Classification Performance of Confidence-Driven Centroids

Ефективність класифікації заснованих на впевненості центроїдів

Laura Smets, Dmitri Rachkovskij, Evgeny Osipov, Olexander Volkov,
Werner Van Leekwijck, Steven Latré

Abstract

UDC 004.855.5

Hyperdimensional computing (HDC) is a powerful algorithmic framework at the intersection of symbolic and neural network Artificial Intelligence. In particular, HDC has received significant attention as a suitable candidate for low-resource machine learning tasks, exemplified by wearable Internet of Things. To solve classification tasks, HDC transforms input data to a high-dimensional space and uses simple component-wise vector operations to create, train, and operate the classification model. While the classical centroid model has been often used in HDC, iterative updating of centroids with wrongly classified samples improves the classification performance. In this paper, using a large and variable collection of 121 UCI datasets, we explore how confidence-driven training of centroids formed from HDC representations further improves the classification accuracy.

Гіпервимірні обчислення (HDC) є потужною алгоритмічною платформою, що поєднує підходи символьного й нейромережевого штучного інтелекту. Зокрема, HDC привертає значну увагу як перспективний кандидат для задач машинного навчання з низькими ресурсами, наприклад, у носимих пристроях Інтернету речей. Для розвязання класифікаційних задач HDC перетворює вхідні дані у високо вимірний простір і використовує прості покомпонентні векторні операції для створення, тренування та застосування класифікаційної моделі. Хоча класична модель центроїда часто застосовується в HDC, ітеративне оновлення центроїдів помилково класифікованими зразками покращує точність класифікації. У цій роботі ми досліджуємо, як тренування центроїдів, засноване на рівні довіри до класифікації, додатково підвищує точність класифікації, використовуючи велику та різноманітну колекцію 121 наборів даних UCI.

Keywords: Centroid, Linear Classifier, Non-linear Data Transformation, Hyperdimensional Computing, Vector Symbolic Architecture

Ключові слова: Центроїд, лінійний класифікатор, нелінійне перетворення даних, гіпервимірні обчислення, векторне символьна архітектура

1 Introduction

Hyperdimensional Computing (HDC) [1], also referred to as Vector Symbolic Architectures (VSA) [2], is a powerful algorithmic framework for the realization of Artificial Intelligence functionality through algebraic operations on high-dimensional vectors known as hypervectors. In machine learning applications, HDC transforms input data into hypervector space, where information is distributed across thousands of vector components. This enables efficient processing through straightforward component-wise operations. HDC offers several key advantages, including energy efficiency, robustness to noise and hardware faults, as well as rapid processing with low latency [1], [3]–[6]. These characteristics make HDC an attractive option for resource-constrained applications, such as wearable Internet of Things.

HDC has been used in a variety of classification tasks, such as text classification [7]–[9], speech recognition [10], human activity recognition [11], hand gesture recognition [8], [12], [13], time series classification [14], [15], classification of medical images [16], [17], technical diagnostics [18], [19], character recognition [20]–[23], robotics [24], and others, see [25]–[28] for comprehensive surveys.

In these tasks, the classical centroid model is often employed as a classification approach in hypervector space. This model constructs class centroids by averaging the hypervectors that represent the samples of each class. Although this method is straightforward and allows for highly efficient training through averaging, its classification accuracy can be quite limited. For example, even in scenarios where the hypervectors of class samples are linearly separable, simple averaging may lead to a model that fails to attain this separability, thereby impacting its overall performance.

Several proposals have emerged in the literature to enhance the naive centroid model. These approaches include iterative training methods that refine class centroids based on the hypervectors of misclassified samples [29]–[38]. By focusing on the instances that are misclassified, these methods aim to improve the robustness and accuracy of the classification process, enabling better separation of classes with refined centroids in hypervector space.

In this paper, we explore two models that enhance previous HDC classifiers by training not only on misclassified samples but also on correctly classified samples that exhibit low classification confidence. To evaluate our approach, we utilize a standard collection of 121 UCI datasets [39]. We elaborate a comprehensive scheme for proper data preprocessing, classification result averaging, and hyperparameter selection, within the context of randomized data transformations in HDC.

Our models are suitable for large and diverse dataset collections, as exemplified by the 121 UCI datasets. They produce binary centroids to facilitate efficient deployment of the trained models in the evaluation setup. We also investigate the impact of initial data quantization and hypervector dimensionality on classification performance. The results indicate that our Confidence-Driven Centroid models

outperform their classical counterparts and other centroid-based HDC classifiers in terms of mean classification accuracy across the 121 UCI datasets.

2 Transformation of numerical vectors to hypervectors

2.1 Types of hypervectors

In HDC, input data are transformed into a high-dimensional vectors called hypervectors (HV). There is a number of HDC frameworks which use different types of HVs and define their own operations over them. Hypervectors could have real-valued components as in Holographic Reduced Representations (HRR) [40], complex-valued components as in Fourier HRR [41], bipolar components as in Multiply-Add-Permute (MAP) [42], binary components as in Binary Spatter Code (BSC) [43], [44] or in Sparse Binary Distributed Representations (SBDR) [45], see also in [25]–[28]. In this paper, we focus on dense binary hypervectors, where each component is either 0 or 1 with equal probability. Binary vectors enable efficient implementation of operations over them. The component-wise addition $+$ is used to combine hypervectors, followed by a majority vote $[.]$ to binarize the resulting components, with ties resolved pseudo-randomly. Component-wise XOR $\oplus$ is used to associate two or more vectors, while Hamming distance serves as the (un)similarity measure.

2.2 Representation of symbolic data

Symbolic data refer to data where each possible value or category is independent of others. This type of data is transformed into hypervectors by assigning a unique random hypervector (HV) to each symbol, with each component of the HV randomly assigned to 0 or 1. Once generated, the HV for each symbol remains fixed.

2.3 Representation of numerical scalars

To represent numerical scalars as hypervectors, it's beneficial for the HVs to reflect "linear" similarity, such that closer numerical values are represented by more similar HVs. This effect is achieved through "linear" mapping [8], [46], [47]. After quantizing numerical values into integer levels $[0, Q]$, the lowest quantization level is assigned a randomly generated HV.

Each subsequent level's HV is generated by flipping randomly chosen b bits in the HV of the previous level, ensuring no previously flipped bits are flipped again [8], [47]. Consequently, the HV of the highest quantization level differs from that of the lowest by Qb bits. The Hamming distance to the lowest quantization level increases linearly with each level.

2.4 Representation of numerical vectors

Consider transforming a numerical vector $\mathbf{z} \in \mathbb{R}^d$ to the hypervector $\mathbf{h}(\mathbf{z}) \in \{0,1\}^D$. Each of d components of $\mathbf{z}$, i.e., z_j, is a numerical scalar representing the value of the j-th feature. Employing the compositional approach proposed in [48]–[50], we get:

$$\mathbf{h}(\mathbf{z}) = [\sum_{j=1}^{d} \mathbf{v}(z_j) \oplus \mathbf{f}(j)]. \tag{1}$$

Here, each $\mathbf{f}(j)$ is the hypervector of the j-th feature / component considered as symbol in Section 2.2. Each $\mathbf{v}(z_j)$ is the hypervector of a numerical scalar z_j quantized and represented as in Section 2.3.

In some papers, exemplified by [51], each subsequent level's HV is generated by flipping $D/(2Q)$ bits in the HV of the previous quantization level, while ensuring no previously flipped bits are flipped again. Consequently, the HV of the highest quantization level Q differs from that of the lowest level 0 by $QD/(2Q) = D/2$ bits.

The remaining $D/2$ bits will be identical for the HVs of all quantization levels. Denoting this part of the HV $\mathbf{v}$ encoding values by $\mathbf{u}$, we get $\mathbf{u}(0) = \mathbf{u}(1) = ... = \mathbf{u}(Q) = \mathbf{u}$. Thus, we can re-write expression (1) for the corresponding $D/2$ bits of $\mathbf{h}(\mathbf{z})$ as $\mathbf{h}_{D/2}(\mathbf{z})$:

$$\mathbf{h}_{D/2}(\mathbf{z}) = [\mathbf{u} \oplus \mathbf{g}(1) + \mathbf{u} \oplus \mathbf{g}(2) + ... + \mathbf{u} \oplus \mathbf{g}(d)] = [\sum_{j=1}^{d} (\mathbf{u} \oplus \mathbf{g}(j)], \tag{2}$$

where $\mathbf{g}(j)$ is the corresponding remaining $D/2$ bits of $\mathbf{f}(j)$. Since the hypervectors $\mathbf{f}(j)$ representing features are the same for all N data samples, $\mathbf{g}(j)$ are also the same, and $\mathbf{h}_{D/2}(\mathbf{z})$ will be identical for all N data samples. As such, they do not influence classification and can be trimmed.

So, in this work we flip D/Q bits per quantization level instead of $D/(2Q)$ bits and thus exploit all D components of hypervectors. This provides the same classification results with D equal to half of the dimensionality considered in [51]. This encoding scheme has also been used in other works, e.g., [52]–[56].

3 Classifiers

In this section, we describe the classifiers used in this study. Each classifier is trained on a training set containing N samples, denoted as $\{\mathbf{x}_i, y_i\}, i = 1...N$, where $\mathbf{x}_i$ is the d-dimensional feature vector of the i-th sample, and y_i is its class label from the set of the dataset classes with cardinality C. Within the HDC framework, these initial feature vectors $\mathbf{x}_i$ are transformed into hypervectors $\mathbf{h}(\mathbf{x}_i)$ using some transformation method, as exemplified in the previous Section.

3.1 Centroid model

The classic centroid model creates C centroids of the dataset classes. In HDC, this is achieved by combining all sample HVs $\mathbf{h}(\mathbf{x}_i)$ that belong to the same class c to obtain the centroid for that class:

$$\mathbf{C}_c = \sum_{i=1}^{N} \{\mathbf{h}(\mathbf{x}_i) \,|\, y_i = c\}, \qquad c = 1...C. \tag{3}$$

In the classic centroid model, $\mathbf{C}_c$ may be divided by number n_c of samples in the class c to get the mean vector of the class, with n_c defined as

$$n_c = \sum_{i=1}^{N} \{1 \,|\, y_i = c\}, \qquad c = 1...C. \tag{4}$$

In HDC, the class centroids are often binarized component-wise by the majority vote (see also Sections 2.1 and 2.4):

$$\mathbf{c}_c = [\mathbf{C}_c], \qquad c = 1...C. \tag{5}$$

This can be implemented by component-wise thresholding of $\mathbf{C}_c$ with the threshold value $\frac{n_c}{2}$.

At the class prediction stage, for the HDC model, class label prediction for the sample $\mathbf{z}$ is produced as:

$$\hat{y}(\mathbf{z}) = \arg\min_c \operatorname{dist}(\mathbf{h}(\mathbf{z}), \mathbf{c}_c), \qquad c = 1...C, \tag{6}$$

where dist(.,.) is the specific distance measure employed, or as

$$\hat{y}(\mathbf{z}) = \arg\max_c \operatorname{sim}(\mathbf{h}(\mathbf{z}), \mathbf{c}_c), \qquad c = 1...C, \tag{7}$$

where sim(.,.) is the specific similarity measure employed.

For the classic centroid model, $\mathbf{C}_c$ is commonly used in (6) or (7) instead of $\mathbf{c}_c$.

3.2 Refined centroid model

The class centroids as obtained above can be further improved by performing an iterative training procedure as follows. For each training sample $\mathbf{x}_i$, its hypervector $\mathbf{h}(\mathbf{x}_i)$ is used in (6) or (7) instead of $\mathbf{h}(\mathbf{z})$. If the prediction is wrong, i.e., $\hat{y}_i \neq y_i$, the sample's HV is used to update two class centroids as

$$\mathbf{C}_{y_i} = \mathbf{C}_{y_i} + \mathbf{h}(\mathbf{x}_i), \qquad \mathbf{C}_{\hat{y}_i} = \mathbf{C}_{\hat{y}_i} - \mathbf{h}(\mathbf{x}_i). \tag{8}$$

In case of a correct prediction, the class centroids remain unchanged. This iterative training procedure is performed for a predefined number of epochs (with epoch being passing through all training samples)

or until a predefined accuracy on the train set is reached.

If binarized centroids are required, binarization of $\mathbf{C}_c$, $c = 1...C$, should be performed. Here, we consider the following binarization procedure. Per-class counters are updated at each wrong prediction as

$$n_{y_i} = n_{y_i} + 1, \qquad n_{\hat{y}_i} = n_{\hat{y}_i} - 1. \tag{9}$$

After each training epoch, $\mathbf{C}_c$, $c = 1...C$, are binarized component-wise with thresholding by $\frac{n_c}{2}$.

We define Refined Centroid classifier with dot product similarity (RefCentrSim) as the instance of the above described classifier that uses dot product similarity sim_{dot} as sim as the similarity measure in (7). Similarly, Refined Centroid classifier with Hamming distance (RefCentrDist) is defined as the instance of the above described classifier that employs the Hamming distance dist_{Ham} as the distance measure dist in (6). In RefCentrDist, an additional restriction is also imposed: it does not update $\mathbf{C}_{\hat{y}_i}$ and $n_{\hat{y}_i}$ in cases when $n_{\hat{y}_i}$ would otherwise be reduced to zero.

3.3 Confidence-driven refining of class centroids

In this work, we investigate how the HDC classifiers RefCentrDist and RefCentrSim can be further improved through our Confidence-Driven Centroid approach. Building on our previous work [51], we define Confidence Centroid classifier with additive confidence bias and Hamming distance (ConfCentrAdd) as a classifier that, as RefCentrDist, updates (8) and (9) when $\hat{y}_i \neq y_i$. However, unlike RefCentrDist, ConfCentrAdd derives $\hat{y}_i$ not through (6) but rather using the following formula:

$$\hat{y}(\mathbf{x}_i) = \arg\min_c (\text{dist}_{\text{Ham}}(\mathbf{h}(\mathbf{x}_i), \mathbf{c}_c) + a_c), \qquad c = 1...C, \tag{10}$$

where

$$\begin{aligned} a_{c=y_i} &= a, \\ a_{c\neq y_i} &= 0. \end{aligned} \tag{11}$$

Here, $a \geq 0$ is an additive bias value applied to the correct class during training, ensuring that the correct classification output $\hat{y}_i = y_i$ is achieved only when all other class centroids are separated from the input hypervector $\mathbf{x}_i$ by a Hamming distance of at least a. This a value effectively quantifies the confidence level with which the correct class centroid must predict the correct class relative to the other centroids during training to prevent centroid updates. Thus, for ConfCentrAdd, correct predictions under (6) may not be sufficient to avoid an update if $a > 0$. However, when $a = 0$, (10) reduces to (6).

To determine a proper a value for improved classification on a given dataset, domain knowledge could be used. Alternatively, a can be treated as a model hyperparameter to be optimized through validation.

To make the range of a-values consistent across different hypervector dimensionalities D, we propose setting $a = \alpha D$, where now $0 \leq \alpha < 1$ serves as the hyperparameter.

Our second confidence-driven prediction approach, inspired by [7], [57], [58], leads to the Confidence Centroid classifier with multiplicative confidence bias and dot product similarity (ConfCentrMul) classifier. In this approach, $\hat{y}_i$ is calculated as

$$\hat{y}(\mathbf{x}_i) = \arg\max_c (\text{sim}_{\text{dot}}(\mathbf{h}(\mathbf{x}_i), \mathbf{c}_c) - a_c), \qquad c = 1...C, \tag{12}$$

where

$$\begin{aligned} a_{c=y_i} &= \alpha \ \text{sim}_{\text{dot}}(\mathbf{h}(\mathbf{x}_i), \mathbf{c}_{y_i}), \\ a_{c \neq y_i} &= 0, \end{aligned} \tag{13}$$

with $0 \leq \alpha < 1$ serving as the model hyperparameter to be selected.

For the final testing / evaluation, the trained confidence-driven centroid classifiers utilize either the prediction rule in (6) or (7).

3.4 Related work

To date, various studies have proposed adjustments to the classical centroid model for classification tasks in the context of HDC. For example, AdaptHD [29] introduces a learning rate in the iterative training procedure (8). It depends on the change in error rate over the last few iterations (i.e., iteration-dependent), on the difference in the similarity of the sample to the correct class and to the misclassified class (i.e., data-dependent), or on the combination of both. OnlineHD [34] further adds (or subtracts) a sample hypervector to (or from) the class centroid with a weight depending on their similarity.

In NeuralHD [35] and DistHD [38], insignificant or misleading dimensions of the class centroids are identified after which they are regenerated to enhance learning capabilities. In contrast to NeuralHD and DistHD, SparseHD [32] enforces sparsity by eliminating the least significant dimensions of the trained class centroids. In class-wise sparsity approach, each class centroid is independently sparsified, while for dimension-wise sparsity, insignificant dimensions shared across all learned centroids are eliminated.

QuantHD [30] binarizes class centroids. To limit the impact on the accuracy, it updates non-quantized centroids by the samples that are misclassified with the binary model. LeHDC [36] incorporates Binary Neural Networks approach for training class centroids. The classifier with non-binary centroids is used to accumulate small gradients and is updated with back-propagation, whereas the binary version is used in feed-forward pass and is updated after each epoch based on non-binary centroids.

In other related lines of HDC research addressing classification problems, let us note the following

ones. To shrink model size and speed up inference, CompHD [33] reduces hypervector dimensionality by splitting them and combining these splits with HDC operations. SemiHD [31] introduces a semi-supervised centroid model where initial training occurs on limited labeled data, which is then expanded by labeling previously unlabeled data samples that were confidently predicted by the model. The MultiCentroid approach of [37] further refines classification by constructing multiple centroids per class; when a sample is misclassified, it is added as a new centroid for its correct class, with less-populated centroids being removed or merged with the most similar ones to limit the number of per- class centroids.

The above-mentioned HDC centroid models (see also review by Vergés, Heddes, Nunes, et al. [28]) still only consider misclassified samples to update class centroids. In contrast, RefineHD [59] also trains with correctly classified samples, whose similarity to their class centroid is lower than the threshold value calculated as the averaged similarity of misclassified samples since the start of the epoch's start.

In our experimental investigation, we compare our results on 121 UCI datasets obtained with ConfCentrAdd and ConfCentrMul with those of the HDC models mentioned in this Section.

4 A unified setup for UCI dataset preprocessing and result evaluation

We observed that presently there is no fully standardized approach to managing dataset splits for the 121 UCI datasets from [39]. Existing studies employing these datasets (e.g., [28], [59]–[61]) adopt somewhat varied approaches in data standardization, hypervector standardization, and averaging results obtained with different random seeds. Along with diverse methods of transforming initial vector data into hypervectors, these differences contribute to incompatibility and inconsistency in the results obtained across studies.

To address these discrepancies, we propose a unified setup for evaluating HDC classification methods on the 121 UCI datasets. This setup includes input data and hypervector preprocessing (actually postprocessing), as well as result averaging across various hypervector realizations, tailored to the train / validation / test splits of the UCI datasets, to ensure fairer comparisons among different HDC classifiers.

The train / validation / test splits for the 121 UCI datasets were introduced in [39], are publicly available, and are widely used for evaluating classifiers. For 19 of these 121 datasets, there is a fixed train and test set available as the files {}_train_R.dat and {}_test_R.dat for each of these datasets. These datasets will be referred to as "separated datasets". The other 102 datasets do not have a test set separated from the train set and can be found in there respective {}_R.dat files. We refer to these datasets as "non-separated datasets".

4.1 Dataset splits

For completeness and the sake of clarity of further exposition, here we describe splitting of data for hyperparameter selection and final test for separated and non-separated datasets from [39].

4.1.1 Separated datasets

Here, for hyperparameter selection, only the fixed train set (i.e., {}_train_R.dat) is used. This set is split in half according to the indices provided in the file conxuntos.dat, forming two subsets: one is used to train the classification model with various hyperparameter configurations (i.e., train subset for validation), while the other subset is used to validate the performance of the models trained using those hyperparameter configurations (i.e., test subset for validation or just validation set).

For final testing / evaluation, the model is then trained on the entire fixed train set (i.e., {}_train_R.dat) using the hyperparameter configuration yielding the best performance on the validation set in the first part of the experiments. Finally, this fully trained model is tested / evaluated on the fixed test set (i.e., {}_test_R.dat).

4.1.2 Non-separated datasets

Here, the indices in conxuntos.dat are used to split the entire dataset {}_R.dat into two subsets of equal size. One subset is used as the train set for hyperparameter selection, while the other subset serves to asses the performance of classification models trained using each hyperparameter configuration.

The final testing / evaluation is performed using 4-fold cross-validation on the entire dataset {}_R.dat. Indices for these splits are provided in conxuntos_kfold.dat. The model is trained using the selected hyperparameter setting on 75% of the samples and tested / evaluated on the remaining 25%. This is repeated four times, i.e., for each of 4 folds.

It should be noted that this kind of data splitting is less than ideal. Specifically, final testing is conducted on samples that were also used in the hyperparameter selection process, which introduces data leakage from the training and/or validation sets into the test set. This contradicts standard machine learning protocols, where the final test set should be entirely independent of both the train and validation sets. Moreover, some of the non-separated datasets contain overlapping samples across the four test folds. Even though this is not a type of data leakage, it deviates from the conventional k-fold cross-validation approach, where the model is trained on $k-1$ folds and tested on the remaining kth fold in a cycle until each fold has served as the test set exactly once.

Despite these limitations, we adhere to this data-splitting procedure to facilitate comparisons with previously reported results on these datasets.

4.2 Preprocessing

In the context of HDC classification, "preprocessing" can refer to two distinct stages, given that input data undergoes transformation into hypervectors. The first stage involves preprocessing the original UCI vector data prior to its transformation into hypervectors. The second stage involves preprocessing the resulting hypervectors themselves before they are input to the classifier.

Here, we focus primarily on data preprocessing in the form of standardization. In standard machine learning practice, it is common to apply standardization parameters calculated from and applied to the training set, also to the validation or test sets. However, as we explain below, the separated and non-separated datasets require distinct approaches to standardization.

4.2.1 Non-separated datasets

In [39], they do not follow the common protocol. Instead, they standardize each non-separated dataset by scaling the entire dataset {}_R.dat as a whole. Their approach again introduces data leakage, as it makes the test folds not independent from the train folds.

In contrast, we propose to follow the standard protocol for the non-separated datasets. Specifically, for hyperparameter selection, both the train and validation sets should be standardized using the parameters derived from the train set. This approach also applies to the four training and test folds used in final testing / evaluation, where the training fold standardization parameters are reused to standardize both itself and the respective test fold.

4.2.2 Separated datasets

In the separated datasets from the UCI 121 collection, the test set usually has a very different data distribution from the train set. For instance, while the train set may include both negative and positive values, the test set may contain only positive values. Given this discrepancy, we propose deviating from the common standardization protocol for the separated datasets.

Specifically, for hyperparameter selection, both the train and validation sets should be standardized independently, with standardization parameters calculated separately for each set. The same applies to the final testing/evaluation stage, where the train and test sets should be standardized independently.

4.3 Obtaining final results

Since HDC classifiers use randomly initialized hypervectors, which commonly impact performance, experiments should therefore be repeated with multiple realizations of randomly generated hypervectors, each obtained with a distinct seed for the pseudo-random number generator.

Hyperparameter selection is conducted for each hyperparameter setting using multiple seeds, following the dataset splits and preprocessing procedures outlined in Sections 4.1 and 4.2. Validation accuracies are averaged across all seeds for each hyperparameter setting, enabling the selection of the setting with the highest mean validation accuracy for each dataset.

For final testing / evaluation, the chosen hyperparameter setting is used to train and test the model for each seed, following the data splitting and preprocessing outlined in Sections 4.1 and 4.2. Thus, for each dataset, we obtain the test accuracy averaged across multiple seeds (for separated datasets) or the test accuracy averaged across the four test folds and multiple seeds (for non-separated datasets).

5 Experiments and discussion of results

We evaluated classification performance of four classifiers from Section 3: RefCentrDist, RefCentrSim, ConfCentrAdd and ConfCentrMul on 121 UCI benchmark datasets.

We employed the data preprocessing methodology from Section 4.2 as follows. The minimalistic original data standardization required for the compositional method of input data vector transformation into hypervectors is quantization to the integer range $[0, Q]$. For each dataset, this was implemented by first calculating min-max values for each component of the original UCI vectors across all data samples from their subset / split, then rescaling each component to be in the real-valued range $[0, Q]$, and finally rounding to the closest integer. When the rescaling parameters determined for the train set were used for rescaling the test set and the values appeared outside $[0, Q]$, they were clipped. We have chosen to not standardize the binary hypervectors obtained from the original vectors.

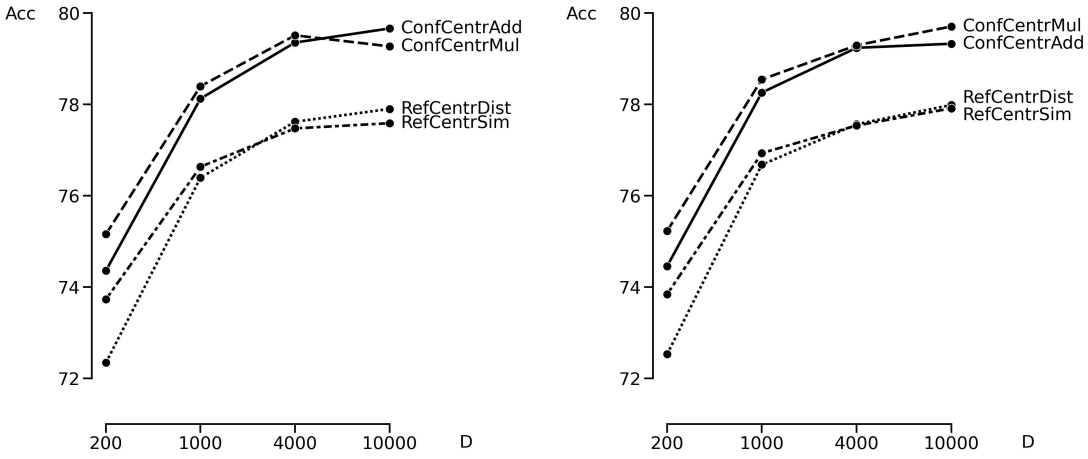
We experimented with two values of quantization levels $Q = \{20, 100\}$ and four values of HV dimensionality $D = \{200, 1000, 4000, 10000\}$. The range of the α hyperparameter values for selecting the proper one for particular dataset by validation was [0.00, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.7] for both ConfCentrAdd and ConfCentrMul. The number of epochs in the training procedure was 100, after which the centroid model at the epoch with the highest training accuracy was selected to be used during final testing / evaluation. Each experiment was repeated with ten different seeds for initializing pseudorandom number generator for generating realizations of hypervectors.

Table 1 provides the test accuracies of four classifiers averaged across all 121 datasets and realizations of hypervectors, also presented in Figure 1. Per-dataset results are included in table 2. We observe that the mean accuracy increases with increasing HV dimensionality, but the difference between $D = 4000$ and $D = 10000$ is minimal. No large differences are seen between $Q = 20$ and $Q = 100$. It is also clear that the pairs of classifiers: ConfCentrAdd and ConfCentrMul, as well as their classical versions RefCentrDist and RefCentrSim perform closely, while both ConfCentrAdd and ConfCentrMul, outperform

their corresponding classical variants.

Table 1: Mean test accuracies (in %) for four classifiers, averaged across all 121 datasets and seeds, using the best validated α value. Q is the number of quantization levels, D is hypervector dimensionality. Standard deviations are shown in parentheses.

Q	Model \ D	200	1000	4000	10000
20	RefCentrDist	72.34 (±17.27)	76.39 (±16.42)	77.62 (±16.33)	77.90 (±16.34)
	RefCentrSim	73.73 (±17.03)	76.63 (±16.71)	77.47 (±16.75)	77.58 (±16.88)
	ConfCentrAdd	74.35 (±16.19)	78.12 (±15.30)	79.35 (±15.01)	**79.66** (±15.02)
	ConfCentrMul	75.15 (±16.77)	78.39 (±15.65)	79.51 (±15.25)	79.27 (±15.59)
100	RefCentrDist	72.52 (±17.40)	76.68 (±16.35)	77.56 (±16.66)	77.98 (±16.57)
	RefCentrSim	73.84 (±17.12)	76.93 (±16.67)	77.53 (±16.94)	77.91 (±16.68)
	ConfCentrAdd	74.45 (±16.10)	78.26 (±15.69)	79.24 (±15.71)	79.33 (±15.76)
	ConfCentrMul	75.22 (±16.69)	78.54 (±15.80)	79.29 (±15.63)	**79.70** (±15.45)



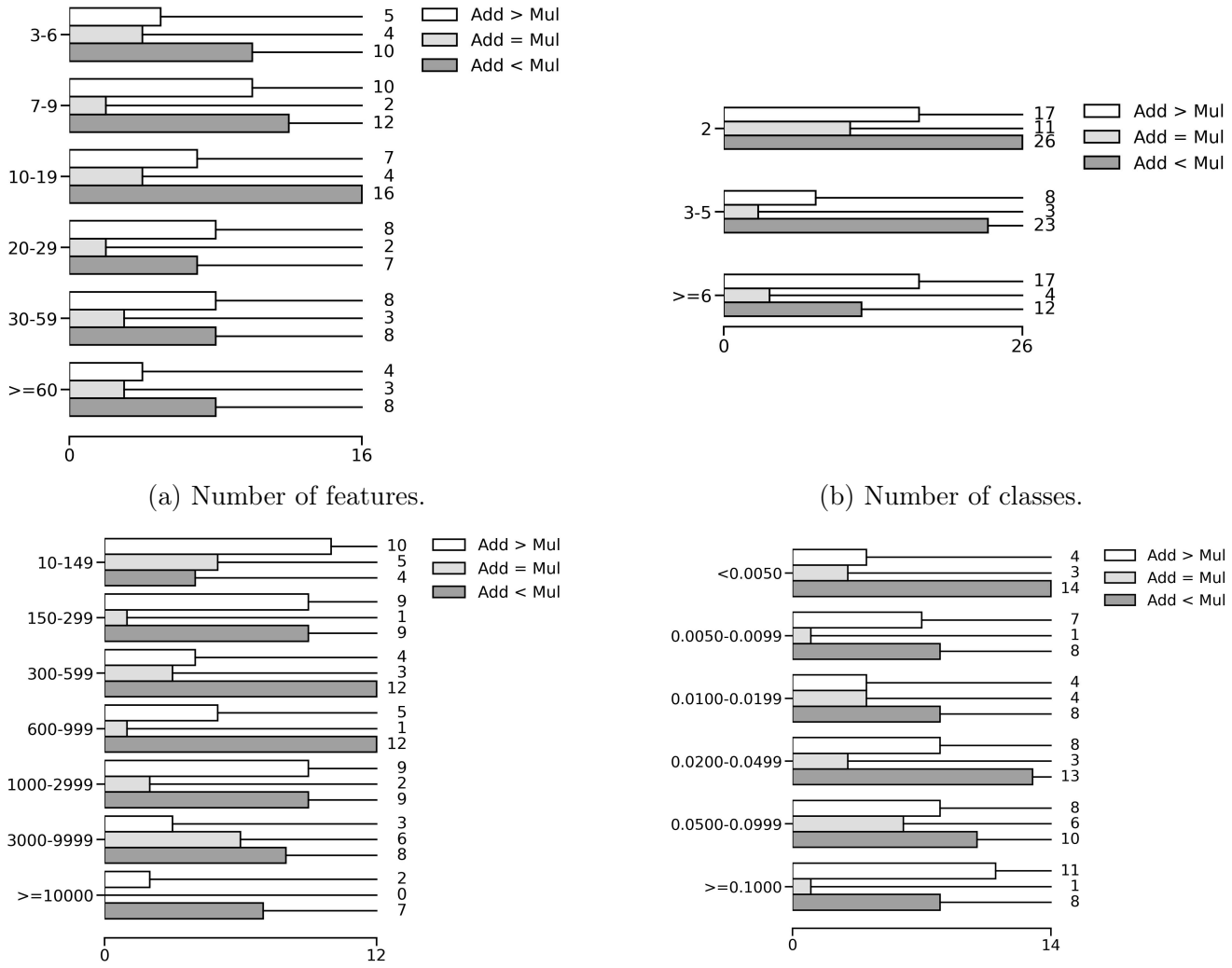
(a) $Q = 20$. (b) $Q = 100$.

Figure 1: The effect of the number of quantization levels Q and the vector dimensionality D on the final mean test accuracy Acc (as reported in Table 1). (a) $Q = 20$, (b) $Q = 100$.

Comparison between ConfCentrAdd and ConfCentrMul is shown in Figure 2 for the hyperparameter values resulting in the best test accuracy (bold in Table 1). This corresponds to $Q = 20$, $D = 10000$ for ConfCentrAdd and $Q = 100$, $D = 10000$ for ConfCentrMul. The figure shows for how many datasets ConfCentrAdd performs better, equally well, or worse than ConfCentrMul. The results are grouped based on the number of features, classes, samples, and the ratio of the number of features to the number of samples in the datasets. We observe that ConfCentrMul performs better for datasets with less than 20 features, less than 6 classes, features/samples number ratio smaller than 0.1, and with more than 300 samples.

Figures 3(a) and (b) present the mean number of training epochs to obtain the classifier with the best training accuracy, as well as the mean time to perform one epoch ($Q = 20$). The number of training epochs is smaller for a larger HV dimensionality, whereas the training time is much larger for a larger HV dimensionality. ConfCentrAdd and ConfCentrMul generally require less number of epochs than classical

RefCentrDist and RefCentrSim. Both RefCentrDist and ConfCentrAdd tend to require more time than RefCentrSim and ConfCentrMul, but this difference becomes smaller for larger HV dimension.



(a) Number of features.

(b) Number of classes.

(c) Number of samples.

(d) Features / samples number ratio.

Figure 2: Comparison of ConfCentrAdd ($Q = 20$ and $D = 10000$) and ConfCentrMul ($Q = 100$ and $D = 10000$) for 121 UCI datasets grouped based on (a) feature number, (b) class number, (c) sample number, and (d) features / samples number ratio.

Figure 4 compares the mean accuracy across 121 UCI datasets of ConfCentrAdd and ConfCentrMul (in white) and the results of other HDC classifiers discussed in Section 3.4 (in gray) taken from [28] and [59]. The best HDC results in the literature, DistHD [28], [38] and RefineHD [59], reach 76.95% and 76.70%, while ConfCentrAdd ($Q = 20$ and $D = 10000$) and ConfCentrMul ($Q = 100$ and $D = 10000$) obtain 79.66% and 79.70%, respectively. Note that [28] and [59] use the HV dimensionality $D = 10000$ with flipping $D/(2Q)$ bits in the linear mapping (Section 2.3), such that their effective HV dimensionality was only 5000. Nevertheless, ConfCentrAdd and ConfCentrMul at $D = 4000$ show mean accuracy of 79.35% and 79.51%, respectively, still outperforming DistHD and RefineHD.

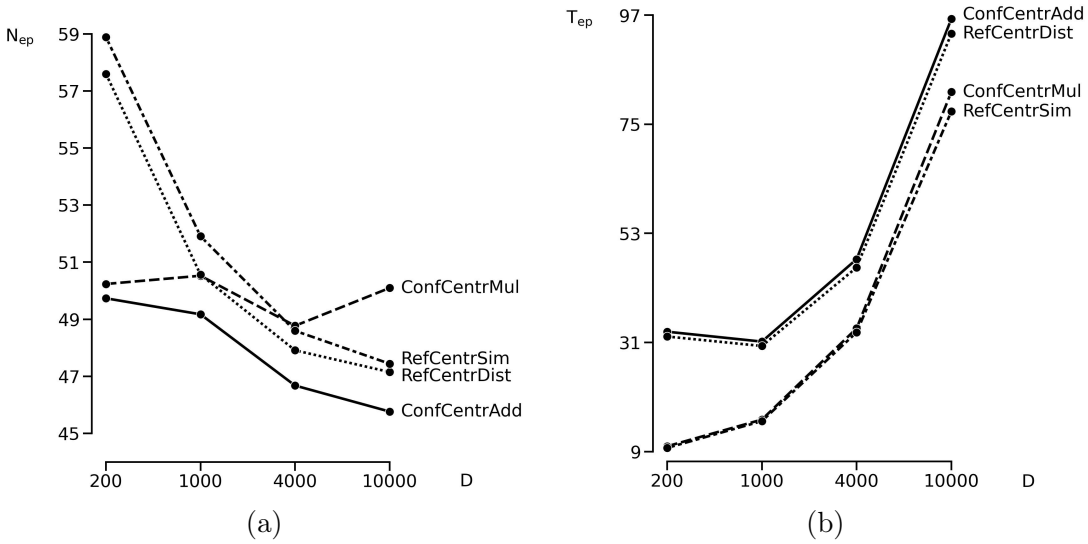


Figure 3: (a) Mean number of training epochs N_{ep} until the best training accuracy, and (b) mean time T_{ep} (in seconds) per epoch. Results are averaged across 121 UCI datasets and random hypervector realizations.

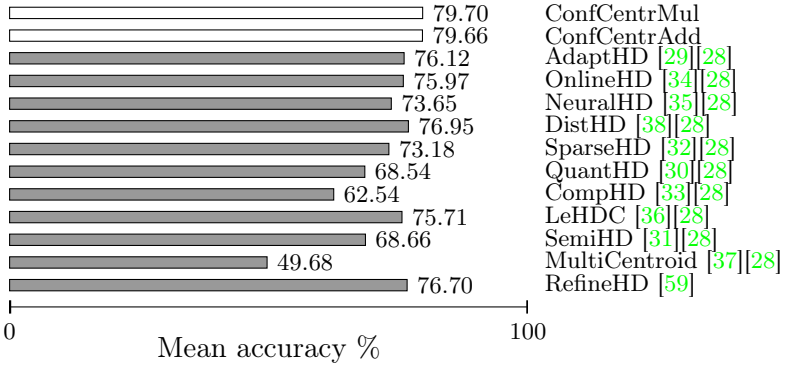


Figure 4: Comparison of the mean classification accuracy results on 121 UCI datasets: the best result of ConfCentrAdd ($Q = 20$ and $D = 10000$) and ConfCentrMul ($Q = 100$ and $D = 10000$) (shown in white), and results of other HDC classifiers (Section 3.4) as reported in [28] and [59]) (shown in gray).

Table 2: Detailed results with the RefCentrDist, RefCentrSim, ConfCentrAdd and ConfCentrMul models for 121 UCI datasets. $D = 10000$, $Q = 20$ and $Q = 100$.

	$Q = 20$				$Q = 100$			
dataset	RefCentr Dist	RefCentr Sim	ConfCentr Add	ConfCentr Mul	RefCentr Dist	RefCentr Sim	ConfCentr Add	ConfCentr Mul
abalone	59.54	59.98	61.06	60.25	60.13	60.34	61.63	62.19
acute-inflammation	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
acute-nephritis	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
adult	84.30	85.03	80.34	85.03	84.49	85.00	80.75	85.00
annealing	41.00	40.40	73.00	69.00	38.40	38.60	71.60	87.20
arrhythmia	71.59	71.81	71.59	71.81	71.73	71.33	72.48	71.33
audiology-std	72.80	73.60	76.00	72.00	72.80	74.40	76.00	73.60
balance-scale	93.24	92.50	92.24	92.50	92.88	93.56	93.08	92.79
balloons	77.50	78.75	83.75	80.00	77.50	77.50	80.00	80.00
bank	89.01	89.36	89.84	90.21	89.11	89.15	89.96	89.45
blood	77.38	76.50	77.38	79.25	76.84	76.84	76.31	78.02
breast-cancer	69.44	67.89	75.00	75.85	67.75	67.89	75.00	73.17
breast-cancer-wisc	96.83	96.77	97.40	97.37	96.86	96.89	97.09	97.60
breast-cancer-wisc-diag	95.95	95.92	96.73	96.37	95.67	95.81	96.48	96.55
breast-cancer-wisc-prog	75.51	74.90	80.10	74.90	77.55	78.37	80.71	76.33
breast-tissue	69.42	66.92	69.62	68.85	69.62	65.77	71.54	65.58
car	89.57	92.55	90.30	92.55	89.78	93.04	89.07	93.04

Continued on next page

Table 2 – continued from previous page

dataset	$Q = 20$				$Q = 100$			
	RefCentr Dist	RefCentr Sim	ConfCentr Add	ConfCentr Mul	RefCentr Dist	RefCentr Sim	ConfCentr Add	ConfCentr Mul
cardiotocography-10clases	81.46	75.20	81.73	75.20	82.32	76.34	82.32	76.33
cardiotocography-3clases	92.12	92.22	91.98	91.03	92.89	93.19	92.02	91.53
chess-krvk	31.99	32.55	35.83	36.98	31.93	32.62	35.25	36.86
chess-krvkp	97.38	97.45	97.38	97.46	97.22	97.20	97.22	97.48
congressional-voting	60.28	59.72	60.69	61.28	59.86	59.91	61.15	61.70
conn-bench-sonar-mines-rocks	79.13	79.90	81.15	82.31	81.63	81.35	83.08	82.98
conn-bench-vowel-deterding	92.03	73.42	88.74	74.07	92.68	74.24	92.99	76.32
connect-4	81.62	82.09	81.62	82.09	81.40	82.02	81.40	82.02
contrac	48.04	50.52	50.86	51.83	47.70	50.18	51.15	51.79
credit-approval	83.81	83.92	87.33	86.89	84.22	84.22	87.35	87.03
cylinder-bands	76.02	78.44	75.94	76.41	75.39	77.23	75.39	75.39
dermatology	97.58	97.53	97.31	97.42	97.47	97.36	97.25	97.25
echocardiogram	78.03	78.64	85.61	82.58	81.06	80.15	85.61	85.76
ecoli	79.46	81.85	85.65	84.70	80.36	82.38	86.13	85.24
energy-y1	91.77	92.92	88.67	91.69	93.23	93.72	89.77	93.72
energy-y2	85.91	87.40	85.91	86.25	88.26	89.32	88.26	86.51
fertility	83.80	83.00	83.80	88.00	84.00	84.40	84.60	87.60
flags	55.52	55.00	55.52	57.92	56.04	54.79	55.63	57.50
glass	69.62	69.25	72.08	68.96	70.94	69.06	71.04	69.06
haberman-survival	71.25	70.92	71.45	71.45	70.66	71.18	73.09	71.84
hayes-roth	66.43	49.29	65.00	50.00	68.57	52.86	58.57	47.14
heart-cleveland	56.84	55.92	58.55	59.47	55.59	56.32	58.95	60.20
heart-hungarian	78.42	77.88	85.14	84.93	78.42	78.56	85.14	85.62
heart-switzerland	38.55	38.06	38.55	38.06	38.71	37.74	39.84	36.61
heart-va	29.50	30.30	30.30	30.30	30.10	32.70	29.20	30.90
hepatitis	83.72	83.85	83.72	83.33	81.79	82.56	83.46	83.46
hill-valley	50.13	50.59	50.17	50.33	50.69	51.06	51.88	51.06
horse-colic	78.53	77.06	85.88	86.18	78.24	80.29	85.29	87.35
ilpd-indian-liver	67.98	71.85	67.95	71.23	69.11	71.75	71.99	71.16
image-segmentation	89.47	89.52	89.84	90.89	89.11	88.11	85.05	88.11
ionosphere	89.83	89.83	91.93	94.77	89.83	89.03	92.05	94.32
iris	95.41	95.81	96.49	96.76	95.68	95.81	96.08	96.35
led-display	71.66	71.14	73.34	74.02	71.78	70.48	73.20	73.78
lenses	74.17	73.33	74.17	73.33	70.00	73.33	70.00	73.33
letter	79.73	69.28	80.69	69.28	80.35	69.26	81.23	69.26
libras	75.44	74.17	76.56	71.11	75.00	74.94	77.89	77.39
low-res-spect	89.62	89.59	89.62	89.81	90.15	90.23	90.15	90.15
lung-cancer	51.25	47.50	51.25	47.50	50.00	48.13	50.00	48.13
lymphography	85.14	86.35	85.27	84.86	85.95	86.08	85.95	85.81
magic	83.77	83.79	83.77	83.88	84.59	84.57	84.59	84.57
mammographic	80.15	79.65	81.81	81.40	79.83	79.56	81.08	80.90
miniboone	84.39	84.45	84.37	84.57	87.03	86.91	87.03	86.91
molec-biol-promoter	87.50	85.96	87.50	85.96	86.54	85.58	86.54	85.58
molec-biol-splice	91.86	92.15	91.86	92.96	91.74	92.14	91.74	91.89
monks-1	81.71	80.60	83.75	83.24	81.06	81.44	83.15	83.75
monks-2	94.58	95.28	90.19	93.66	94.91	94.58	92.82	93.70
monks-3	91.67	91.76	86.81	92.08	91.44	92.27	86.62	93.19
mushroom	99.78	99.79	99.99	99.99	99.77	99.80	99.99	99.99
musk-1	66.05	84.12	66.05	82.48	65.55	83.53	65.55	82.27
musk-2	95.51	96.87	95.51	96.87	94.80	95.64	94.80	95.64
nursery	92.11	94.12	92.77	94.77	92.04	93.99	92.56	94.88
oocytes_merluccius_nucleus_4d	68.94	73.61	71.57	73.61	68.90	73.55	70.16	73.59
oocytes_merluccius_states_2f	90.39	90.22	91.18	91.04	90.57	90.55	91.96	92.08
oocytes_trisopterus_nucleus_2f	60.90	61.75	72.02	74.32	61.32	62.11	70.59	73.09
oocytes_trisopterus_states_5b	89.93	89.39	90.18	90.37	91.25	91.03	91.23	91.21
optical	93.87	93.99	95.82	96.17	93.86	93.96	95.84	96.13
ozone	96.70	96.47	97.01	97.11	96.81	96.49	97.08	97.13
page-blocks	94.86	94.69	95.23	94.81	95.22	95.30	95.71	95.29
parkinsons	88.27	88.06	89.59	88.57	88.57	89.29	90.10	89.29
pendigits	95.27	95.58	94.29	96.15	95.53	95.59	94.62	96.54

Continued on next page

Table 2 – continued from previous page

dataset	$Q = 20$				$Q = 100$			
	RefCentr Dist	RefCentr Sim	ConfCentr Add	ConfCentr Mul	RefCentr Dist	RefCentr Sim	ConfCentr Add	ConfCentr Mul
pima	73.46	74.43	75.03	74.77	73.15	73.78	74.27	75.03
pittsburg-bridges-MATERIAL	93.27	92.69	93.08	92.69	93.08	92.88	92.69	92.69
pittsburg-bridges-REL-L	71.35	74.23	71.15	71.35	71.54	73.85	74.04	73.85
pittsburg-bridges-SPAN	61.09	60.43	65.22	68.70	58.70	61.30	60.43	58.26
pittsburg-bridges-T-OR-D	88.20	89.40	90.80	88.40	89.80	88.80	89.80	89.00
pittsburg-bridges-TYPE	59.23	60.19	71.15	72.31	62.31	59.81	70.38	71.15
planning	54.11	53.78	57.89	62.78	55.33	54.11	60.67	63.78
plant-margin	78.90	78.11	79.15	77.95	78.93	78.69	79.55	78.55
plant-shape	53.39	40.11	53.20	50.41	54.85	42.74	55.23	50.31
plant-texture	75.19	74.25	78.50	74.25	74.71	74.50	79.73	74.55
post-operative	48.64	47.50	66.14	72.72	47.50	46.14	67.50	72.50
primary-tumor	44.76	45.00	47.07	51.89	44.39	45.98	46.71	51.65
ringnorm	91.80	97.97	91.80	97.98	92.32	98.08	92.32	98.10
seeds	91.92	92.21	91.92	91.92	92.02	92.02	92.31	92.69
semeion	90.09	89.74	91.37	92.55	90.05	90.04	91.87	92.73
soybean	86.60	86.81	86.86	85.90	86.76	86.91	87.61	86.65
spambase	90.91	90.87	90.91	90.87	92.47	92.25	92.47	92.25
spect	67.42	66.99	69.78	66.99	66.45	65.70	69.35	66.99
spectf	44.81	40.86	54.22	40.86	42.03	46.52	42.03	52.30
statlog-australian-credit	64.51	67.27	68.02	68.69	64.56	65.81	68.08	69.13
statlog-german-credit	74.20	75.74	76.36	75.74	74.46	76.16	75.92	76.14
statlog-heart	82.91	83.28	86.94	88.51	82.54	82.31	86.86	88.21
statlog-image	96.12	96.06	96.12	95.21	96.66	96.74	96.66	95.94
statlog-landsat	83.74	83.41	84.83	82.25	84.52	84.09	85.18	81.37
statlog-shuttle	58.59	52.88	75.92	57.82	51.96	54.47	38.20	63.85
statlog-vehicle	71.02	71.30	71.09	70.52	72.68	71.30	72.13	71.30
steel-plates	70.33	69.05	71.56	70.66	70.10	69.71	70.10	70.97
synthetic-control	96.23	95.80	98.07	97.30	96.30	96.60	98.17	98.27
teaching	51.97	51.84	50.13	51.58	53.16	54.47	50.53	52.76
thyroid	89.15	90.01	89.15	90.01	89.50	88.37	89.50	88.37
tic-tac-toe	97.55	97.59	98.01	97.87	97.45	97.45	97.95	97.41
titanic	77.86	77.99	77.78	78.05	77.84	77.97	77.86	78.05
trains	87.50	80.00	87.50	80.00	82.50	77.50	82.50	77.50
twonorm	96.63	96.77	97.41	97.47	96.54	96.76	97.53	97.55
vertebral-column-2clases	82.21	82.34	83.12	81.57	83.51	81.56	83.51	81.57
vertebral-column-3clases	78.77	77.79	79.94	80.39	81.88	82.08	79.87	82.08
wall-following	91.22	91.90	91.22	91.90	92.45	93.67	92.45	93.67
waveform	83.94	84.12	85.47	85.59	83.82	83.96	85.90	85.70
waveform-noise	85.28	85.06	85.12	85.43	85.28	85.23	86.15	86.14
wine	97.16	97.16	98.41	98.07	97.16	97.27	98.18	98.64
wine-quality-red	57.56	57.90	59.65	59.51	58.06	59.40	60.78	60.49
wine-quality-white	52.19	53.66	54.69	54.86	52.09	53.84	54.43	55.22
yeast	54.96	55.62	58.89	57.32	54.95	54.95	59.96	58.36
zoo	99.00	99.20	99.00	99.20	99.00	99.00	99.00	99.00
mean	77.90	77.58	79.66	79.27	77.98	77.91	79.33	79.70
standard deviation	16.34	16.88	15.02	15.58	16.57	16.68	15.76	15.45

6 Conclusion

We conducted an extensive comparative study examining the classification accuracy of two HDC-based centroid classifiers that differ from previous proposals by leveraging training driven by either additive or multiplicative confidence levels in class prediction. Both confidence-driven classifiers are designed to be

hardware-efficient, operating with binary hypervectors and binary class representations, particularly in prediction mode, making them well-suited for resource-constrained applications.

Our results, obtained across a diverse collection of 121 UCI datasets, demonstrate that both our confidence-driven classifiers outperform their classical counterparts as well as other traditional HDC classifiers evaluated on this dataset collection.

We found that the number of quantization levels for transforming input data to hypervectors has minimal impact on our classifier performance, whereas hypervector dimensionality plays a critical role: increasing dimensionality enhances accuracy but requires more processing time. When comparing our classifiers, ConfCentrAdd based on an additive confidence threshold and Hamming distance between the sample and the centroid hypervectors, tends to perform better on datasets with a higher number of classes and fewer samples. While the ConfCentrMul classifier with multiplicative confidence threshold, employing dot product similarity, achieves slightly higher overall accuracy and excels on datasets with fewer features and classes.

An interesting direction for future research would be the development of hardware implementations optimized for gate efficiency and energy consumption, along with evaluating classification performance in practical applications of wearable Internet of Things systems.

The work of L.S. 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.

References

[1] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," Cognitive Computation, vol. 1, pp. 139–159, 2 2009, ISSN: 18669956. DOI: https://doi.org/10.1007/s12559-009-9009-8.

[2] R. W. Gayler, "Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience," in Proceedings of the ICCS/ASCS International Conference on Cognitive Science, 2003, pp. 1–6. DOI: https://doi.org/10.48550/arXiv.cs/0412059.

[3] A. Rahimi, P. Kanerva, L. Benini, and J. M. Rabaey, "Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of exg signals," Proceedings of the IEEE, vol. 107, pp. 123–143, 1 Jan. 2019, ISSN: 15582256. DOI: https://doi.org/10.1109/JPROC.2018.2871163.

[4] D. Widdows and T. Cohen, "Reasoning with vectors: A continuous model for fast robust inference," Logic Journal of the IGPL, vol. 23, pp. 141–173, 2 Jul. 2015, ISSN: 13689894. DOI: https://doi.org/10.1093/jigpal/jzu028.

[5] R. Vdovychenko and V. Tulchinsky, "Increasing the semantic storage density of sparse distributed memory," Cybernetics and Systems Analysis, vol. 58, pp. 331–342, 3 May 2022, ISSN: 15738337. DOI: https://doi.org/10.1007/s10559-022-00465-y.

[6] R. Vdovychenko and V. Tulchinsky, "Sparse distributed memory for sparse distributed data," in 2023, vol. 542, pp. 74–81. DOI: https://doi.org/10.1007/978-3-031-16072-1_5.

[7] D. A. Rachkovskij, "Linear classifiers based on binary distributed representations," Information Theories and Applications, vol. 14, pp. 270–274, 3 2007.

[8] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in Proceedings of the International Symposium on Low Power Electronics and Design, IEEE, Aug. 2016, pp. 64–69, ISBN: 9781450341851. DOI: https://doi.org/10.1145/2934583.2934624.

[9] I. Misuno, D. Rachkovskij, S. Slipchenko, and A. Sokolov, "Searching for text information with the help of vector representations," Problems of Programming, vol. 4, pp. 50–59, 2005.

[10] M. Imani, D. Kong, A. Rahimi, and T. Rosing, "Voicehd: Hyperdimensional computing for efficient speech recognition," in IEEE International Conference on Rebooting Computing (ICRC), 2017, pp. 1–8, ISBN: 9781538615539. DOI: https://doi.org/10.1109/ICRC.2017.8123650.

[11] Y. Kim, M. Imani, and T. S. Rosing, "Efficient human activity recognition using hyperdimensional computing," in Proceedings of the 8th International Conference on the Internet of Things, Association for Computing Machinery, Oct. 2018, pp. 1–6, ISBN: 9781450365642. DOI: https://doi.org/10.1145/3277593.3277617.

[12] A. Moin, A. Zhou, A. Rahimi, et al., "A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition," Nature Electronics, vol. 4, pp. 54–63, 1 Jan. 2021, ISSN: 25201131. DOI: https://doi.org/10.1038/s41928-020-00510-8.

[13] A. Zhou, R. Muller, and J. Rabaey, "Memory-efficient, limb position-aware hand gesture recognition using hyperdimensional computing," in TinyML Research Symposium, Mar. 2021, pp. 1–8. DOI: https://doi.org/10.48550/arXiv.2103.05267. [Online]. Available: http://arxiv.org/abs/2103.05267.

[14] K. Schlegel, P. Neubert, and P. Protzel, "Hdc-minirocket: Explicit time encoding in time series classification with hyperdimensional computing," in International Joint Conference on Neural Network, Feb. 2022, pp. 1–8. DOI: https://doi.org/10.48550/arXiv.2202.08055. [Online]. Available: http://arxiv.org/abs/2202.08055.

[15] D. A. Rachkovskij, "Shift-equivariant similarity-preserving hypervector representations of sequences," Cognitive Computation, vol. 16, pp. 909–923, 2024. DOI: https://doi.org/10.1007/s12559-024-10258-4.

[16] D. Kleyko, S. Khan, E. Osipov, and S. P. Yong, "Modality classification of medical images with distributed representations based on cellular automata reservoir computing," in Proceedings - International Symposium on Biomedical Imaging, IEEE Computer Society, Jun. 2017, pp. 1053–1056, ISBN: 9781509011711. DOI: https://doi.org/10.1109/ISBI.2017.7950697.

[17] N. Watkinson, T. Givargis, V. Joe, A. Nicolau, and A. Veidenbaum, "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, IEEE, 2021, pp. 3970–3973, ISBN: 9781728111797. DOI: https://doi.org/10.1109/EMBC46164.2021.9630898.

[18] V. Lukovich, A. Goltsev, and D. Rachkovskij, "Neural network classifiers for micromechanical equipment diagnostics and micromechanical product quality inspection," in EUFIT, 1997, pp. 534–536.

[19] E. M. Kussul, L. M. Kasatkina, D. A. Rachkovskij, and D. C. Wunsch, "Application of random threshold neural networks for diagnostics of micro machine tool condition," in IEEE International Joint Conference on Neural Networks Proceedings, IEEE World Congress on Computational Intelligence, 1998, pp. 241–244. DOI: https://doi.org/10.1109/IJCNN.1998.682270.

[20] A. Goltsev and D. Rachkovskij, "Combination of the assembly neural network with a perceptron for recognition of handwritten digits arranged in numeral strings," Pattern Recognition, vol. 38, pp. 315–322, 3 Mar. 2005, ISSN: 00313203. DOI: https://doi.org/10.1016/j.patcog.2004.09.001.

[21] A. X. Manabat, C. R. Marcelo, A. L. Quinquito, and A. Alvarez, "Performance analysis of hyperdimensional computing for character recognition," in International Symposium on Multimedia and Communication Technology (ISMAC), 2019, pp. 1–5. DOI: https://doi.org/10.1109/ISMAC.2019.8836136.

[22] D. A. Rachkovskij, "Representation of spatial objects by shift-equivariant similarity-preserving hypervectors," Neural Computing and Applications, pp. 22 387–22 403, 2022, ISSN: 14333058. DOI: https://doi.org/10.1007/s00521-022-07619-1.

[23] L. Smets, W. V. Leekwijck, I. J. Tsang, and S. Latré, "An encoding framework for binarized images using hyperdimensional computing," Frontiers in Big Data, vol. 7, Jun. 2024, ISSN: 2624-909X. DOI: https://doi.org/10.3389/fdata.2024.1371518. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fdata.2024.1371518/full.

[24] P. Neubert, S. Schubert, and P. Protzel, "An introduction to hyperdimensional computing for robotics," KI - Kunstliche Intelligenz, vol. 33, pp. 319–330, 4 Dec. 2019, ISSN: 16101987. DOI: https://doi.org/10.1007/s13218-019-00623-z.

[25] D. Kleyko, D. A. Rachkovskij, E. Osipov, and A. Rahimi, "A survey on hyperdimensional computing aka vector symbolic architectures, part ii: Applications, cognitive models, and challenges," ACM Computing Serveys, vol. 55, pp. 1–52, 9 2023. DOI: https://doi.org/10.1145/3558000.

[26] L. Ge and K. K. Parhi, "Classification using hyperdimensional computing: A review," IEEE Circuits and Systems Magazine, vol. 20, pp. 30–47, 2 2020. DOI: https://doi.org/10.1109/MCAS.2020.2988388.

[27] S. Aygun, M. S. Moghadam, M. H. Najafi, and M. Imani, "Learning from hypervectors: A survey on hypervector encoding," pp. 1–14, Aug. 2023. DOI: https://doi.org/10.48550/arXiv.2308.00685. [Online]. Available: http://arxiv.org/abs/2308.00685.

[28] P. Vergés, M. Heddes, I. Nunes, T. Givargis, and A. Nicolau, "Classification using hyperdimensional computing: A review with comparative analysis," pp. 1–49, 2023. DOI: https://doi.org/10.21203/rs.3.rs-3425561/v1. [Online]. Available: https://doi.org/10.21203/rs.3.rs-3425561/v1.

[29] M. Imani, J. Morris, S. Bosch, H. Shu, G. D. Micheli, and T. Rosing, "Adapthd: Adaptive efficient training for brain-inspired hyperdimensional computing," in IEEE Biomedical Circuits and Systems Conference (BioCAS), 2019, pp. 1–4, ISBN: 9781509006175. DOI: https://doi.org/10.1109/BIOCAS.2019.8918974.

[30] M. Imani, S. Bosch, S. Datta, et al., "Quanthd: A quantization framework for hyperdimensional computing," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, pp. 2268–2278, 10 Oct. 2019, ISSN: 19374151. DOI: https://doi.org/10.1109/TCAD.2019.2954472.

[31] M. Imani, S. Bosch, M. Javaheripi, et al., "Semihd: Semi-supervised learning using hyperdimensional computing," in 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2019, pp. 1–8, ISBN: 9781728123509. DOI: https://doi.org/10.1109/ICCAD45719.2019.8942165.

[32] M. Imani, S. Salamat, B. Khaleghi, M. Samragh, F. Koushanfar, and T. Rosing, "Sparsehd: Algorithm-hardware co-optimization for efficient high-dimensional computing," in Proceedings - 27th IEEE International Symposium on Field-Programmable Custom Computing Machines, FCCM 2019, IEEE, Apr. 2019, pp. 190–198, ISBN: 9781728111315. DOI: https://doi.org/10.1109/FCCM.2019.00034.

[33] J. Morris, M. Imani, S. Bosch, A. Thomas, H. Shu, and T. Rosing, "Comphd: Efficient hyperdimensional computing using model compression," in IEEE/ACM International Symposium on Low Power Electronics and Design, 2019, pp. 1–6. DOI: https://doi.org/10.1109/ISLPED.2019.8824908.

[34] A. Hernández-Cano, N. Matsumoto, E. Ping, and M. Imani, "Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system," in Design, Automation and Test in Europe Conference and Exhibition (DATE), 2021, pp. 1–6. DOI: https://doi.org/10.23919/DATE51398.2021.9474107. [Online]. Available: https://gitlab.com/biaslab/onlinehd.

[35] Z. Zou, Y. Kim, F. Imani, H. Alimohamadi, R. Cammarota, and M. Imani, "Scalable edge-based hyperdimensional learning system with brain-like neural adaptation," in International Conference for High Performance Computing, Networking, Storage and Analysis, SC, IEEE Computer Society, Nov. 2021, pp. 1–15, ISBN: 9781450384421. DOI: https://doi.org/10.1145/3458817.3480958.

[36] S. Duan, Y. Liu, S. Ren, and X. Xu, "Lehdc: Learning-based hyperdimensional computing classifier," in Design Automation Conference, Mar. 2022, pp. 1–7. DOI: https://doi.org/10.48550/arXiv.2203.09680. [Online]. Available: http://arxiv.org/abs/2203.09680.

[37] U. Pale, T. Teijeiro, and D. Atienza, "Multi-centroid hyperdimensional computing approach for epileptic seizure detection," Frontiers in Neurology, vol. 13, pp. 1–13, Mar. 2022, ISSN: 16642295. DOI: https://doi.org/10.3389/fneur.2022.816294.

[38] J. Wang, S. Huang, and M. Imani, "Disthd: A learner-aware dynamic encoding method for hyperdimensional classification," in Proceedings - Design Automation Conference, vol. 2023-July, IEEE, 2023, pp. 1–6, ISBN: 9798350323481. DOI: https://doi.org/10.1109/DAC56929.2023.10247876.

[39] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, and A. Fernández-Delgado, "Do we need hundreds of classifiers to solve real world classification problems?" Journal of Machine Learning Research, vol. 15, pp. 3133–3181, 1 2014. [Online]. Available: http://www.mathworks.es/products/neural-network..

[40] T. A. Plate, "Holographic reduced representations," IEEE Transactions on Neural Networks, vol. 6, pp. 623–641, 3 1995, ISSN: 19410093. DOI: https://doi.org/10.1109/72.377968.

[41] T. A. Plate, Holographic reduced representation: distributed representation for cognitive structures. CSLI Publications, 2003.

[42] R. W. Gayler, "Multiplicative binding, representation operators & analogy," in Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences, 1998, pp. 1–4.

[43] P. Kanerva, "The spatter code for encoding concepts at many levels," in International Conference on Artificial Neural Networks (ICANN), 1994, pp. 226–229. DOI: https://doi.org/10.1007/978-1-4471-2097-1_52.

[44] P. Kanerva, "Binary spatter-coding of ordered k-tuples," in Artificial Neural Networks - ICANN, 1996, pp. 869–873. DOI: https://doi.org/10.1007/3-540-61510-5_146.

[45] M. Laiho, J. H. Poikonen, P. Kanerva, and E. Lehtonen, "High-dimensional computing with sparse vectors," in IEEE Biomedical Circuits and Systems Conference: Engineering for Healthy Minds and Able Bodies, BioCAS 2015 - Proceedings, IEEE, Dec. 2015, pp. 1–4, ISBN: 9781479972333. DOI: https://doi.org/10.1109/BioCAS.2015.7348414.

[46] D. A. Rachkovskij, S. V. Slipchenko, E. M. Kussul, and T. N. Baidyk, "Sparse binary distributed encoding of scalars," Journal of Automation and Information Sciences, vol. 37, pp. 12–23, 6 2005. DOI: https://doi.org/10.1615/JAutomatInfScien.v37.i6.20. [Online]. Available: https://www.researchgate.net/publication/245407061.

[47] D. Kleyko, A. Rahimi, D. A. Rachkovskij, E. Osipov, and J. M. Rabaey, "Classification and recall with binary hyperdimensional computing: Tradeoffs in choice of density and mapping characteristics," IEEE Transactions on Neural Networks and Learning Systems, vol. 29, pp. 5880–5898, 12 Dec. 2018, ISSN: 21622388. DOI: https://doi.org/10.1109/TNNLS.2018.2814400.

[48] D. A. Rachkovskij and T. V. Fedoseeva, "On audio signals recognition by multilevel neural network," in The International Symposium on Neural Networks and Neural Computing - NEURONET'90, 1990, pp. 281–283. [Online]. Available: https://www.researchgate.net/publication/341966170.

[49] E. Kussul and D. A. Rachkovskij, "On image texture recognition by associative-projective neurocomputer," in Proceedings of the ANNIE'91 conference "Intelligent engineering systems through artificial neural networks", 1991, pp. 453–458. [Online]. Available: https://www.researchgate.net/publication/342466737.

[50] D. A. Rachkovskij, S. V. Slipchenko, I. S. Misuno, E. M. Kussul, and T. N. Baidyk, "Sparse binary distributed encoding of numeric vectors," Journal of Automation and Information Sciences, vol. 37, pp. 47–61, 11 2005. DOI: https://doi.org/10.1615/JAutomatInfScien.v37.i11.60. [Online]. Available: https://www.researchgate.net/publication/245406876.

[51] L. Smets, W. V. Leekwijck, I. J. Tsang, and S. Latre, "Training a hyperdimensional computing classifier using a threshold on its confidence," Neural Computation, vol. 35, pp. 2006–2023, 12 May 2023. DOI: https://doi.org/10.1162/neco_a_01618. [Online]. Available: http://arxiv.org/abs/2305.19007.

[52] M. Nazemi, A. Esmaili, A. Fayyazi, and M. Pedram, "Synergiclearning: Neural network-based feature extraction for highly-accurate hyperdimensional learning," in IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD, vol. 2020-November, IEEE, Nov. 2020, pp. 1–9. DOI: https://doi.org/10.1145/3400302.3415696.

[53] M. Imani, X. Yin, J. Messerly, et al., "Searchd: A memory-centric hyperdimensional computing with stochastic training," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, pp. 2422–2433, 10 Oct. 2020, ISSN: 19374151. DOI: https://doi.org/10.1109/TCAD.2019.2952544.

[54] Y.-R. Hsiao, Y.-C. Chuang, C.-Y. Chang, and A.-Y. Wu, "Hyperdimensional computing with learnable projection for user adaptation framework," in IFIP International Conference on Artificial Intelligence Applications and Innovations (AIAI), 2021, pp. 436–447. DOI: https://doi.org/10.1007/978-3-030-79150-6_62. [Online]. Available: https://link.springer.com/10.1007/978-3-030-79150-6_62.

[55] C. T. Huang, C. Y. Chang, Y. C. Chuang, and A. Y. A. Wu, "Pq-hdc: Projection-based quantization scheme for flexible and efficient hyperdimensional computing," in IFIP Advances in Information and Communication Technology, vol. 627, Springer Science and Business Media Deutschland GmbH, 2021, pp. 425–435, ISBN: 9783030791490. DOI: https://doi.org/10.1007/978-3-030-79150-6_34.

[56] M. Heddes, I. Nunes, T. Givargis, A. Nicolau, and A. Veidenbaum, "Hyperdimensional hashing: A robust and efficient dynamic hash table," in Proceedings - Design Automation Conference, IEEE, Jul. 2022, pp. 907–912, ISBN: 9781450391429. DOI: https://doi.org/10.1145/3489517.3530553.

[57] E. Kussul, T. Baidyk, L. Kasatkina, and V. Lukovich, "Rosenblatt perceptrons for handwritten digit recognition," in International Joint Conference on Neural Networks, 2001, pp. 1516–1520. DOI: https://doi.org/10.1109/IJCNN.2001.939589.

[58] E. Kussul and T. Baidyk, "Improved method of handwritten digit recognition tested on mnist database," Image and Vision Computing, vol. 22, pp. 971–981, 12 Oct. 2004, ISSN: 02628856. DOI: https://doi.org/10.1016/j.imavis.2004.03.008.

[59] P. Vergés, T. Givargis, and A. Nicolau, "Refinehd: Accurate and efficient single-pass adaptive learning using hyperdimensional computing," in IEEE International Conference on Rebooting Computing (ICRC), 2023, pp. 1–8. DOI: https://doi.org/10.1109/ICRC60800.2023.10386671.

[60] D. Kleyko, M. Kheffache, E. P. Frady, U. Wiklund, and E. Osipov, "Density encoding enables resource-efficient randomly connected neural networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, pp. 3777–3783, 8 2021. DOI: https://doi.org/10.1109/TNNLS.2020.3015971. [Online]. Available: http://arxiv.org/abs/1909.09153http://dx.doi.org/10.1109/TNNLS.2020.3015971.

[61] E. P. Frady, D. Kleyko, and F. T. Sommer, "Variable binding for sparse distributed representations: Theory and applications," IEEE Transactions on Neural Networks and Learning Systems, vol. 34, pp. 2191–2204, 5 May 2023, ISSN: 21622388. DOI: https://doi.org/10.1109/TNNLS.2021.3105949.