big data and cognitive computing

MDPI

Article

Margin-Based Training of HDC Classifiers

Laura Smets [1,*,†], Dmitri Rachkovskij [2,3,†], Evgeny Osipov [2,†], Werner Van Leekwijck [4], Olexander Volkov [3] and Steven Latré [1,4]

[1] IDLab—Department of Computer Science, University of Antwerp—imec, 2000 Antwerp, Belgium; steven.latre@uantwerpen.be
[2] Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, 971 87 Luleå, Sweden; dmitri.rachkovskij@ltu.se (D.R.); evgeny.osipov@ltu.se (E.O.)
[3] Institute of Information Technologies and Systems, 03187 Kyiv, Ukraine; alexvolk@ukr.net
[4] imec, 3000 Leuven, Belgium; werner.vanleekwijck@imec.be
* Correspondence: laura.smets@uantwerpen.be
† These authors contributed equally to this work.

Abstract: The explicit kernel transformation of input data vectors to their distributed high-dimensional representations has recently been receiving increasing attention in the field of hyperdimensional computing (HDC). The main argument is that such representations endow simpler last-leg classification models, often referred to as HDC classifiers. HDC models have obvious advantages over resource-intensive deep learning models for use cases requiring fast, energy-efficient computations both for model training and deploying. Recent approaches to training HDC classifiers have primarily focused on various methods for selecting individual learning rates for incorrectly classified samples. In contrast to these methods, we propose an alternative strategy where the decision to learn is based on a margin applied to the classifier scores. This approach ensures that even correctly classified samples within the specified margin are utilized in training the model. This leads to improved test performances while maintaining a basic learning rule with a fixed (unit) learning rate. We propose and empirically evaluate two such strategies, incorporating either an additive or multiplicative margin, on the standard subset of the UCI collection, consisting of 121 datasets. Our approach demonstrates superior mean accuracy compared to other HDC classifiers with iterative error-correcting training.

Keywords: hyperdimensional computing; HDC classifier; compositional representation; hypervector; margin classifier; confidence

Academic Editor: Alberto Abelló

Received: 30 October 2024
Revised: 23 January 2025
Accepted: 12 March 2025
Published: 14 March 2025

Citation: Smets, L.; Rachkovskij, D.; Osipov, E.; Van Leekwijck, W.; Volkov, O.; Latré, S. Margin-Based Training of HDC Classifiers. *Big Data Cogn. Comput.* **2025**, *9*, 68. https://doi.org/10.3390/bdcc9030068

1. Introduction

Simple linear classifiers are receiving increased attention for use cases requiring fast and energy-efficient computations. The co-design of linear classifiers and hardware, intensively researched within the framework of hyperdimensional computing (HDC), has already demonstrated promising results in this direction [1,2].

The basic HDC classifier is a simple classical prototype/centroid model [3,4]. In these prototype models, per-class data vectors of the training set are averaged to obtain a single prototype per class, possibly followed by a prototype quantization step. While this is a computationally efficient approach in terms of training and predicting [5–9], the classification results are modest, since the features (i.e., the individual components of data vectors) are not weighted to reflect the actual structure (e.g., covariance) of the classes.

HDC adds to the picture the non-linear transformation of vector data to a high-dimensional vector space, resulting in hypervectors. This is beneficial for linear classifica-

tion of input data with classes that are not linearly separable in the input space, as well as for other similarity-based tasks/applications. To improve performance, HDC classifiers are commonly trained using misclassified training samples [10–16]. The landscape of HDC applications for solving classification tasks is broad, starting from language detection with HDC-represented N-grams [17] to classification of arrhythmia types [6], time series classification [18,19], classification of medical data [20–22], and others [23] (see also [24–27] for a comprehensive overview).

Recent approaches to training HDC classifiers often involve various approaches to selecting the individual learning rate for only incorrectly classified samples. Unlike these approaches, we propose alternative strategies to make a decision about classification error using a margin on the classifier scores to improve test performance while using a basic learning rule with a fixed (and unit) learning rate. Class prototypes are thus adjusted to better classify not only wrongly classified samples but also correctly classified samples within the margin, so that a better classification performance is expected. Our contributions are as follows:

1. **Establishment of criteria and decision rules** to trigger the refinement of HDC classifiers, based on the confidence levels of individual sample classifications.
2. **Development of HDC classifiers** trained using either a multiplicative or additive margin applied to confidence scores, which operate on quantized class prototypes, and a comparative analysis of these methods.
3. **Configuration and enhancement of margin-based HDC classifiers** for scalability, enabling effective operation on larger and more diverse datasets, exemplified by experimentation on 121 UCI datasets.
4. **Comprehensive experimental evaluation** across 121 UCI datasets, demonstrating improved mean accuracy of the proposed HDC classifiers compared to other HDC models.
5. **Assessment of the impact of design choices** on the classification performance of the proposed HDC classifiers.

The paper is organized as follows. Section 2 provides background on HDC. Related works are presented and discussed in Section 3. The article continues by describing the methods and experiments in Sections 4 and 5, respectively. The results are presented in Section 6. Finally, Section 7 concludes this article.

2. Background on HDC and Basic HDC Classifiers

2.1. Formation of HDC Representations

In HDC, input data are transformed into a hyperdimensional vector (hypervector; HV) representation space in which the number of components of the HVs (i.e., the dimensionality) ranges from hundreds to tens of thousands. The key task in applications of HDC is to transform instances of the various fundamental data types of computer science, such as scalars, vectors, sequences, graphs, into HV representations. To be useful in various similarity-based tasks, the similarity of the obtained HVs should reflect the similarity of the input data objects (see Section 2.3).

These hypervectors could be real-valued, as in holographic reduced representations (HRRs) [28]; complex-valued, as in Fourier HRR [29]; bipolar vectors, as in multiply–add–permute (MAP) [30]; dense binary vectors, as in binary spatter code (BSC) [31,32]; or sparse binary vectors, as in sparse binary distributed representations (SBDRs) [33]. Each of these HDC frameworks has its own basic operations of binding and superposition; see, e.g., Section 2.2. Despite the heterogeneity of data transformation approaches, they can all be used in similar applications, such as solving classification tasks.

In this paper, we consider only dense binary HV representations, i.e., with vector components of either 0 or 1, and about half of the components are 1. This is the most commonly used variant because of its efficiency of working with binary vectors.

2.2. HDC Operations

Compositional HDC representations are produced using two fundamental HDC operations, *superposition* and *binding*, which are mathematical operators on the vector space. These operations do not change the dimensionality D of their input HVs.

The superposition operation *combines* several HVs into a single HV, which is used to represent (multi)sets. It is commonly implemented via component-wise addition. For dense binary HVs, component-wise addition is used to superimpose vectors followed by a majority vote (denoted as $[.]$) to binarize the vector components and where ties are broken pseudo-randomly. The binding operation *associates* HVs with each other. HDC allows considerable latitude in the details of binding implementation. For dense binary representations, the component-wise XOR operation (denoted as $\oplus$) is the common choice that we use here.

2.3. Hypervector Representation of Scalars and Numerical Vectors

In this paper, we are concerned with the representation of scalar values and numerical vectors and employ the compositional approach [34], proposed in [35–37], for the construction of their HVs.

Using this approach, it is desirable to map similar numerical values to similar HVs, rather than numerical values that are further apart; that is, the HVs should preserve the similarity between numerical values. One of the possible ways to transform numerical scalars to HVs is "linear mapping" [23,38]. Namely, after quantizing the numerical scalars to the range of integer values $[0, Q]$, the lowest quantization level is assigned a randomly generated HV. Then, an HV for each consecutive quantization level is obtained by flipping a certain fixed number of bits in the HV of the previous quantization level without flipping those bits that have already been flipped [23]. As a result, the similarity value to the lowest quantization level decreases linearly from the lowest level up to the highest level. Now, let us represent a numerical vector $\mathbf{x} \in \mathbb{R}^d$ consisting of d scalar features (i.e., $\mathbf{x} = [x_1, x_2, \ldots, x_d]$) as HV $\mathbf{v_x} \in \mathbb{B}^D$.

This is achieved in the following way:

1. Quantizing the numerical values of each feature to the range of values $[0, Q]$ such that $x \in [0, Q]^d$.
2. Representing the quantized feature values as their HVs $\mathbf{v}_{x_j}$ using linear mapping.
3. Assigning a random HV $\mathbf{v}_{f_j}$ to each of the d features f_j.
4. Binding each feature value HV with its corresponding feature HV using the component-wise XOR, i.e., $\mathbf{v}_{x_j} \oplus \mathbf{v}_{f_j}$.
5. Finally, obtaining the HV $\mathbf{v_x}$ by the addition of all d bound feature–value pairs followed by a majority vote:

$$\mathbf{v_x} = [\mathbf{v}_{x_1} \oplus \mathbf{v}_{f_1} + \mathbf{v}_{x_2} \oplus \mathbf{v}_{f_2} + \ldots + \mathbf{v}_{x_d} \oplus \mathbf{v}_{f_d}] = [\sum_{j=1}^{d} (\mathbf{v}_{x_j} \oplus \mathbf{v}_{f_j})]. \quad (1)$$

2.4. Basic HDC Classifiers

The simplest HDC classifier is prototype HDC classifier (Prot-HDCL), which creates K class prototypes (also referred to as centroids), with K being the number of classes in

the dataset. This is achieved by superimposing all sample HVs $\mathbf{v}_{\mathbf{x},i}$ of the training set that belong to the same class to obtain the class prototype for that class:

$$\mathbf{P}_k = \sum_{i=1}^{N} \{\mathbf{v}_{\mathbf{x},i} \,|\, y_i = k\} \qquad \text{for} \qquad k = 1 \ldots K. \tag{2}$$

In (2), y_i is the label of the ith sample's class. In the variant with further binarization of prototypes, a counter variable keeps track of the number of sample HVs that are included in each class prototype:

$$n_k = \sum_{i=1}^{N} I(y_i = k) \qquad \text{for} \qquad k = 1 \ldots K, \tag{3}$$

where $I(.)$ is the indicator function.

The class prototype $\mathbf{P}_k$ is then binarized with a majority vote by the threshold $n_k/2$:

$$\mathbf{p}_k = [\mathbf{P}_k]. \tag{4}$$

Those binarized prototypes are used for classification, i.e., class prediction on new input samples.

Building on the basic structure of the HDC classifier described earlier, various training strategies can be employed to enhance classification performance. The simplest of these is the prototype refinement rule, which we denote here as HDC classifier with prototype refinement (Ref-HDCL):

1. For the current training sample i, predict its class using the classifier.
2. If the prediction is incorrect, then update both the correct class prototype and the incorrect class prototype.
3. If the prediction is correct, leave all class prototypes unchanged.

An important variant of Ref-HDCL, which includes class prototype binarization, operates as follows: In step 1, class prediction is made by calculating the similarity values between the input sample's HV $\mathbf{v}_{\mathbf{x},i}$ and all binarized class prototypes $\mathbf{p}_k$. The predicted class $\hat{y}_i$ is the class associated with the binary prototype that has the highest similarity score to the sample's HV, calculated with the employed similarity measure:

$$\hat{y}_i = \text{argmax}_k \, \text{sim}(\mathbf{v}_{\mathbf{x},i}, \mathbf{p}_k) \qquad \text{for} \qquad k = 1 \ldots K. \tag{5}$$

In step 2, the update of class prototypes is triggered in case of a prediction mistake:

$$\hat{y}_i \neq y_i. \tag{6}$$

The following learning rule is applied to update the non-binary prototypes. The sample's HV is added to the prototype of the correct class $\mathbf{P}_{y_i}$ and subtracted from the prototype of the misclassified class $\mathbf{P}_{\hat{y}_i}$. Additionally, the class counters are updated by incrementing n_{y_i} and decrementing $n_{\hat{y}_i}$. This update process is described by the following expressions:

$$\mathbf{P}_{y_i} = \mathbf{P}_{y_i} + \mathbf{v}_{\mathbf{x},i}; \qquad \mathbf{P}_{\hat{y}_i} = \mathbf{P}_{\hat{y}_i} - \mathbf{v}_{\mathbf{x},i}. \tag{7}$$

$$n_{y_i} = n_{y_i} + 1; \qquad n_{\hat{y}_i} = n_{\hat{y}_i} - 1. \tag{8}$$

When all training samples are processed, and thus one epoch is completed, the obtained non-binary class prototypes $\mathbf{P}_k$ are binarized with the corresponding updated thresholds $n_k/2$ to produce binary class prototypes to be used in the next epoch. This

iterative training procedure is performed for a predefined number of epochs or until a predefined accuracy on the training set is reached.

After training completion, the binarized class prototypes are used for prediction. As stated above, classification in HDC is achieved based on the similarity values (i.e., class scores) between the input sample HV and the *binarized* class prototypes. The choice of the similarity measure (sim) employed influences the classification results.

3. Related Work

3.1. Prototype Classifiers

The simplest form of an HDC classifier, as described in [23], is the prototype/centroid classifier [3,4]. In this approach, class prototypes are constructed by averaging the data vectors for each class in the training set, producing a single prototype per class, with an option to quantize these prototypes. We refer to this approach as Prot-HDCL, as mentioned above. While this method is computationally efficient for both training and prediction, its classification performance is often modest. This is due to the fact that the features (or components of the data HVs) are not weighted according to their relative importance in distinguishing between classes.

3.2. Prototype Refinement

Several variations of HDC classifiers employing "prototype refining" approaches essentially implement different versions of the perceptron algorithm. In VoiceHD [39], error correction is applied with a single pass through the dataset, resulting in non-binary (integer) weights. In contrast, ref. [40] explores an approach where the weights are binarized after multiple training epochs.

In BinHD [41], binary class prototypes are used for prediction, but in the event of a classification error, the non-binary prototypes are updated and subsequently binarized. This method also uses multiple training epochs. Similarly, QuantHD [42] maintains two sets of prototypes—one non-quantized and one quantized—with quantization occurring after each epoch. Quantization in this case can include either binarization or ternarization.

This approach, referred to as Ref-HDCL above, has been employed in many HDC-related studies, such as [43]. It does not incorporate the notion of confidence in prediction.

3.3. Classification Scores and Confidence in Classification Results

In the data-dependent mode of AdaptHD [44], the learning rate is adjusted based on the difference between class scores for the correct and misclassified classes. After training, the class prototypes are binarized. In OnlineHD [45], non-binary data HVs are used to improve classification accuracy, with no mention of prototype quantization. Like AdaptHD [44], the learning rate in OnlineHD [45] is adapted according to the confidence in the class scores.

In TD-HDC [46], both integer and binary prototypes are supported. During training, integer prototypes are updated in case of a misclassification by either the binary or integer classifier, with final binarized prototypes obtained after training. During inference, they use a "classification confidence" measure, defined as the difference between the two smallest Hamming distances between the input HV and the class prototypes of the binary classifier. If this confidence value falls below a predefined threshold (or margin), the integer classifier is employed to make the prediction. Thus, all of the HDC classifiers mentioned above update prototypes only when the highest class score is provided for an incorrect class.

In RefineHD [47], the authors also mention using the difference between the top two class scores for the prototype update decision.

The works most relevant to this paper in the context of HDC classification are Confidence Centroid (CONFcentroid) [48] and linear classifiers [49–51]. These approaches utilize

the confidence in class scores to determine when to update class prototypes. However, the mentioned linear classifiers do not address the case of binary prototypes, and none of the previous works explore setups involving multiple diverse datasets.

3.4. Other HDC Classifiers

In the context of HDC classifiers, the use of randomized non-linear transformations of input data into HVs followed by a linear classifier was initially proposed in [35,36] for a distributed representation of class labels. Since then, large-margin linear classifiers have been employed, with their first explicit mention in English appearing in [49,50], and further development in [51]. For comprehensive reviews on HDC classifiers, please see [25–27].

4. Methods

In this work, we propose a novel approach for improving the quality of HDC classifiers by introducing unified criteria for deciding when to update class prototypes during the training process. The decision rule for refining class prototypes given the HV of the input data sample during training is different from that of Ref-HDCL as presented above in expressions (5) and (6).

The different decision rules are based on the notion of the confidence level c_i with which the input sample i belonging to class y_i is classified given its HV $\mathbf{v}_{\mathbf{x},i}$. It is defined as the difference between the similarity of the data sample HV and the correct class prototype and its similarity to the most similar class prototype of any other class:

$$c_i = \mathrm{sim}(\mathbf{v}_{\mathbf{x},i}, \mathbf{p}_{y_i}) - \max_{k \neq y_i} \mathrm{sim}(\mathbf{v}_{\mathbf{x},i}, \mathbf{p}_k). \tag{9}$$

Here, $\mathrm{sim}(\mathbf{v}_{\mathbf{x},i}, \mathbf{p}_k)$, $k = 1 \ldots K$, are the class scores given the input HV $\mathbf{v}_{\mathbf{x},i}$ to be classified. They evidently depend on the employed similarity measure (sim).

Then, various decision rules for prototype updates can be constructed by comparing the confidence margin value with specific threshold values, which can be defined in different ways. It is important to note that while the confidence margin c_i is calculated based on the similarity between the input HV and the binarized class prototypes, the updates are performed on the second set of non-binarized prototypes (which are integer in the setup we consider, though other setups may use different representations).

In the following subsections, we present two instances of our approach, each resulting in different update decision rules. Additionally, we outline essential design choices for the deployment of margin-based HDC classifiers based on our approach.

4.1. HDC Classifiers with Additive Margin

The additive margin strategy results in a family of HDC classifiers with additive margin (AM-HDCL). An instance of this family is the CONFcentroid classifier [48]. AM-HDCL will not update the class prototypes in the training mode when

$$c_i > \alpha, \;\; 0 \leq \alpha < 1, \tag{10}$$

where α is some constant predefined threshold. Therefore, to avoid the prototype update, the difference in class scores of the correct class and any other class should not only be positive, but also greater than α. In case of an update (when $c_i \leq \alpha$), it is performed by (7) and (8), with $\hat{y}_i = \mathrm{argmax}_{k \neq y_i} \mathrm{sim}(\mathbf{v}_{\mathbf{x},i}, \mathbf{p}_k)$. This is different from the update decision rule of Ref-HDCL, where correct classification when training (and hence no update of class prototypes) requires $c_i > 0$, taking into account (5) and (6).

In the case of $\alpha = 0$, AM-HDCL becomes Ref-HDCL. However, when $\alpha > 0$, we expect more updates from AM-HDCL, as some samples that Ref-HDCL would not update

are treated as requiring an update by AM-HDCL. Evaluation of the trained AM-HDCL classifier is performed by (5) and (6), equivalent to setting $\alpha = 0$ in (10).

4.2. HDC Classifiers with Multiplicative Margin

The multiplicative margin strategy results in a family of HDC classifiers with multiplicative margin (MM-HDCL). It is instantiated by the classifiers considered in [49–51]. MM-HDCL avoids update only in the case that

$$c_i > \alpha \cdot \mathrm{sim}(\mathbf{v}_{\mathbf{x},i}, \mathbf{p}_{y_i}), \quad 0 \leq \alpha < 1. \tag{11}$$

That is, the similarity margin should be larger than a fraction α of the correct class score.

This approach enables more frequent weight updates compared to Ref-HDCL, converging slower but towards a solution that ensures a relative (multiplicative) margin. Prototype updates follow the same procedure as in Ref-HDCL, which is also equivalent to updates in AM-HDCL. Setting $\alpha = 0$ recovers the behavior of Ref-HDCL.

When the training set is correctly classified with $\alpha > 0$, we anticipate an improved test set accuracy for AM-HDCL and MM-HDCL compared to a correct classification with $\alpha = 0$, implying better generalization performance [52,53].

4.3. Design Choices

The performance of machine learning models is influenced by specific design choices made during the implementation of their methods. In this section, we isolate particular design choices for the methods applied at different stages of HDC classifiers.

4.3.1. Input Data Preprocessing/Standardization

Input data preprocessing and standardization are essential for the performance of machine learning models, including HDC classifiers.

The initial step is standardizing input features before transforming them into HVs. A common approach is per-feature z-score standardization, which scales each feature to have a mean of 0 and a standard deviation of 1, preserving the relative positioning of data points within their distributions. This method is useful for datasets that approximate a normal distribution and when there are outliers or features with varying variances.

After standardization, input data should be quantized in order to apply the compositional method for transforming input data into hypervectors.

A further decision involves how to standardize the resulting HVs of data samples.

4.3.2. Initialization of Class Prototypes

An important factor influencing the performance of HDC classifiers is the choice of how to initialize class prototypes before starting error-driven iterative training. This concerns the two sets of prototypes: integer and quantized. The three straightforward options are (1) initializing with zeros, (2) initializing randomly, and (3) initializing with "incremental" prototypes resulting from Prot-HDCL.

4.3.3. Quantization of Class Prototypes

The primary distinction between the HDC classifiers being examined and other linear classifiers lies in the learning of binary prototypes. This is accomplished by maintaining two sets of prototypes: an "original" set, similar to the weights in traditional linear classifiers; and a quantized (binarized, in this paper) set. Unlike post-training quantization of prototypes, which can significantly degrade accuracy, we apply quantization during the training process, as explained in Section 2.4.

This opens up several related design choices, from which we here consider just two: when to quantize; and how to quantize. The simplest alternative for *when* to quantize is quantization after each epoch, as given in Section 3 and employed in the CONFcentroid model [48]. The primary alternatives for *how* to quantize involve component-wise thresholding with a zero threshold, or component-wise thresholding using half-values of per-class counters (8).

4.3.4. Similarity Measures Between Hypervectors and Prototypes

Linear models are evidently based on the dot-product similarity $\text{sim}_{\text{dot}}(\mathbf{a}, \mathbf{b}) = \langle \mathbf{a}, \mathbf{b} \rangle = \sum_{j=1}^{D} a_j \; b_j$. On the other side, a number of HDC classification models operating with binary HVs and prototypes, including CONFcentroid as the instance of AM-HDCL, use the Hamming distance dist_{Ham} to compute class scores: $\text{dist}_{\text{Ham}}(\mathbf{a}, \mathbf{b}) = \sum_{j=1}^{D} I(a_j \neq b_j)$, where $I(.)$ is the indicator function.

Here, we analyze the implications for ranking the class prototypes when using sim_{dot} versus dist_{Ham}. Consider the following identities for binary $\{0, 1\}$ vectors:

$$\text{dist}_{\text{Ham}}(\mathbf{a}, \mathbf{b}) = \text{dist}^2_{\text{Euc}}(\mathbf{a}, \mathbf{b}) = ||\mathbf{a} - \mathbf{b}||^2 = ||\mathbf{a}||^2 + ||\mathbf{b}||^2 - 2\langle \mathbf{a}, \mathbf{b} \rangle, \tag{12}$$

where $||\mathbf{a}||^2 = |\mathbf{a}|$ is equal to the number of ones in $\mathbf{a}$. The Hamming similarity is then

$$\text{sim}_{\text{Ham}}(\mathbf{a}, \mathbf{b}) = D - \text{dist}_{\text{Ham}}(\mathbf{a}, \mathbf{b}). \tag{13}$$

Taking into account that $\text{sim}_{\text{dot}}(\mathbf{a}, \mathbf{b}) = \langle \mathbf{a}, \mathbf{b} \rangle = \sum_{j=1}^{D} I(a_j = 1 \;\&\; b_j = 1) = |\mathbf{a}\&\mathbf{b}|$, with & being the component-wise conjunction, we obtain

$$2\langle \mathbf{a}, \mathbf{b} \rangle = |\mathbf{a}| + |\mathbf{b}| - \text{dist}_{\text{Ham}}(\mathbf{a}, \mathbf{b}). \tag{14}$$

Therefore, generally, the ranking of class prototypes by the value of $\langle \mathbf{a}, \mathbf{b} \rangle$ would not be identical to that by $-\text{dist}_{\text{Ham}}(\mathbf{a}, \mathbf{b})$ due to varied $|\mathbf{a}| + |\mathbf{b}|$. On the other hand, those rankings by sim_{dot} and sim_{Ham} would be identical if the binarization of prototypes and encodings of input vectors ensured a constant value of the sum of the number of ones in them: $|\mathbf{a}| + |\mathbf{b}| = const$. The most straightforward way to ensure this is to binarize both prototypes and sample HVs such that the number of ones in both is $D/2$. Then, the total number of ones is $|\mathbf{a}| + |\mathbf{b}| = D$, and we obtain

$$2\langle \mathbf{a}, \mathbf{b} \rangle = \text{sim}_{\text{Ham}}(\mathbf{a}, \mathbf{b}). \tag{15}$$

In this case, the ranking by similarity values of sim_{dot} and sim_{Ham} is identical. Note that in dense binary randomly generated vectors, $p(1) = 1/2$, hence $\text{mean}(|\mathbf{a}|) = D/2$. The coefficient of variation of $|\mathbf{a}|$ decreases as $1/\sqrt{D}$, and so with increasing D, $|\mathbf{a}| + |\mathbf{b}|$ becomes relatively closer to D, and we could expect closer similarity ranking results for HVs of larger D. Similar reasoning is also applicable to real-valued vectors, which we do not consider in this paper.

Importantly, in the case of bipolar $\{-1,+1\}$ HVs, which are often used in HDC instead of binary $\{0,1\}$, we have $||\mathbf{a}||^2 = D$ and $4\text{dist}_{\text{Ham}}(\mathbf{a}, \mathbf{b}) = \text{dist}^2_{\text{Euc}}(\mathbf{a}, \mathbf{b}) = ||\mathbf{a} - \mathbf{b}||^2$. Therefore, we obtain exact equality $4\text{dist}_{\text{Ham}}(\mathbf{a}, \mathbf{b}) = 2D - 2\langle \mathbf{a}, \mathbf{b} \rangle$, $2\text{dist}_{\text{Ham}}(\mathbf{a}, \mathbf{b}) = D - \langle \mathbf{a}, \mathbf{b} \rangle$ and, as such, identical ranking for any bipolar HVs of any D.

4.3.5. The Training Procedure

For both AM-HDCL and MM-HDCL, the simplest update rule we use is given by (7). However, decisions regarding the stopping criterion and selection of class prototypes for the evaluation phase play a crucial role in achieving robust classifier performance.

A straightforward stopping criterion is to halt training after a predefined number of epochs. Another common approach is early stopping, triggered when a specified accuracy threshold on the training set is reached.

Upon completion of training, the choice of which set of class prototypes to use in evaluation becomes pertinent. While it is straightforward to use the final prototypes obtained at the end of training, this can lead to challenges, especially in determining the optimal number of training epochs on a per-dataset basis. This issue is exacerbated when handling two sets of class prototypes, one integer and one binary.

4.3.6. The Choice of Design Setup for Experiments

In this section, we outline the finalized design setup chosen for our instantiations of AM-HDCL and MM-HDCL, selected from the numerous combinations of design choices, as defined above.

1. **Input data standardization**: Per-feature z-score standardization. In preliminary evaluations, we investigated more intensive approaches for managing outliers; however, these yielded inconsistent effects across datasets. Consequently, we adopt the standard z-score approach for consistency.
2. **Input data quantization**: Uniform quantization into the [0, Q] range. We use $Q = 100$.
3. **Input data transformation to hypervectors** (Section 2.3): Here, instead of flipping $D/(2Q)$ bits per quantization level as in our previous work [48], we adopt a scheme that flips D/Q bits to generate HVs for scalar values. This effectively halves the dimensionality of the hypervectors.
4. **Hypervector standardization**: We maintain the binary format of generated HVs by avoiding further standardization.
5. **Prototypes initialization**: Before starting error-driven iterative training to refine class prototypes, we initialize the two sets of prototypes (integer and quantized) using the initial "incremental" prototypes from Prot-HDCL.
6. **The similarity measures between HVs and class prototypes**: For AM-HDCL, we use the Hamming similarity normalized by the HV dimensionality, and for MM-HDCL, we apply dot-product similarity.
7. **Selection of α values**: The hyperparameter α is evaluated over the range of [0.00, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.7] for both AM-HDCL and MM-HDCL.
8. **Integer prototype update rule**: According to rule (7).
9. **Binary prototype update rule**: Binarization of integer prototypes is applied after each epoch. For AM-HDCL, binarization is achieved by thresholding based on class counter values, while for MM-HDCL, the thresholding is performed with zero values.
10. **Training stopping**: In contrast to [48], we significantly reduce the number of training epochs from 2500 to 100, making training feasible for the large collection of 121 UCI datasets. Additionally, we halt training early if the training accuracy reached 0.9999.
11. **Selection of class prototypes for deployment**: Upon completion of the training, we do not use the final class prototypes for testing. Nor do we apply any procedure or validate some hyperparameters to select some intermediate set of prototypes. Instead, we select the prototypes obtained at the epoch with the highest training accuracy, as in our previous work [48]. While this method may not always be optimal (we observed that maximum testing accuracy often occurs in earlier epochs, before potential "overtraining"), it remains reasonable since prototypes obtained after the training completion may lead to worse performance.

5. Experiments

The AM-HDCL and MM-HDCL instances are distinguished by their respective strategies based on the different types of confidence margin, as well as the design choices specified above. In this section, we describe the experimental setup for evaluating the performance of AM-HDCL and MM-HDCL. The performance was measured as classification accuracy. For performance comparison, we include AM-HDCL with $\alpha = 0$ (AM-HDCL $\alpha = 0$) and MM-HDCL with $\alpha = 0$ (MM-HDCL $\alpha = 0$) as baselines, along with corresponding versions of Prot-HDCL, which are prototype classifiers without prototype refinement.

We experimented with 121 UCI benchmark datasets from [54] (see Appendix A), containing the training/validation/testing dataset splits of 121 UCI datasets proposed in [54]. Though those splits themselves have drawbacks from the point of view of classic ML basic principles (see also [55]), they are widely used and selected by us to ensure better compatibility with the existing performance results.

For the encoding part, we experimented with the influence of the dimensionality $D = \{200, 1000, 4000, 10{,}000\}$ of HVs on the classification performance.

Since HDC relies on random initialization of HVs, which influences performance, experiments with HDC were repeated for multiple realizations of randomly generated HVs, i.e., with multiple seeds for initializing a random number generator. For each seed, we transformed the initial data from the splits into their HVs. We conducted experiments with ten different seeds for each classification setup/configuration.

To select the α value to be used for training/testing, we trained/tested on the HVs of the validation split obtained for each seed at each combination of hyperparameter value from its range at the particular design setup. Thereafter, the obtained validation accuracies for each design setting/hyperparameter value were averaged across all seeds. The value of α providing the highest mean validation accuracy was selected for each dataset.

In the final training/testing, the selected hyperparameter value was used to train and test the final (selected) model for each seed. As such, a test accuracy averaged across the multiple seeds (for the "separated" datasets) or a test accuracy averaged across the four test folds and multiple seeds (for the "non-separated" datasets) was obtained for each dataset.

Setting the single value $\alpha = 0$ gave us marginless versions AM-HDCL $\alpha = 0$ and MM-HDCL $\alpha = 0$ of AM-HDCL and MM-HDCL, respectively. Since the AM-HDCL $\alpha = 0$ and MM-HDCL $\alpha = 0$ classifiers did not need selection of the hyperparameter α, only their final training/testing was performed.

6. Results and Discussion

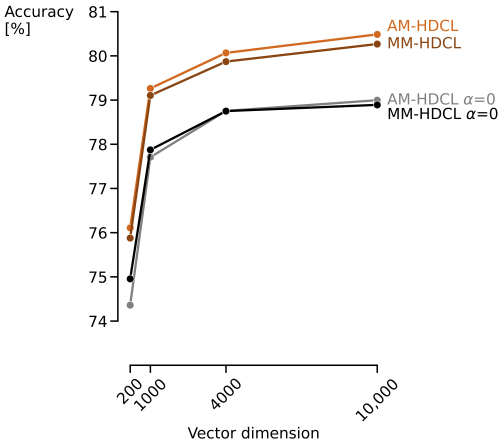
Table 1 reports the mean test classification accuracies of the Prot-HDCL sim_{Ham}, Prot-HDCL sim_{dot}, AM-HDCL $\alpha = 0$, MM-HDCL $\alpha = 0$, AM-HDCL, and MM-HDCL classifiers, averaged across all 121 UCI datasets and seeds (detailed results per dataset can be found in Appendix B). Figure 1a further shows how varying the HV dimensionality D influences the final test accuracy, corresponding to the data in Table 1. Figure 1b presents the mean number of training epochs required to reach the best training accuracy, while Figure 1c provides the mean training time per epoch measured when running our python scripts in a distributed system of a set of heterogeneous clusters (consisting of NVIDIA A100 80 GB PCIe, Quadro RTX 4000, Tesla V100-SXM2-32 GB and Tesla V100-SXM3-32 GB) using five CPUs.

As indicated in Table 1 and Figure 1a, the mean classification accuracy improves with increasing HV dimensionality. The results for both AM-HDCL $\alpha = 0$ and MM-HDCL $\alpha = 0$, as well as AM-HDCL and MM-HDCL, are quite similar, with AM-HDCL and MM-HDCL consistently outperforming their marginless variants. Notably, even the marginless pro-

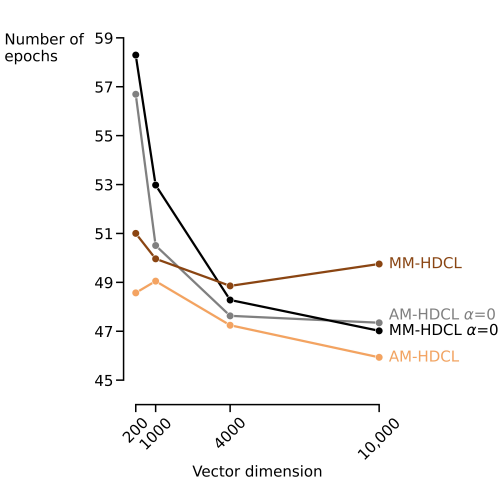
totype refinement classifiers AM-HDCL $\alpha = 0$ and MM-HDCL $\alpha = 0$ yield substantially higher accuracy than the Prot-HDCL variants.

Table 1. Mean test accuracies (in %) across all 121 datasets and seeds, reported for the best α value obtained through validation. The results are presented for Prot-HDCL sim_{Ham}, Prot-HDCL sim_{dot}, AM-HDCL $\alpha = 0$, MM-HDCL $\alpha = 0$, AM-HDCL, and MM-HDCL at various HV dimensionalities D. Standard deviations are provided in brackets, with the highest mean accuracy in bold.

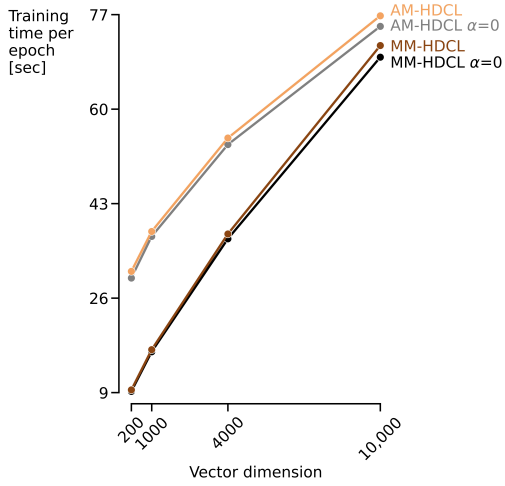
Classifier \ D	200	1000	4000	10,000
Prot-HDCL sim_{Ham}	66.36 (±16.28)	69.70 (±15.96)	70.66 (±15.98)	70.92 (±16.05)
Prot-HDCL sim_{dot}	61.36 (±19.10)	65.73 (±19.20)	66.99 (±19.46)	67.43 (±19.35)
AM-HDCL $\alpha = 0$	74.36 (±17.11)	77.71 (±16.64)	78.75 (±16.45)	79.00 (±16.55)
MM-HDCL $\alpha = 0$	74.95 (±16.90)	77.87 (±16.45)	78.75 (±16.27)	78.89 (±16.32)
AM-HDCL	76.11 (±16.18)	79.26 (±15.62)	80.07 (±15.59)	**80.49** (±15.40)
MM-HDCL	75.88 (±16.54)	79.10 (±15.59)	79.87 (±15.61)	80.27 (±15.25)



(**a**) Accuracy.



(**b**) Number of training epochs.



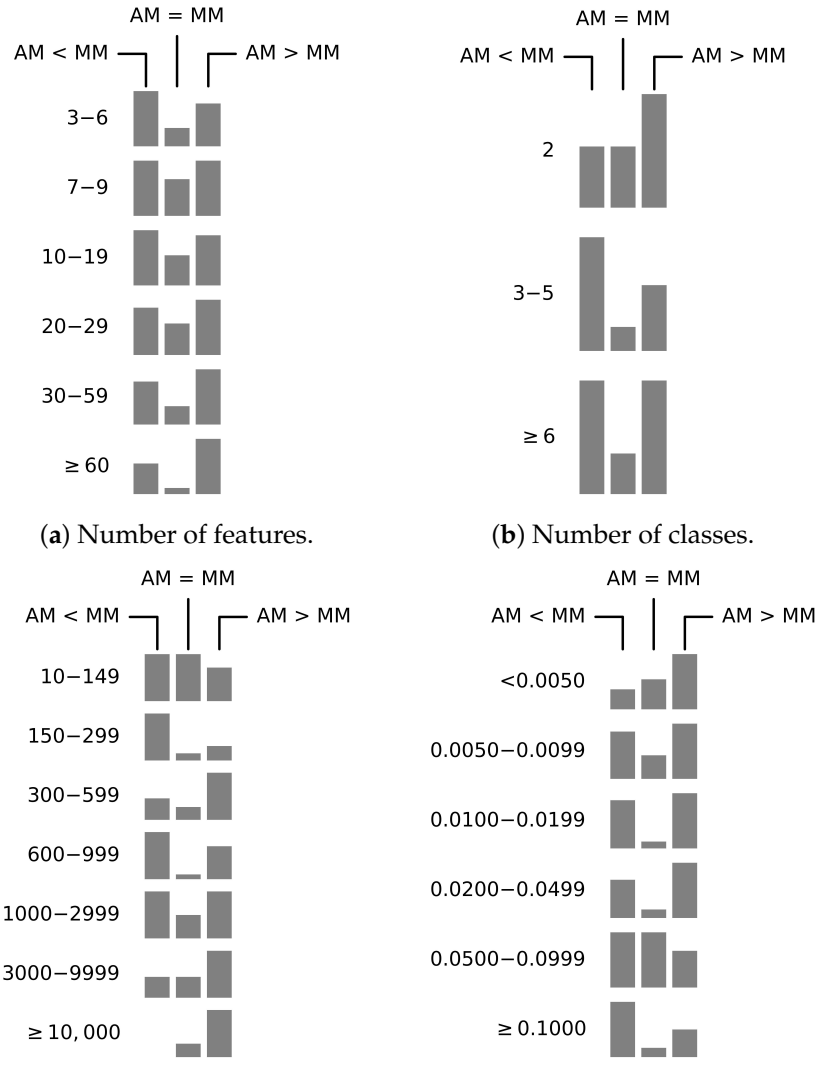
(**c**) Training time.

Figure 1. Impact of HV dimensionality D on (**a**) the mean final test accuracy (as detailed in Table 1), (**b**) the mean number of training epochs required to achieve the best training accuracy, and (**c**) the mean time per epoch (in seconds). All values are averaged across 121 UCI datasets and ten seeds for AM-HDCL $\alpha = 0$, MM-HDCL $\alpha = 0$, AM-HDCL, and MM-HDCL.

Figure 1b,c illustrate that the mean number of training epochs needed to achieve the highest accuracy on the training set decreases with increasing HV dimensionality. Furthermore, AM-HDCL and MM-HDCL generally require fewer epochs compared to the margin-

less variants AM-HDCL $\alpha = 0$ and MM-HDCL $\alpha = 0$. In contrast, the mean training time increases significantly with HV dimensionality. Although AM-HDCL $\alpha = 0$ and AM-HDCL generally take longer to train than MM-HDCL $\alpha = 0$ and MM-HDCL, this time difference diminishes as the HV dimensionality increases. Notably, with $D = 10{,}000$, the additional time required results in only a marginal accuracy improvement over $D = 4000$. Thus, $D = 4000$ may provide a practical trade-off between accuracy and computational complexity.

Figure 2 presents a comparison between AM-HDCL and MM-HDCL at $D = 10{,}000$, indicating the number of datasets where AM-HDCL performs better, equally well, or worse than MM-HDCL. The datasets are grouped based on the number of features, number of classes, number of samples, and the ratio of the number of features to the number of samples. The comparison shows that AM-HDCL outperforms MM-HDCL on datasets with more than 30 features, on datasets with two classes, and on datasets with more than 3000 samples. Additionally, AM-HDCL achieves better performance on datasets with a feature-to-sample ratio smaller than 0.05.



(**a**) Number of features. (**b**) Number of classes.

(**c**) Number of samples. (**d**) Ratio num features/num samples.

Figure 2. Comparison of classification results for AM-HDCL and MM-HDCL at $D = 10{,}000$ across 121 UCI datasets, categorized by (**a**) the number of features, (**b**) the number of classes, (**c**) the number of samples, and (**d**) the ratio of the number of features to the number of samples.

Figure 3 compares the performance of AM-HDCL and MM-HDCL at $D = 10{,}000$ against other HDC classifiers on the 121 UCI datasets. The results from [27,47] are displayed

in gray, while those from [56,57] are shown in light gray. The figure highlights that AM-HDCL and MM-HDCL outperform other HDC classifier variants. Specifically, the highest-performing HDC classifier in the literature, RefineHD [47], achieves a mean accuracy of 76.7%. In comparison, AM-HDCL and MM-HDCL achieve accuracies of 80.49% and 80.27%, respectively. Notably, AM-HDCL reaches an accuracy of 76.11% even with low HV dimensionality, $D = 200$ (Table 1).

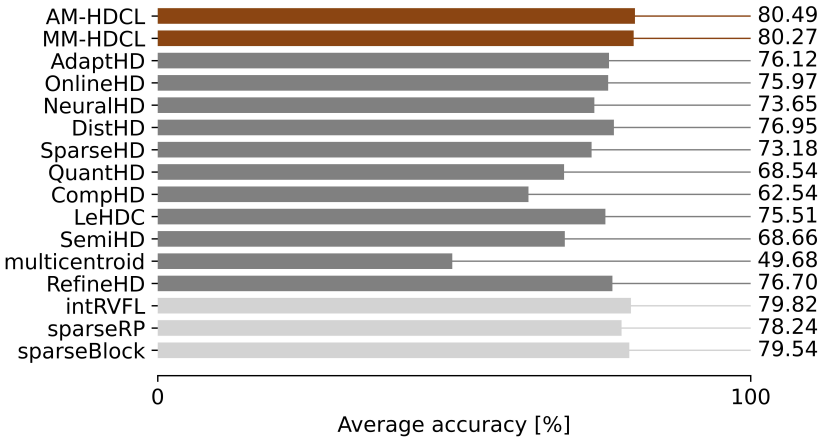


Figure 3. Comparison of the best mean accuracy of AM-HDCL and MM-HDCL (shown in red) and other HDC classifiers. Results from [27,47] are shown in gray, while those from [56,57] are shown in light gray.

Additionally, it is worth noting that our results are better than previously reported accuracies obtained using Ridge classifier (79.82% for IntRVFL [56], 78.24% for sparseRP [57], and 79.54% for SparseBlock [57]). Ridge classifier utilizes real-valued instead of binary prototypes, as well as integer-valued rather than binary hypervectors. The closeness of their results to each other also underscores the difficulty in improving upon this task for the HDC approach.

Our HDC classifiers leverage binary class prototypes and hypervectors, enabling extremely efficient deployment. In prediction mode, the core operation reduces to simply counting the number of 1-bits in machine words—a computation highly optimized on modern hardware and efficiently implemented on chips. This efficiency is particularly beneficial in scenarios with limited computational resources or where online and real-time performance is required. Moreover, the inherent simplicity of binary representations and the lightweight nature of HDC classifiers address the growing demand for fast, resource-constrained machine learning solutions, such as those needed for IoT, EdgeAI, and AIoT applications.

However, HDC classifiers with binary vectors and prototypes require larger dimensionality to achieve high classification performance and currently do not reach the top results achieved by some non-HDC classifiers. Additionally, there remains a notable gap in the theoretical understanding of these methods.

In contrast, non-HDC classifiers typically rely on real-valued vector representations and classifier parameters, with algorithms that demand significantly greater computational resources and extensive training. Although these models may achieve higher performance—as demonstrated by the top classifiers on the 121 UCI datasets in [54], including non-linear SVMs, Random Forests, and Boosting methods—they are tailored for applications where high predictive accuracy is paramount and ample computational power is available. Furthermore, deep neural networks require large datasets for extensive co-training of features and classifiers, and they are commonly employed in complex tasks like

image recognition and natural language processing, where the extra computational cost and energy consumption are acceptable trade-offs for enhanced accuracy.

7. Conclusions

This work proposed an approach to training HDC classifiers which leverages confidence-triggered refinement of class prototypes based on certainty levels in individual sample classifications. The enhancement of classification accuracy is demonstrated in comparison with results obtained from classical marginless HDC counterparts and has been validated across a diverse set of 121 UCI datasets. We show that applying either additive or multiplicative margin-based training strategies results in a good performance, with the additive margin configuration yielding slightly higher results. Both strategies consistently outperform alternative modifications of classical HDC classifiers, given the results available in the literature on the same set of UCI datasets.

Additionally, we achieved better results compared to the HDC approach using a Ridge classifier, which operates with real-valued prototypes and integer-valued hypervectors rather than binary ones.

By operating with quantized and ultimately binary data and class prototype vector representations—particularly in prediction mode—our approach remains inherently fast, hardware- and energy-efficient, and well suited for resource-constrained, online, and real-time classification applications.

Promising directions for future research include exploring the combination of our confidence margin-based training with existing HDC classifiers that incorporate learning rate adjustments, which could potentially complement each other to further enhance performance. Another important avenue is investigating whether this or other improvements to HDC classifiers could enable them to achieve the performance of top non-HDC classifiers on the 121 UCI datasets while preserving the efficiency benefits of binary vectors and weights. Additionally, developing hardware implementations optimized in terms of gate efficiency and energy consumption presents a valuable direction for further research. Conducting a theoretical analysis of the proposed approach would also provide deeper insights into its fundamental properties and potential improvements.

Author Contributions: Conceptualization, L.S., D.R., E.O., O.V. and W.V.L.; methodology, L.S., D.R., E.O. and W.V.L.; software, L.S. and D.R.; validation, L.S., D.R., E.O. and W.V.L.; formal analysis, L.S. and D.R.; investigation, L.S. and D.R.; writing—original draft preparation, L.S., D.R. and E.O.; writing—review and editing, W.V.L., O.V. and S.L.; visualization, L.S.; supervision, S.L.; funding acquisition, S.L. All authors have read and agreed to the published version of the manuscript.

Funding: The work of L.S. and W.V.L. was supported by the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" program. The work of D.R. was supported in part by the Swedish Foundation for Strategic Research (SSF, grant nos. UKR22-0024, UKR24-0014), the Swedish Research Council (VR) Scholars at Risk (SAR) Sweden (VR SAR grant no. GU 2022/1963), the National Research Fund of Ukraine (NRFU, grant no. 2023.04/0082), and LTU support grant. The work of E.O. was supported in part by the Swedish Research Council (VR grant no. 2022-04657). E.O. and D.R. acknowledge the computational resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725, as well as help and support of Denis Kleyko and Philip Gard.

Data Availability Statement: The datasets for this study can be found in the original article of [54].

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AM-HDCL	HDC classifier with additive margin
AM-HDCL $\alpha = 0$	AM-HDCL with $\alpha = 0$
HDC	Hyperdimensional computing
HV	Hypervector
MM-HDCL	HDC classifier with multiplicative margin
MM-HDCL $\alpha = 0$	MM-HDCL with $\alpha = 0$
Prot-HDCL	Prototype HDC classifier
Ref-HDCL	HDC classifier with prototype refinement

Appendix A. Details on Datasets

We evaluate classifiers on 121 UCI repository datasets, as introduced in [54], which are publicly available. Table A1 contains descriptions of all 121 datasets, including the total number of samples, the number of features, the number of classes, the ratio of the number of features to the number of samples, and the mean number of samples per class. For 19 of these 121 datasets, there are fixed training and testing sets, so we refer to them as "separated" and mark them in gray in the table. The other 102 datasets do not have separated training/testing sets. For both types of datasets, the procedures for splitting them into subsets for hyperparameter selection (in our case, α) and final evaluation as described in [54] are followed. More specifically, training and validation sets are generated randomly, each consisting of 50% of the samples making sure all classes are balanced between the two sets. The final evaluation is performed by a 4-fold cross validation with the whole dataset for the "non-separated" datasets, or by using the fixed training and test set of the "separated" datasets.

Table A1. Description of the 121 UCI datasets. The table includes the total number of samples, the number of features, the number of classes, the ratio of the number of features to the number of samples, and the mean number of samples per class.

ID	Dataset	No. Samples	No. Features	No. Classes	Ratio Feat/Samples	Avg Samples per Class
1	abalone	4177	8	3	0.0019	1392
2	acute-inflammation	120	6	2	0.0500	60
3	acute-nephritis	120	6	2	0.0500	60
4	adult	48,842	14	2	0.0003	24,421
5	annealing	898	31	5	0.0345	180
6	arrhythmia	452	262	13	0.5796	35
7	audiology-std	196	59	18	0.3010	11
8	balance-scale	625	4	3	0.0064	208
9	balloons	16	4	2	0.2500	8
10	bank	4521	16	2	0.0035	2260
11	blood	748	4	2	0.0053	374
12	breast-cancer	286	9	2	0.0315	143
13	breast-cancer-wisc	699	9	2	0.0129	350
14	breast-cancer-wisc-diag	569	30	2	0.0527	284
15	breast-cancer-wisc-prog	198	33	2	0.1667	99
16	breast-tissue	106	9	6	0.0849	18
17	car	1728	6	4	0.0035	432
18	cardiotocography-10clases	2126	21	10	0.0099	213
19	cardiotocography-3clases	2126	21	3	0.0099	709
20	chess-krvk	28,056	6	18	0.0002	1559
21	chess-krvkp	3196	36	2	0.0113	1598
22	congressional-voting	435	16	2	0.0368	218

Table A1. *Cont.*

ID	Dataset	No. Samples	No. Features	No. Classes	Ratio Feat/Samples	Avg Samples per Class
23	conn-bench-sonar-mines-rocks	208	60	2	0.2885	104
24	conn-bench-vowel-deterding	990	11	11	0.0111	90
25	connect-4	67,557	42	2	0.0006	33,778
26	contrac	1473	9	3	0.0061	491
27	credit-approval	690	15	2	0.0217	345
28	cylinder-bands	512	35	2	0.0684	256
29	dermatology	366	34	6	0.0929	61
30	echocardiogram	131	10	2	0.0763	66
31	ecoli	336	7	8	0.0208	42
32	energy-y1	768	8	3	0.0104	256
33	energy-y2	768	8	3	0.0104	256
34	fertility	100	9	2	0.0900	50
35	flags	194	28	8	0.1443	24
36	glass	214	9	6	0.0421	36
37	haberman-survival	306	3	2	0.0098	153
38	hayes-roth	160	3	3	0.0188	53
39	heart-cleveland	303	13	5	0.0429	61
40	heart-hungarian	294	12	2	0.0408	147
41	heart-switzerland	123	12	5	0.0976	25
42	heart-va	200	12	5	0.0600	40
43	hepatitis	155	19	2	0.1226	78
44	hill-valley	1212	100	2	0.0825	606
45	horse-colic	368	25	2	0.0679	184
46	ilpd-indian-liver	583	9	2	0.0154	292
47	image-segmentation	2310	18	7	0.0078	330
48	ionosphere	351	33	2	0.0940	176
49	iris	150	4	3	0.0267	50
50	led-display	1000	7	10	0.0070	100
51	lenses	24	4	3	0.1667	8
52	letter	20,000	16	26	0.0008	769
53	libras	360	90	15	0.2500	24
54	low-res-spect	531	100	9	0.1883	59
55	lung-cancer	32	56	3	1.7500	11
56	lymphography	148	18	4	0.1216	37
57	magic	19,020	10	2	0.0005	9510
58	mammographic	961	5	2	0.0052	480
59	miniboone	130,064	50	2	0.0004	65,032
60	molec-biol-promoter	106	57	2	0.5377	53
61	molec-biol-splice	3190	60	3	0.0188	1063
62	monks-1	556	6	2	0.0108	278
63	monks-2	601	6	2	0.0100	300
64	monks-3	554	6	2	0.0108	277
65	mushroom	8124	21	2	0.0026	4062
66	musk-1	476	166	2	0.3487	238
67	musk-2	6598	166	2	0.0252	3299
68	nursery	12,960	8	5	0.0006	2592
69	oocytes_merluccius_nucleus_4d	1022	41	2	0.0401	511
70	oocytes_merluccius_states_2f	1022	25	3	0.0245	341
71	oocytes_trisopterus_nucleus_2f	912	25	2	0.0274	456
72	oocytes_trisopterus_states_5b	912	32	3	0.0351	304
73	optical	5620	62	10	0.0110	562
74	ozone	2536	72	2	0.0284	1268
75	page-blocks	5473	10	5	0.0018	1095
76	parkinsons	195	22	2	0.1128	98
77	pendigits	10,992	16	10	0.0015	1099
78	pima	768	8	2	0.0104	384
79	pittsburg-bridges-MATERIAL	106	7	3	0.0660	35
80	pittsburg-bridges-REL-L	103	7	3	0.0680	34

Table A1. *Cont.*

ID	Dataset	No. Samples	No. Features	No. Classes	Ratio Feat/Samples	Avg Samples per Class
81	pittsburg-bridges-SPAN	92	7	3	0.0761	31
82	pittsburg-bridges-T-OR-D	102	7	2	0.0686	51
83	pittsburg-bridges-TYPE	105	7	6	0.0667	18
84	planning	182	12	2	0.0659	91
85	plant-margin	1600	64	100	0.0400	16
86	plant-shape	1600	64	100	0.0400	16
87	plant-texture	1599	64	100	0.0400	16
88	post-operative	90	8	3	0.0889	30
89	primary-tumor	330	17	15	0.0515	22
90	ringnorm	7400	20	2	0.0027	3700
91	seeds	210	7	3	0.0333	70
92	semeion	1593	256	10	0.1607	159
93	soybean	683	35	18	0.0512	38
94	spambase	4601	57	2	0.0124	2300
95	spect	265	22	2	0.0830	132
96	spectf	267	44	2	0.1648	134
97	statlog-australian-credit	690	14	2	0.0203	345
98	statlog-german-credit	1000	24	2	0.0240	500
99	statlog-heart	270	13	2	0.0481	135
100	statlog-image	2310	18	7	0.0078	330
101	statlog-landsat	6435	36	6	0.0056	1072
102	statlog-shuttle	58,000	9	7	0.0002	8286
103	statlog-vehicle	846	18	4	0.0213	212
104	steel-plates	1941	27	7	0.0139	277
105	synthetic-control	600	60	6	0.1000	100
106	teaching	151	5	3	0.0331	50
107	thyroid	7200	21	3	0.0029	2400
108	tic-tac-toe	958	9	2	0.0094	479
109	titanic	2201	3	2	0.0014	1100
110	trains	10	29	2	2.9000	5
111	twonorm	7400	20	2	0.0027	3700
112	vertebral-column-2clases	310	6	2	0.0194	155
113	vertebral-column-3clases	310	6	3	0.0194	103
114	wall-following	5456	24	4	0.0044	1364
115	waveform	5000	21	3	0.0042	1667
116	waveform-noise	5000	40	3	0.0080	1667
117	wine	178	13	3	0.0730	59
118	wine-quality-red	1599	11	6	0.0069	266
119	wine-quality-white	4898	11	7	0.0022	700
120	yeast	1484	8	10	0.0054	148
121	zoo	101	16	7	0.1584	14

Marked in gray are the 19 "separated" datasets.

Appendix B. Detailed Results per Dataset

Table A2 includes the detailed results for each of the 121 UCI datasets obtained with the Prot-HDCL sim_{Ham}, Prot-HDCL sim_{dot}, AM-HDCL $\alpha = 0$, MM-HDCL $\alpha = 0$, AM-HDCL, and MM-HDCL models for hypervector dimensionality $D = 10{,}000$. In addition, the best α hyperparameter value for the AM-HDCL and MM-HDCL models obtained through validation is shown in the last two columns.

Table A2. Detailed results of the Prot-HDCL sim_{Ham}, Prot-HDCL sim_{dot}, AM-HDCL $\alpha = 0$, MM-HDCL $\alpha = 0$, AM-HDCL, and MM-HDCL models for the 121 UCI datasets. Hypervector dimensionality $D = 10{,}000$. The last two columns present the α hyperparameter value for AM-HDCL and MM-HDCL selected through validation.

ID	Prot-HDCL sim_{Ham}	Prot-HDCL sim_{dot}	AM-HDCL $\alpha = 0$	MM-HDCL $\alpha - 0$	AM-HDCL	MM-HDCL	AMα	MMα
1	54.10 (±0.21)	54.03 (±0.24)	60.46 (±0.82)	61.80 (±0.91)	59.70 (±0.50)	62.61 (±0.56)	0.2	0.1
2	96.42 (±0.97)	96.25 (±0.90)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	0	0
3	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	100.00 (±0.00)	0	0
4	75.12 (±0.38)	75.55 (±1.55)	85.08 (±0.20)	84.54 (±0.46)	85.08 (±0.20)	84.54 (±0.46)	0	0
5	55.50 (±1.51)	47.50 (±3.17)	97.30 (±0.82)	95.80 (±1.40)	97.70 (±0.82)	96.60 (±0.84)	0.02	0.02
6	59.89 (±1.53)	61.59 (±1.54)	71.70 (±0.92)	61.59 (±1.54)	71.70 (±0.92)	61.75 (±1.33)	0	0.02
7	77.60 (±2.80)	48.40 (±7.88)	72.40 (±6.10)	76.00 (±3.77)	72.40 (±5.80)	77.60 (±6.31)	0.02	0.04
8	66.96 (±0.59)	61.84 (±0.98)	92.98 (±0.86)	95.95 (±0.54)	92.98 (±0.86)	95.95 (±0.54)	0	0
9	85.63 (±3.02)	79.38 (±10.64)	80.00 (±2.64)	80.63 (±8.56)	75.00 (±6.59)	81.88 (±6.22)	0.2	0.4
10	64.01 (±0.64)	63.64 (±2.08)	89.17 (±0.30)	89.89 (±0.26)	90.15 (±0.15)	90.02 (±0.19)	0.04	0.02
11	65.88 (±0.44)	65.57 (±0.86)	76.24 (±0.77)	78.64 (±0.37)	79.97 (±0.50)	79.41 (±0.62)	0.2	0.2
12	71.94 (±0.60)	66.02 (±1.38)	68.70 (±1.20)	73.03 (±1.90)	75.85 (±1.08)	74.61 (±0.35)	0.2	0.4
13	96.83 (±0.09)	96.74 (±0.11)	96.97 (±0.22)	97.50 (±0.18)	97.46 (±0.09)	97.71 (±0.00)	0.12	0.15
14	94.61 (±0.36)	94.51 (±0.35)	95.48 (±0.33)	94.42 (±0.74)	96.57 (±0.27)	96.71 (±0.31)	0.02	0.04
15	61.68 (±1.28)	54.29 (±3.51)	74.95 (±2.88)	77.45 (±2.55)	76.63 (±0.89)	82.09 (±0.88)	0.3	0.3
16	65.00 (±0.50)	64.81 (±0.67)	69.13 (±1.89)	70.96 (±1.68)	67.40 (±3.38)	66.92 (±3.21)	0.12	0.1
17	68.96 (±0.50)	76.86 (±1.47)	92.59 (±0.58)	91.45 (±0.99)	92.59 (±0.58)	91.45 (±0.99)	0	0
18	51.69 (±0.29)	49.06 (±1.02)	77.59 (±2.03)	73.01 (±1.57)	77.59 (±2.03)	73.78 (±0.89)	0	0.12
19	77.37 (±0.45)	76.32 (±1.82)	93.05 (±0.25)	93.12 (±0.31)	91.34 (±0.16)	93.12 (±0.31)	0.06	0
20	23.27 (±0.10)	22.36 (±0.43)	33.23 (±0.72)	31.18 (±0.72)	36.71 (±0.41)	33.62 (±0.74)	0.2	0.1
21	83.30 (±0.61)	82.73 (±1.17)	97.42 (±0.17)	95.54 (±0.11)	97.06 (±0.21)	95.54 (±0.11)	0.02	0
22	61.47 (±0.00)	61.47 (±0.00)	59.45 (±0.28)	59.98 (±0.93)	62.22 (±0.27)	61.08 (±0.19)	0.12	0.7
23	67.07 (±1.86)	66.63 (±1.91)	81.15 (±1.36)	77.31 (±4.52)	82.98 (±1.34)	80.72 (±0.86)	0.08	0.15
24	80.28 (±1.50)	73.61 (±2.50)	76.88 (±2.41)	98.07 (±0.39)	84.98 (±2.65)	89.20 (±2.61)	0.06	0.02
25	65.88 (±1.29)	67.14 (±6.70)	81.93 (±0.40)	80.14 (±0.31)	81.93 (±0.40)	80.14 (±0.31)	0	0
26	46.24 (±0.38)	46.13 (±0.48)	50.69 (±0.42)	52.53 (±1.29)	52.81 (±0.21)	51.02 (±0.50)	0.1	0.25
27	87.12 (±0.28)	86.80 (±0.74)	84.16 (±1.07)	86.89 (±0.57)	87.05 (±0.44)	87.50 (±0.65)	0.5	0.3
28	66.88 (±0.78)	66.37 (±1.67)	77.13 (±1.22)	75.96 (±0.87)	77.64 (±1.26)	76.84 (±1.16)	0.08	0.15
29	97.28 (±0.16)	97.20 (±0.17)	97.53 (±0.29)	92.39 (±0.86)	97.42 (±0.27)	97.47 (±0.28)	0.04	0.08
30	81.21 (±0.48)	81.36 (±0.39)	80.91 (±1.55)	78.26 (±1.75)	85.61 (±0.00)	85.68 (±0.24)	0.2	0.4
31	85.33 (±0.54)	51.34 (±2.76)	81.93 (±1.25)	83.93 (±1.25)	85.83 (±0.90)	85.27 (±0.58)	0.15	0.15
32	83.13 (±0.43)	82.19 (±0.42)	93.72 (±0.12)	93.98 (±0.61)	91.80 (±0.51)	93.98 (±0.61)	0.08	0
33	84.58 (±0.29)	84.31 (±0.24)	89.26 (±0.54)	88.93 (±0.35)	85.09 (±0.26)	85.65 (±0.74)	0.2	0.06
34	72.50 (±1.35)	76.00 (±1.76)	84.00 (±1.49)	85.30 (±0.67)	86.90 (±0.57)	86.30 (±0.67)	0.15	0.25
35	47.81 (±1.12)	34.11 (±1.87)	55.36 (±1.25)	57.71 (±1.34)	57.50 (±1.82)	59.06 (±1.67)	0.04	0.04
36	53.44 (±1.34)	54.15 (±1.57)	67.41 (±2.08)	68.44 (±1.21)	67.41 (±2.08)	69.86 (±1.36)	0	0.02
37	66.55 (±0.64)	66.94 (±1.26)	71.41 (±0.81)	70.66 (±1.53)	70.43 (±0.80)	74.01 (±0.00)	0.3	0.7
38	50.00 (±0.00)	50.00 (±0.00)	49.29 (±6.02)	39.29 (±0.00)	59.64 (±13.26)	67.14 (±10.49)	0.02	0.08
39	55.20 (±0.98)	49.05 (±1.28)	54.93 (±1.15)	55.23 (±1.84)	59.97 (±0.64)	59.67 (±0.96)	0.4	0.4
40	85.58 (±0.25)	85.31 (±0.38)	78.49 (±0.58)	78.66 (±1.61)	85.68 (±0.22)	85.24 (±0.25)	0.3	0.4
41	33.39 (±1.02)	7.18 (±0.71)	35.32 (±1.69)	40.48 (±1.51)	39.27 (±2.19)	40.73 (±2.56)	0.06	0.12
42	31.00 (±1.00)	30.15 (±0.88)	31.30 (±1.70)	31.30 (±2.53)	31.35 (±2.84)	31.30 (±2.53)	0.04	0
43	78.59 (±0.45)	76.86 (±1.02)	82.56 (±0.95)	83.21 (±1.27)	83.46 (±0.84)	84.81 (±0.80)	0.02	0.12
44	51.68 (±0.38)	51.67 (±1.07)	51.52 (±0.55)	53.18 (±1.42)	51.82 (±0.17)	53.91 (±1.37)	0.06	0.02
45	76.91 (±1.56)	77.06 (±2.21)	82.94 (±1.73)	83.68 (±0.83)	87.94 (±0.93)	87.50 (±1.59)	0.15	0.25
46	60.12 (±0.58)	59.61 (±1.46)	71.92 (±1.29)	70.96 (±1.06)	71.92 (±0.00)	72.07 (±0.26)	0.4	0.7
47	84.60 (±0.38)	84.11 (±0.86)	90.17 (±0.99)	90.87 (±1.53)	90.20 (±1.11)	91.39 (±0.50)	0.08	0.06
48	69.69 (±0.45)	69.77 (±0.57)	89.94 (±0.93)	93.21 (±0.94)	94.40 (±0.73)	94.09 (±0.42)	0.08	0.25
49	92.77 (±0.72)	92.23 (±0.97)	96.08 (±0.70)	97.70 (±0.47)	96.28 (±0.57)	96.76 (±0.53)	0.06	0.06
50	63.16 (±1.37)	63.60 (±1.33)	71.08 (±0.98)	71.21 (±1.08)	74.20 (±0.25)	73.33 (±0.48)	0.25	0.12
51	84.58 (±3.43)	61.67 (±2.64)	79.17 (±3.40)	72.08 (±3.95)	79.17 (±3.40)	79.17 (±2.78)	0	0.1
52	65.09 (±0.41)	64.84 (±0.60)	69.54 (±0.59)	65.44 (±0.98)	69.54 (±0.59)	65.44 (±0.98)	0	0
53	57.11 (±0.79)	57.58 (±1.00)	73.97 (±1.37)	76.69 (±0.85)	76.14 (±1.68)	76.69 (±0.85)	0.04	0

Table A2. *Cont.*

ID	Prot-HDCL sim_{Ham}	Prot-HDCL sim_{dot}	AM-HDCL $\alpha = 0$	MM-HDCL $\alpha - 0$	AM-HDCL	MM-HDCL	AMα	MMα
54	77.33 (±0.44)	51.73 (±3.24)	90.51 (±0.47)	89.21 (±0.64)	89.61 (±0.58)	89.21 (±0.64)	0.02	0
55	51.56 (±2.21)	41.88 (±4.22)	47.50 (±4.84)	50.31 (±4.76)	44.06 (±2.31)	48.13 (±4.93)	0.06	0.4
56	85.47 (±0.80)	67.70 (±1.85)	86.28 (±0.90)	87.77 (±0.67)	86.82 (±1.40)	89.12 (±1.08)	0.08	0.2
57	75.01 (±0.35)	74.76 (±0.40)	84.89 (±0.16)	82.81 (±0.64)	84.89 (±0.16)	82.59 (±0.59)	0	0.02
58	79.64 (±0.17)	79.75 (±0.38)	80.07 (±0.66)	79.00 (±0.62)	81.07 (±0.27)	79.00 (±0.62)	0.3	0
59	80.73 (±0.65)	80.78 (±1.25)	86.61 (±0.40)	87.85 (±0.25)	87.66 (±0.19)	87.85 (±0.25)	0	0
60	83.75 (±2.00)	85.38 (±1.42)	85.96 (±1.77)	88.85 (±2.18)	89.90 (±1.38)	88.75 (±1.57)	0.15	0.3
61	84.18 (±0.34)	81.88 (±3.76)	91.86 (±0.33)	92.07 (±1.14)	92.78 (±0.36)	92.07 (±1.14)	0.04	0
62	63.84 (±0.37)	60.76 (±0.97)	79.91 (±1.09)	77.87 (±1.10)	81.25 (±1.01)	77.87 (±1.10)	0.02	0
63	80.90 (±0.76)	80.97 (±0.83)	94.31 (±1.14)	90.02 (±1.60)	88.17 (±0.92)	86.46 (±1.19)	0.12	0.2
64	88.68 (±0.71)	77.01 (±1.77)	93.01 (±0.71)	92.94 (±0.78)	92.94 (±0.93)	92.15 (±0.57)	0.08	0.25
65	87.06 (±0.48)	87.53 (±0.91)	99.78 (±0.05)	99.83 (±0.08)	99.99 (±0.01)	99.94 (±0.04)	0.02	0.02
66	65.15 (±0.47)	64.98 (±1.25)	84.33 (±0.51)	74.12 (±3.00)	83.34 (±1.52)	74.12 (±3.00)	0.02	0
67	75.56 (±0.12)	75.66 (±0.21)	97.11 (±0.31)	96.09 (±0.30)	97.11 (±0.31)	96.09 (±0.30)	0	0
68	83.17 (±0.19)	66.98 (±1.83)	93.63 (±0.45)	88.72 (±0.56)	94.68 (±0.17)	91.49 (±0.26)	0.04	0.12
69	56.91 (±0.20)	57.16 (±0.31)	73.63 (±1.62)	74.31 (±0.75)	73.63 (±1.62)	72.03 (±0.73)	0	0.08
70	83.70 (±0.24)	83.76 (±0.45)	90.84 (±0.46)	91.50 (±0.47)	91.93 (±0.35)	92.06 (±0.28)	0.04	0.02
71	59.45 (±0.85)	59.40 (±1.43)	60.96 (±1.57)	68.00 (±2.27)	74.29 (±0.83)	70.68 (±2.26)	0.08	0.2
72	72.45 (±0.52)	72.70 (±0.79)	90.86 (±0.51)	90.83 (±0.39)	88.84 (±0.49)	89.67 (±0.69)	0.08	0.06
73	89.00 (±0.22)	89.03 (±0.36)	94.00 (±0.19)	94.75 (±0.34)	95.89 (±0.21)	95.39 (±0.28)	0.04	0.04
74	69.87 (±0.52)	71.14 (±1.61)	96.65 (±0.11)	96.88 (±0.15)	97.15 (±0.02)	97.16 (±0.00)	0.1	0.04
75	73.75 (±0.93)	73.89 (±1.52)	95.27 (±0.25)	96.22 (±0.15)	95.48 (±0.17)	96.22 (±0.15)	0.02	0
76	76.58 (±0.78)	76.53 (±0.96)	89.34 (±0.70)	90.20 (±1.20)	89.34 (±0.70)	90.51 (±0.55)	0	0.02
77	78.79 (±0.19)	78.68 (±0.45)	95.34 (±0.12)	95.05 (±0.16)	95.99 (±0.18)	95.35 (±0.41)	0.04	0.04
78	67.93 (±0.39)	67.92 (±0.46)	73.66 (±1.30)	73.95 (±1.01)	74.70 (±0.30)	74.24 (±0.37)	0.12	0.15
79	84.90 (±0.46)	83.17 (±0.82)	93.17 (±1.24)	91.44 (±1.39)	92.50 (±1.68)	93.27 (±1.01)	0.08	0.06
80	62.50 (±1.01)	54.23 (±2.91)	71.83 (±2.13)	74.90 (±1.95)	73.08 (±1.01)	74.13 (±2.62)	0.2	0.25
81	59.89 (±1.08)	59.46 (±1.45)	59.67 (±3.01)	54.67 (±1.54)	68.59 (±0.95)	65.11 (±1.66)	0.25	0.25
82	89.10 (±1.29)	91.20 (±0.63)	88.70 (±1.49)	87.70 (±1.34)	88.80 (±0.79)	88.70 (±0.48)	0.12	0.4
83	54.62 (±1.09)	50.10 (±0.84)	61.92 (±2.65)	62.88 (±1.58)	71.25 (±0.84)	71.83 (±1.57)	0.4	0.5
84	49.44 (±2.22)	53.67 (±2.67)	55.72 (±3.34)	56.72 (±3.73)	71.11 (±0.00)	71.11 (±0.00)	0.5	0.4
85	79.34 (±0.42)	78.89 (±0.61)	79.08 (±0.63)	69.21 (±1.16)	79.08 (±0.63)	73.83 (±1.30)	0	0.04
86	51.68 (±0.33)	51.88 (±0.67)	42.11 (±1.00)	53.56 (±1.63)	47.51 (±0.78)	53.56 (±1.63)	0.02	0
87	70.98 (±0.49)	71.13 (±0.64)	74.77 (±0.43)	69.53 (±0.87)	74.77 (±0.43)	73.19 (±1.28)	0	0.02
88	51.02 (±2.48)	0.68 (±0.59)	47.27 (±2.35)	47.95 (±2.61)	70.34 (±0.99)	70.23 (±0.72)	0.4	0.7
89	41.65 (±0.54)	35.30 (±0.69)	45.03 (±1.23)	49.09 (±1.55)	52.23 (±0.83)	52.65 (±1.23)	0.25	0.25
90	83.22 (±0.47)	62.87 (±4.17)	98.07 (±0.08)	98.30 (±0.09)	98.09 (±0.07)	98.11 (±0.07)	0.08	0.4
91	89.62 (±0.52)	89.90 (±0.60)	92.36 (±0.58)	92.12 (±0.61)	92.88 (±0.63)	93.32 (±0.48)	0.06	0.04
92	81.60 (±0.39)	81.06 (±0.69)	90.20 (±0.23)	90.68 (±0.43)	92.31 (±0.25)	92.99 (±0.25)	0.08	0.04
93	81.81 (±0.60)	78.86 (±0.49)	86.97 (±0.80)	85.00 (±1.30)	86.57 (±0.65)	86.94 (±1.55)	0.2	0.12
94	82.29 (±0.33)	82.25 (±0.74)	92.36 (±0.44)	92.13 (±0.36)	92.36 (±0.44)	92.13 (±0.36)	0	0
95	69.68 (±0.73)	63.33 (±3.00)	64.25 (±2.02)	68.71 (±1.29)	59.73 (±0.53)	63.23 (±1.57)	0.7	0.4
96	44.17 (±1.75)	52.19 (±3.66)	51.55 (±3.66)	58.66 (±6.89)	51.55 (±3.66)	59.25 (±3.45)	0	0.15
97	54.68 (±0.90)	55.00 (±2.77)	66.41 (±1.08)	65.87 (±1.23)	66.85 (±0.41)	68.39 (±1.15)	0.3	0.3
98	70.14 (±0.68)	69.60 (±1.57)	75.12 (±0.97)	77.34 (±0.58)	76.76 (±0.97)	77.34 (±0.58)	0.04	0
99	84.81 (±0.50)	84.59 (±0.43)	82.61 (±0.95)	86.90 (±1.00)	87.84 (±0.40)	87.31 (±0.63)	0.15	0.3
100	85.43 (±0.41)	85.57 (±0.61)	96.70 (±0.23)	96.55 (±0.28)	95.44 (±0.29)	96.55 (±0.28)	0.02	0
101	77.75 (±0.35)	77.81 (±0.64)	83.98 (±0.88)	87.49 (±1.16)	83.98 (±0.88)	87.49 (±1.16)	0	0
102	76.49 (±0.94)	75.98 (±2.99)	99.66 (±0.10)	99.59 (±0.13)	99.66 (±0.10)	99.59 (±0.13)	0	0
103	50.63 (±0.95)	50.79 (±0.76)	71.37 (±1.34)	68.87 (±1.14)	71.37 (±1.34)	67.63 (±1.01)	0	0.2
104	53.61 (±0.41)	53.02 (±0.85)	69.97 (±1.15)	68.28 (±0.62)	71.02 (±0.75)	68.28 (±0.62)	0.06	0
105	91.60 (±0.83)	91.55 (±0.47)	96.55 (±0.25)	97.32 (±0.58)	98.08 (±0.50)	98.57 (±0.33)	0.04	0.02
106	49.87 (±0.42)	51.18 (±0.87)	54.34 (±2.50)	53.42 (±2.52)	54.74 (±1.34)	56.71 (±1.61)	0.06	0.25
107	66.75 (±2.50)	64.06 (±7.99)	93.92 (±0.65)	96.45 (±0.18)	93.91 (±0.28)	96.45 (±0.18)	0.04	0
108	69.45 (±0.73)	69.59 (±0.60)	97.56 (±0.28)	97.54 (±0.29)	97.91 (±0.21)	97.81 (±0.33)	0.02	0.02
109	75.50 (±0.00)	74.48 (±0.70)	77.84 (±0.04)	78.83 (±0.17)	77.95 (±0.27)	78.83 (±0.17)	0.3	0
110	87.50 (±0.00)	87.50 (±0.00)	87.50 (±0.00)	62.50 (±0.00)	87.50 (±0.00)	62.50 (±0.00)	0	0

Table A2. *Cont.*

ID	Prot-HDCL sim_{Ham}	Prot-HDCL sim_{dot}	AM-HDCL $\alpha = 0$	MM-HDCL $\alpha - 0$	AM-HDCL	MM-HDCL	AMα	MMα
111	96.72 (±0.14)	95.73 (±1.16)	96.76 (±0.12)	97.50 (±0.11)	97.47 (±0.06)	97.50 (±0.11)	0.04	0
112	71.14 (±0.56)	72.24 (±0.80)	82.56 (±0.98)	83.67 (±0.97)	82.56 (±0.98)	83.67 (±0.97)	0	0
113	76.56 (±0.76)	76.95 (±0.53)	81.95 (±1.06)	81.46 (±1.03)	81.95 (±1.06)	82.56 (±0.87)	0	0.02
114	61.89 (±0.31)	61.87 (±1.05)	93.96 (±0.52)	78.58 (±1.77)	93.96 (±0.52)	82.31 (±0.49)	0	0.15
115	79.99 (±0.29)	79.61 (±0.71)	84.03 (±0.44)	84.96 (±0.57)	86.02 (±0.30)	85.75 (±0.29)	0.06	0.1
116	79.43 (±0.31)	81.64 (±1.01)	85.43 (±0.28)	85.43 (±0.67)	85.89 (±0.30)	85.43 (±0.67)	0.06	0
117	97.67 (±0.18)	96.82 (±0.72)	96.88 (±0.40)	98.24 (±0.63)	98.30 (±0.60)	98.75 (±0.59)	0.04	0.02
118	42.56 (±0.82)	29.13 (±1.41)	59.11 (±0.63)	60.30 (±0.52)	60.51 (±0.51)	60.31 (±0.37)	0.12	0.1
119	34.27 (±0.41)	30.92 (±0.88)	53.56 (±0.66)	53.63 (±0.42)	55.50 (±0.21)	55.13 (±0.41)	0.06	0.06
120	50.06 (±0.53)	49.82 (±1.06)	54.89 (±1.21)	57.93 (±0.83)	58.14 (±0.81)	57.99 (±1.01)	0.25	0.12
121	96.40 (±0.52)	93.10 (±1.29)	99.00 (±0.00)	98.60 (±0.70)	98.80 (±0.42)	98.90 (±0.32)	0.3	0.3
mean	70.92 (±16.05)	67.43 (±19.35)	79.00 (±16.55)	78.89 (±16.32)	80.49 (±15.40)	80.27 (±15.25)		

References

1. Xu, W.; Gupta, S.; Morris, J.; Shen, X.; Imani, M.; Aksanli, B.; Rosing, T. Tri-HD: Energy-Efficient On-Chip Learning with In-Memory Hyperdimensional Computing. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2024**, *44*, 525–539. [CrossRef]
2. Kang, J.; Xu, W.; Bittremieux, W.; Moshiri, N.; Rosing, T.Š. DRAM-Based Acceleration of Open Modification Search in Hyperdimensional Space. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2024**, *43*, 2592–2605. [CrossRef]
3. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2009.
4. Reed, S.K. Pattern recognition and categorization. *Cogn. Psychol.* **1972**, *3*, 382–407. [CrossRef]
5. Kanerva, P. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cogn. Comput.* **2009**, *1*, 139–159. [CrossRef]
6. Rahimi, A.; Kanerva, P.; Benini, L.; Rabaey, J.M. Efficient biosignal processing using hyperdimensional computing: network templates for combined learning and classification of ExG signals. *Proc. IEEE* **2019**, *107*, 123–143. [CrossRef]
7. Vdovychenko, R.; Tulchinsky, V. Increasing the Semantic Storage Density of Sparse Distributed Memory. *Cybern. Syst. Anal.* **2022**, *58*, 331–342. [CrossRef]
8. Vdovychenko, R.; Tulchinsky, V. *Sparse Distributed Memory for Sparse Distributed Data*; Springer: Cham, Switzerland, 2023; Volume 542, pp. 74–81. [CrossRef]
9. Widdows, D.; Cohen, T. Reasoning with vectors: A continuous model for fast robust inference. *Log. J. IGPL* **2015**, *23*, 141–173. [CrossRef]
10. Imani, M.; Bosch, S.; Javaheripi, M.; Rouhani, B.; Wu, X.; Koushanfar, F.; Rosing, T. SemiHD: Semi-supervised learning using hyperdimensional computing. In Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Westminster, CO, USA, 4–7 November 2019; pp. 1–8. [CrossRef]
11. Imani, M.; Salamat, S.; Khaleghi, B.; Samragh, M.; Koushanfar, F.; Rosing, T. SparseHD: Algorithm-hardware co-optimization for efficient high-dimensional computing. In Proceedings of the 27th IEEE International Symposium on Field-Programmable Custom Computing Machines, FCCM 2019, San Diego, CA, USA, 28 April–1 May 2019; IEEE: New York, NY, USA, 2019; pp. 190–198. [CrossRef]
12. Morris, J.; Imani, M.; Bosch, S.; Thomas, A.; Shu, H.; Rosing, T. CompHD: Efficient hyperdimensional computing using model compression. In Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design, Lausanne, Switzerland, 29–31 July 2019; pp. 1–6. [CrossRef]
13. Zou, Z.; Kim, Y.; Imani, F.; Alimohamadi, H.; Cammarota, R.; Imani, M. Scalable edge-based hyperdimensional learning system with brain-like neural adaptation. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, St. Louis, MO, USA, 14–19 November 2021; IEEE Computer Society: New York, NY, USA, 2021; pp. 1–15. [CrossRef]
14. Duan, S.; Liu, Y.; Ren, S.; Xu, X. LeHDC: Learning-based hyperdimensional computing classifier. In Proceedings of the Design Automation Conference, San Francisco, CA, USA, 10–14 July 2022; pp. 1–7. [CrossRef]
15. Pale, U.; Teijeiro, T.; Atienza, D. Multi-centroid hyperdimensional computing approach for epileptic seizure detection. *Front. Neurol.* **2022**, *13*, 816294. [CrossRef]
16. Wang, J.; Huang, S.; Imani, M. DistHD: A learner-aware dynamic encoding method for hyperdimensional classification. In Proceedings of the Proceedings—Design Automation Conference, San Francisco, CA, USA, 9–13 July 2023; IEEE: New York, NY, USA, 2023; Volume 2023, pp. 1–6. [CrossRef]

17. Joshi, A.; Halseth, J.T.; Kanerva, P. Language Geometry Using Random Indexing. In *Quantum Interaction*; de Barros, J.A., Coecke, B., Pothos, E., Eds.; Springer: Cham, Switzerland, 2017; pp. 265–274.
18. Schlegel, K.; Mirus, F.; Neubert, P.; Protzel, P. Multivariate time series analysis for driving style classification using neural networks and hyperdimensional computing. In Proceedings of the IEEE Intelligent Vehicles Symposium, Nagoya, Japan, 11–17 July 2021; IEEE: New York, NY, USA, 2021; Volume 2021, pp. 602–609. [CrossRef]
19. Schlegel, K.; Neubert, P.; Protzel, P. HDC-MiniROCKET: Explicit time encoding in time series classification with hyperdimensional computing. In Proceedings of the International Joint Conference on Neural Network, Padua, Italy, 18–23 July 2022; pp. 1–8. [CrossRef]
20. Schlegel, K.; Kleyko, D.; Brinkmann, B.H.; Nurse, E.S.; Gayler, R.W.; Neubert, P. Lessons from a challenge on forecasting epileptic seizures from non-cerebral signals. *Nat. Mach. Intell.* **2024**, *6*, 243–244. [CrossRef]
21. Watkinson, N.; Givargis, T.; Joe, V.; Nicolau, A.; Veidenbaum, A. Detecting COVID-19 related pneumonia on CT scans using hyperdimensional computing. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, Virtual, 1–5 November 2021; IEEE: New York, NY, USA, 2021; pp. 3970–3973. [CrossRef]
22. Cumbo, F.; Cappelli, E.; Weitschek, E. A brain-inspired hyperdimensional computing approach for classifying massive DNA methylation data of cancer. *Algorithms* **2020**, *13*, 233. [CrossRef]
23. Rahimi, A.; Kanerva, P.; Rabaey, J.M. A robust and energy-efficient classifier using brain-inspired hyperdimensional computing. In Proceedings of the International Symposium on Low Power Electronics and Design, San Francisco Airport, CA, USA, 8–10 August 2016; IEEE: New York, NY, USA, 2016; pp. 64–69. [CrossRef]
24. Neubert, P.; Schubert, S.; Protzel, P. An introduction to hyperdimensional computing for robotics. *KI-Kunstl. Intell.* **2019**, *33*, 319–330. [CrossRef]
25. Ge, L.; Parhi, K.K. Classification using hyperdimensional computing: A review. *IEEE Circuits Syst. Mag.* **2020**, *20*, 30–47. [CrossRef]
26. Aygun, S.; Moghadam, M.S.; Najafi, M.H.; Imani, M. Learning from hypervectors: A survey on hypervector encoding. *arXiv* **2023**, arXiv:2308.00685. [CrossRef]
27. Vergés, P.; Heddes, M.; Nunes, I.; Givargis, T.; Nicolau, A. *Classification Using Hyperdimensional Computing: A Review with Comparative Analysis*; Research Square: Durham, NC, USA, 2023; pp. 1–49. [CrossRef]
28. Plate, T.A. Holographic reduced representations. *IEEE Trans. Neural Netw.* **1995**, *6*, 623–641. [CrossRef]
29. Plate, T.A. *Holographic Reduced Representation: Distributed Representation for Cognitive Structures*; CSLI Publications: Stanford, CA, USA, 2003.
30. Gayler, R.W. Multiplicative binding, representation operators & analogy. In Proceedings of the Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences, Sofia, Bulgaria, July 1998; pp. 1–4.
31. Kanerva, P. The spatter code for encoding concepts at many levels. In Proceedings of the International Conference on Artificial Neural Networks (ICANN), Sorrento, Italy, 26–29 May 1994; pp. 226–229. [CrossRef]
32. Kanerva, P. Binary spatter-coding of ordered K-tuples. In Proceedings of the Artificial Neural Networks—ICANN, Bochum, Germany, 16–19 July 1996; pp. 869–873. [CrossRef]
33. Laiho, M.; Poikonen, J.H.; Kanerva, P.; Lehtonen, E. High-dimensional computing with sparse vectors. In Proceedings of the IEEE Biomedical Circuits and Systems Conference: Engineering for Healthy Minds and Able Bodies, BioCAS 2015, Atlanta, GA, USA, 22–24 October 2015; IEEE: New York, NY, USA, 2015; pp. 1–4. [CrossRef]
34. Imani, M.; Huang, C.; Kong, D.; Rosing, T. Hierarchical hyperdimensional computing for energy efficient classification. In Proceedings of the ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 24–29 June 2018; IEEE: New York, NY, USA, 2018; Volume Part F137710, pp. 1–6. [CrossRef]
35. Rachkovskij, D.A.; Fedoseeva, T.V. On audio signals recognition by multilevel neural network. In Proceedings of the International Symposium on Neural Networks and Neural Computing—NEURONET'90, Prague, Czech Republic, 10–14 September 1990; pp. 281–283.
36. Kussul, E.; Rachkovskij, D.A. On image texture recognition by associative-projective neurocomputer. In Proceedings of the ANNIE'91 Conference "Intelligent Engineering Systems Through Artificial Neural Networks", St. Louis, MO, USA, 10–13 November 1991; pp. 453–458.
37. Rachkovskij, D.A.; Slipchenko, S.V.; Misuno, I.S.; Kussul, E.M.; Baidyk, T.N. Sparse binary distributed encoding of numeric vectors. *J. Autom. Inf. Sci.* **2005**, *37*, 47–61. [CrossRef]
38. Rachkovskij, D.A.; Slipchenko, S.V.; Kussul, E.M.; Baidyk, T.N. Sparse binary distributed encoding of scalars. *J. Autom. Inf. Sci.* **2005**, *37*, 12–23. [CrossRef]
39. Imani, M.; Kong, D.; Rahimi, A.; Rosing, T. VoiceHD: Hyperdimensional computing for efficient speech recognition. In Proceedings of the IEEE International Conference on Rebooting Computing (ICRC), Washington, DC, USA, 8–9 November 2017; pp. 1–8. [CrossRef]

40. Kim, Y.; Imani, M.; Rosing, T.S. Efficient human activity recognition using hyperdimensional computing. In Proceedings of the 8th International Conference on the Internet of Things, Santa Barbara, CA, USA, 15–18 October 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 1–6. [CrossRef]
41. Imani, M.; Messerly, J.; Wu, F.; Pi, W.; Rosing, T. A binary learning framework for hyperdimensional computing. In Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 126–131. [CrossRef]
42. Imani, M.; Bosch, S.; Datta, S.; Ramakrishna, S.; Salamat, S.; Rabaey, J.M.; Rosing, T. QuantHD: A quantization framework for hyperdimensional computing. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2019**, *39*, 2268–2278. [CrossRef]
43. Hsiao, Y.R.; Chuang, Y.C.; Chang, C.Y.; Wu, A.Y. Hyperdimensional computing with learnable projection for user adaptation framework. In Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations (AIAI), Crete, Greece, 25–27 June 2021; pp. 436–447. [CrossRef]
44. Imani, M.; Morris, J.; Bosch, S.; Shu, H.; Micheli, G.D.; Rosing, T. AdaptHD: Adaptive efficient training for brain-inspired hyperdimensional computing. In Proceedings of the IEEE Biomedical Circuits and Systems Conference (BioCAS), Nara, Japan, 17–19 October 2019; pp. 1–4. [CrossRef]
45. Hernández-Cano, A.; Matsumoto, N.; Ping, E.; Imani, M. OnlineHD: Robust, efficient, and single-pass online learning using hyperdimensional system. In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE), Grenoble, France, 1–5 February 2021; pp. 1–6. [CrossRef]
46. Chuang, Y.C.; Chang, C.Y.; Wu, A.Y.A. Dynamic hyperdimensional computing for improving accuracy-energy efficiency trade-offs. In Proceedings of the IEEE Workshop on Signal Processing Systems, SiPS: Design and Implementation, Coimbra, Portugal, 20–22 October 2020; IEEE: New York, NY, USA, 2020; pp. 1–5. [CrossRef]
47. Vergés, P.; Givargis, T.; Nicolau, A. RefineHD: Accurate and efficient single-pass adaptive learning using hyperdimensional computing. In Proceedings of the IEEE International Conference on Rebooting Computing (ICRC), San Diego, CA, USA, 5–6 December 2023; pp. 1–8. [CrossRef]
48. Smets, L.; Leekwijck, W.V.; Tsang, I.J.; Latre, S. Training a hyperdimensional computing classifier using a threshold on its confidence. *Neural Comput.* **2023**, *35*, 2006–2023. [CrossRef]
49. Kussul, E.; Baidyk, T.; Kasatkina, L.; Lukovich, V. Rosenblatt perceptrons for handwritten digit recognition. In Proceedings of the International Joint Conference on Neural Networks, Washington, DC, USA, 15–19 July 2001; pp. 1516–1520. [CrossRef]
50. Kussul, E.; Baidyk, T. Improved method of handwritten digit recognition tested on MNIST database. *Image Vis. Comput.* **2004**, *22*, 971–981. [CrossRef]
51. Rachkovskij, D.A. Linear classifiers based on binary distributed representations. *Inf. Theor. Appl.* **2007**, *14*, 270–274.
52. Bartlett, P.; Freund, Y.; Lee, W.S.; Schapire, R.E. Boosting the margin: A new explanation for the effectiveness of voting methods. *Ann. Stat.* **1998**, *26*, 1651–1686. [CrossRef]
53. Mohri, M.; Rostamizadeh, A.; Talwalkar, A. *Foundations of Machine Learning*; The MIT Press: Cambridge, MA, USA, 2012.
54. Fernández-Delgado, M.; Cernadas, E.; Barro, S.; Amorim, D.; Fernández-Delgado, A. Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.* **2014**, *15*, 3133–3181.
55. Wainberg, M.; Alipanahi, B.; Frey, B.J. Are Random Forests Truly the Best Classifiers? *J. Mach. Learn. Res.* **2016**, *17*, 1–5.
56. Kleyko, D.; Kheffache, M.; Frady, E.P.; Wiklund, U.; Osipov, E. Density encoding enables resource-efficient randomly connected neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 3777–3783. [CrossRef]
57. Frady, E.P.; Kleyko, D.; Sommer, F.T. Variable binding for sparse distributed representations: Theory and applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 2191–2204. [CrossRef]