element gets a single permutation and in general i^{th} position in n-gram is permuted by $n-1$ position. This technique differentiates the impact of bits, as well as their physical position on the final sequence hypervector. Next, an n-gram window shifts by a single position over DNA sequence and encodes the new sequence in n-gram windows to a binarized hypervector(S_2). This process continues until n-gram windows cover all elements in DNA sequence and generate the last n-gram hypervector (S_{m-n+1}).

All generated n-gram hypervectors are added together (element-wise) in order to generate a new hypervector representing the DNA sequence. The generated sequence hypervector can have integer elements. Hypervectors with integer elements increase the cost of HDNA computation. Hence, HDNA binarizes such hypervector by applying *majority* function over each dimension of S_1.

$$S = [S1 + S2 + \cdots + S_{m-n+1}]$$

In this equation, *Majority* is denoted as $[+]$ and it checks each dimension of all hypervectors combined together. If there exists more 1s than 0s on that dimension, the binarized hypervector sets to 1 on that dimension, otherwise it assigns to 0. The result of the *majority* function preserves similarity to its component hypervectors i.e., $\delta([L_A + L_C + L_T],\ L_A) < D/2$. Hence, the majority function is well suited for representing sets. Since each class can have multiple DNA sequence within it, our design generates a DNA hypervector using the same encoding and then adds these hypervectors to generate a unique hypervector representing each class. HDNA generate all class hypervectors in the same way.

In training, HDNA generates the class hypervectors and then stores them in an associative memory module. During test/inference, HDNA uses the same encoding scheme to encode an unknown DNA sequence to a *query hypervector*. To perform classification task, associative memory measures the similarity of query hypervector to all class hypervectors and selects a class with the maximum similarity. This similarity is defined as Hamming distance between the query and class hypervectors.

C. Encoder II:Record-based encoding

Encoding Module: Although HDNA using Encoder I achieves classification accuracy of 96%, this accuracy can be further improved using a unique signature for each base that exists within the DNA sequence. The Encoder I saves the sequence of the bases within each n-gram using permutation, however, it cannot store the order of the n-grams in the final sequence hypervector. This is important in DNA sequencing as DNAs can often span over long lengths. In order to consider the order of n-grams in the encoded DNA hypervector, we propose another encoding scheme which considers a unique identifier for each DNA position within the sequence. This encoding assigns a unique identification (*ID*) hypervector

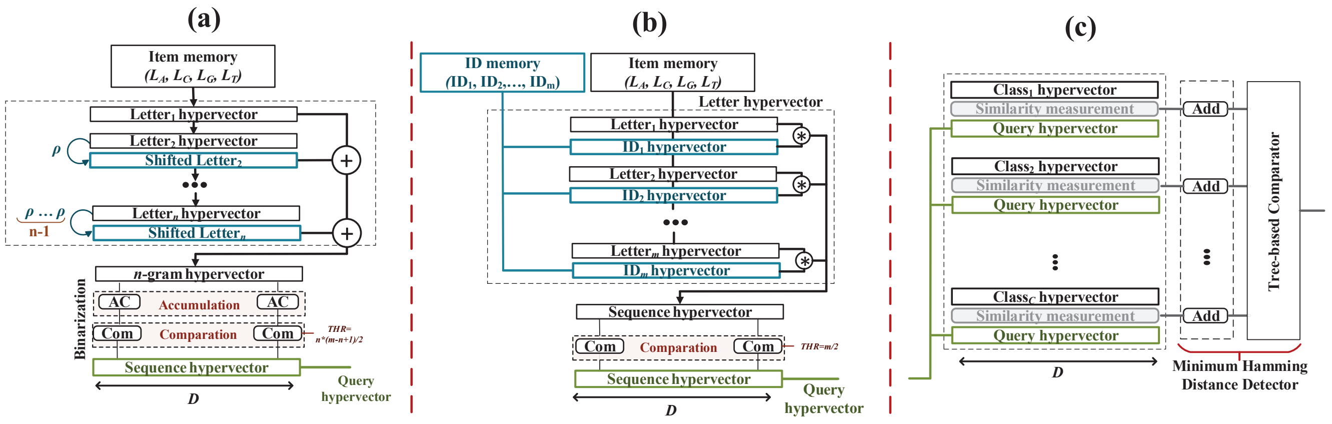

Fig. 1. Overview of HDNA architecture consisting of: (a) Encoder I architecture, (b) Encoder II architecture and (c) associative memory.

to each base position. These *ID* hypervectors are generated randomly such that each base position in sequence will get a unique hypervector $\{ID_1,\ ID_2,\ldots,ID_m\}$. These hypervectors are semi-orthogonal as they are generated in fully random manner.

$$\delta(ID_i,\ ID_j) < D/2, \qquad 1 \leqslant i,j \quad \& \quad i \neq j$$

The m is defined by the length of the longest DNA sequence in the training dataset. Using these positional hypervectors, the DNA sequence can be generated in a single step using the following equations:

$$S = [ID_1 * L_1 + ID_2 * L_2 + ID_3 * L_3 + \cdots + ID_m * (L_m)]$$
$$\{L_1,\ L_2,\ldots,L_m\} \in \{L_A,\ L_C,\ L_G,\ L_T\}$$

Encoder II requires element-wise multiplication of the position hypervectors with the base associated hypervectors. This technique differentiates the impact of each base on the final sequence hypervector depending on the position of such base in the sequence. Similar to Encoder I, the sequence hypervector is binarized using majority function over each dimension. This encoder uses the same associative memory explained above. Our evaluation shows that using this encoding improves the classification accuracy of HDNA to 99%. In terms of hardware efficiency, this encoding would have higher memory requirement and energy cost compare to scheme.

III. Hardware Implementation

In this section, we describe the digital hardware implementation of the HDNA accelerator and the trade-off of HDNA using Encoder I and Encoder II. Figure 1 shows the overview architecture of proposed HDNA consisting of encoder (Encoder I or Encoder II) and associative memory.

1) Encoder I: Figure 1a shows the structure of the Encoder I. Encoder I works based on permutation and addition. Encoders use an item memory block to store four pre-generated base hypervectors ($\{L_A,\ L_C,\ L_G,\ L_T\}$). During the test/inference, Encoder I reads the DNA sequences and accordingly fetches a base hypervector from the item memory. The encoder applies a permutation to each vector in n-gram depending on their physical positions. Next, all permuted hypervectors within the n-gram add together element-wise in order to generate a unique sequence hypervector. Finally, a DNA hypervectors are binarized using comaprator block, which compares each hypervector element with half of the maximum possible value that elements can get ($THR = \ n*(m-n+1)/2$). In each dimension, if the sequence value is less than THR, the value in that dimension will go to 0, otherwise it will be assigned to 1 bit.

2) Encoder II: Figure 1b shows the overview of Encoder II architecture. This encoder has two memory blocks: item memory and position memory. Similar to Encoder I, item memory stores the base hypervectors while position memory stores a unique hypervector corresponding to each position in a sequence. In comparison to item memory, the size of required position memory is very large and is determined by the maximum length of DNA sequence in the dataset. This memory increases the cost of Encoder II. In Encoder II, the encoding happens by multiplying the position and base hypervectors over the whole DNA sequence. This multiplication in hardware is implemented using an XOR array. Then, the m generated hypervectors are accumulated element-wise using a counter block. Finally, comparator blocks binarizes the vector by comparing each element with half of a maximum value each element can get ($THR = \ m/2$). In any dimension, if the value is larger than $m/2$, it will be assigned to 1, otherwise it will be set to 0.

3) Associative Memory: As Figure 1c shows, both proposed encoding schemes use the same associative memory architecture for classification. In hardware, Hamming distance similarity implements using an XOR array. XOR gates compare bit similarity of the query and class hypervectors. An adder block counts the number of 1s at the output of XORs comparing two vectors. Finally, a comparator block in tree structure compares the Hamming distance similarities and selects a class which has the minimum distance with a query hypervector.

IV. Experimental Results

A. Experimental Setup

We describe the functionality of the proposed HDNA using a Python implementation. We compare the power consumption and execution time of the HDNA architectures running on traditional CPU cores. We used an Intel core i7 7600 processor with 16 GB memory (4-core, 2.8GHz) to test different designs. Power consumption is measured by Hioki 3334 power meter. To estimate the cost of digital design, we also use a standard cell-based flow to design dedicated hardware for HDNA. We describe the proposed designs using RTL System-Verilog. For the synthesis, we use *Synopsys Design Compiler* with the TSMC 45 nm technology library, the general purpose process with high V_{TH} cells. We measured the power consumption of HD designs using *Synopsys PrimeTime* at (1 V, 25°C, TT) corner.
To assess the efficiency of proposed design, we apply the application of HDNA over two popular DNA classification datasets, *Empirical* [11] and *Molecular Biology* [12] datasets. Both datasets are split into two parts: 80% per species for training and 20% for testing.

B. HDNA Accuracy

We compare the classification accuracy of of HDNA and the state-of-the-art classification techniques over Empirical and Molecular biology dataset [13], listed in Table I and Table II respectively. HDNA using Encoder I ad Encoder II can achieve at least 5.21% and 4.87% higher classification accuracy as compared to prior techniques. For molecular biology dataset, our evaluation shows that HDNA using Encoder I can achieve comparable accuracy as other classification techniques while Encoder II can provide 100% classification accuracy. This accuracy is 5.87% higher than other classification algorithms.

In addition, we compare the efficiency of HDNA designs with SVM and K-NN designs. We run all algorithms implemented in python code on CPU over Empirical and Molecular biology datasets. Table I and Table II show the average energy consumption and execution times of different designs when a query runs on CPU cores. All algorithms are written to provide the maximum parallelism. Comparing HDNA design with prior work shows that HDNA using Encoder I (Encoder II) can achieve at least 2.98× (4.32×) speedup and 3.26× (2.05×) energy efficiency improvement over empirical dataset. Similarly, over molecular biology dataset, Encoder I (Encoder II) provides at least 4.38× (5.44%) speedup and 4.34× (2.47×) energy efficiency improvement as compare to other classification techniques. As traditional cores have not been designed to work with long hypervectors, we expect HDNA provides much more efficiency when it implements on digital RTL design. The following sections show the efficiency of HDNA design over digital implementation.

TABLE I
Accuracy and Efficiency of SVM, Bayes and the proposed HDNA over Empirical dataset (Encoder I with $n = 12$).

	Classes	SVM	Bayes	Encoder I	Encoder II
Accuracy	*Cypraeidae*	94.3%	93.2%	100%	100%
	Drosophila	98.3%	96.5%	100%	100%
	Inga	89.8%	91.5%	100%	100%
	Bats	100.0%	100.0	98.2%	100%
	Fishes	95.5%	97.3%	100%	95.2%
	Birds	98.4%	94.3%	99.7%	100%
	Fungi	80.0%	70.0%	100%	100%
	Algae	100.0%	100.0%	100%	100%
	Average	**94.53%**	**92.85%**	**99.74%**	**99.40%**
Energy Consumption (mJ)		62.03	47.51	14.53	23.16
Execution Time (ms)		2.77	1.73	0.58	0.44

TABLE II
Accuracy and Efficiency of K-NN, KBANN and HDNA over molecular biology dataset (Encoder I with $n = 10$)

	Classes	K-NN	KBANN	Encoder I	Encoder II
Accuracy	*Exon/Intron*	94.3%	93.2%	100%	96.7%
	Intron/Exon	98.3%	96.5%	100%	91.5%
	Neither	89.8%	91.5%	100%	92.15%
	Average	**94.13%**	**93.7%**	**100%**	**93.4%**
Energy Consumption (mJ)		46.60	42.56	9.79	17.21
Execution Time (ms)		2.07	1.36	0.31	0.25

Acknowledgements

This work was partially supported by CRISP, one of six centers in JUMP, an SRC program sponsored by DARPA, and also NSF grants #1730158 and #1527034.
DISTRIBUTION STATEMENT A. Approved for public release: distribution is unlimited.

References

[1] A. McKenna *et al.*, “The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data,” *Genome research*, vol. 20, no. 9, pp. 1297–1303, 2010.

[2] H. Erlich, *PCR technology: principles and applications for DNA amplification*. Springer, 2015.

[3] R. C. Green *et al.*, “Acmg recommendations for reporting of incidental findings in clinical exome and genome sequencing,” *Genetics in medicine: official journal of the American College of Medical Genetics*, vol. 15, no. 7, p. 565, 2013.

[4] M. Imani, T. Nassar, A. Rahimi, and T. Rosing, “Hdna: Energy-efficient dna sequencing using hyperdimensional computing,” in *Biomedical & Health Informatics (BHI), 2018 IEEE EMBS International Conference on*, pp. 271–274, IEEE, 2018.

[5] P. Kanerva, “Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors,” *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009.

[6] P. Kanerva *et al.*, “Random indexing of text samples for latent semantic analysis,” in *Proceedings of the 22nd annual conference of the cognitive science society*, vol. 1036, Citeseer, 2000.

[7] M. Imani *et al.*, “Voicehd: Hyperdimensional computing for efficient speech recognition,” in *IEEE International Conference on Rebooting Computing (ICRC)*, IEEE, 2017.

[8] M. Imani, C. Huang, D. Kong, and T. Rosing, “Hierarchical hyperdimensional computing for energy efficient classification,” in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2018.

[9] M. Imani *et al.*, “Exploring hyperdimensional associative memory,” in *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*, pp. 445–456, IEEE, 2017.

[10] M. Imani *et al.*, “Low-power sparse hyperdimensional encoder for language recognition,” *IEEE Design & Test*, vol. 34, no. 6, pp. 94–101, 2017.

[11] “Empirical datasets:.” http://dmb.iasi.cnr.it/supbarcodes.php.

[12] “Molecular Biology datasets.” https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences).

[13] E. Weitschek *et al.*, “Supervised dna barcodes species classification: analysis, comparisons and results,” *BioData mining*, vol. 7, no. 1, p. 4, 2014.