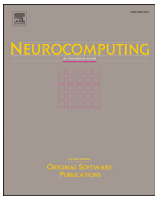
Neurocomputing 657 (2025) 131643

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Explaining and interpreting hyperdimensional computing classifiers on tabular data

Laura Smets [a,*], Werner Van Leekwijck [b], Steven Latré [a], José Oramas [a]

[a] IDLab, University of Antwerp - imec, Department of Computer Science, Sint-Pietersvliet 7, Antwerp, 2000, Belgium
[b] imec, Kapeldreef 75, Leuven, 3000, Belgium

HIGHLIGHTS

- An explanation and an interpretation method are proposed for HDC on tabular data.
- The methods are fast as they employ HDC's efficient arithmetic vector operations.
- The methods are faithful, validated with coherence checks and ablation studies.
- The deletion and insertion metrics are adjusted to apply to the HDC classifier.

ARTICLE INFO

Communicated by D. Wu

Keywords:
Hyperdimensional computing
Vector symbolic architectures
Model explanation
Model interpretation
Classification
Tabular data

ABSTRACT

Given the rise in the usage of artificial intelligence models and machine learning approaches in our day-to-day lives, it has become increasingly important to explain these models to increase user trust. Hyperdimensional Computing (HDC) has been introduced as a powerful, energy-efficient algorithmic framework that is intrinsically less opaque than (deep) neural networks. Nevertheless, the possibility of explaining and interpreting the HDC-based classification model has not yet been explored explicitly. Therefore, this work proposes an explanation method and an interpretation method for the HDC-based classification model working with tabular data. The proposed methods have been successfully evaluated on three tabular data sets with a diverse number of samples, features, and classes. Their faithfulness is validated with coherence checks, the deletion and insertion metrics, and a feature ablation study. The results of the proposed explanation method align well with the well-studied LIME explanations.

1. Introduction

Artificial intelligence (AI) and machine learning (ML) models are increasingly being used in our everyday lives. One of the main obstacles when it comes to trusting the model is (the lack of) explainability and interpretability of these models' predictions, especially in critical applications such as medicine or autonomous driving. Therefore, a large body of research has been dedicated to designing explanation methods for mostly (deep convolutional) neural networks (DCNNs) which include feature attribution methods [1–6] and counterfactual explanation methods [7]. These methods often rely on backpropagation, gradients, and input perturbation where the prediction of an input is compared to the one of the input perturbed by masking out several subsets of features.

Hyperdimensional Computing (HDC) [8], also known as Vector Symbolic Architectures (VSA) [9], has been introduced as a powerful algorithmic framework that originated at the intersection of symbolic AI and connectionism, suitable for low-resource ML tasks. HDC transforms input data into a hyperdimensional space, more specifically into hyperdimensional vectors, where it uses efficient, simple component-wise operations. It has already been used in several classification tasks, such as text classification [10–12], speech recognition [13], human activity recognition [14], hand gesture recognition [12,15,16], time series classification [17,18], classification of medical images [19,20], technical diagnostics [21,22], character recognition [23–26], robotics [27], and others [28–31].

* Corresponding author.
Email addresses: Laura.Smets@uantwerpen.be (L. Smets), Werner.VanLeekwijck@imec.be (W. Van Leekwijck), Steven.Latre@uantwerpen.be (S. Latré), Jose.Oramas@uantwerpen.be (J. Oramas).

https://doi.org/10.1016/j.neucom.2025.131643
Received 5 May 2025; Received in revised form 17 July 2025; Accepted 20 September 2025
Available online 24 September 2025

Several researchers have reported that HDC is—compared to the previously mentioned DCNNs—intrinsically less opaque because of its simple, reversible operations [32–35]. This is highlighted in several studies examining logical reasoning with HDC [8,30,36–39]. However, it is an open problem how an HDC-based classification model for tabular data can be explained on a local instance level and interpreted on a global model level [3]. Therefore, this article aims to propose a generic posthoc analysis using these simple, reversible operations to determine the individual features' contributions in the HDC-based classification model with the following contributions:

1. **An explanation and an interpretation method are proposed** for an HDC-based classifier trained with tabular data; the proposed methods are fast as they employ the efficient arithmetic vector operations of HDC;
2. **The proposed explanation and interpretation methods are shown to be faithful**, validated by employing coherence checks, the deletion and insertion metrics, and a feature ablation study;
3. **The deletion and insertion metrics are adjusted** to be applicable to the HDC-based classification model;
4. **The proposed explanation method is shown to be aligned** with the well-studied LIME explanation method.

The remaining part of this article proceeds as follows: Section 2 introduces and describes Hyperdimensional Computing. The third section provides an overview of the related work. This is followed by the description of the proposed framework for model explanation and interpretation in Section 4 with the performed experiments and the obtained results in Section 5. Section 6 concludes this work.

2. Hyperdimensional computing

2.1. HD vectors

In Hyperdimensional Computing (HDC), input data are transformed into high-dimensional (HD) vectors (also called hypervectors (HVs)) of dimension D, e.g., $D = 10,000$. Different HDC framework variants use their own types of HVs, with either binary [40–43], bipolar [44], real-valued [45], or complex-valued [46] vector components.

As the majority of applications use HDC with dense binary vectors, i.e., vector components are either 0 or 1, and about half of the components are 1, this article will focus on this type of HDC. Its atomic vectors are defined as randomly initialized dense binary HVs:

Property 1 (Dense binary HV). *Each vector component of a randomly initialized dense binary HV $\boldsymbol{v}$ follows a Bernoulli distribution with probability $\frac{1}{2}$:*

$$\mathcal{P}(\boldsymbol{v}[k] = 0) = \mathcal{P}(\boldsymbol{v}[k] = 1) = \frac{1}{2}$$

with $\boldsymbol{v}[k]$ the kth component of $\boldsymbol{v}$.[1]

2.2. HD operations

HDC employs three simple HD arithmetic operations for computations on tabular data in the HD space: (1) similarity, to measure the degree of likeness between two vectors; (2) superposition, to combine and superimpose two or more vectors; and (3) binding, to associate vectors with one another.

2.2.1. Similarity

In binary HDC, the similarity $s(\mathbf{a}, \mathbf{b})$ between two HVs $\mathbf{a}$ and $\mathbf{b}$ is calculated using the normalized Hamming distance:

$$s(\mathbf{a}, \mathbf{b}) = 1 - \frac{H(\mathbf{a}, \mathbf{b})}{D} = 1 - \frac{|\mathbf{a} \oplus \mathbf{b}|}{D} \tag{1}$$

with D the dimension of the HVs and $\oplus$ the component-wise exclusive disjunction (XOR)[2] operation. A similarity of 1 between two vectors implies having two identical vectors, while two opposite vectors will have a similarity of 0. The similarity between two vectors is approximately 0.5 in the case of pseudo-orthogonal vectors. In the HD space, two randomly initialized dense binary HVs (Property 1) are pseudo-orthogonal to each other, and converge to exact orthogonality with increasing HV dimension [47]:

Property 2 (Pseudo-orthogonality). *Two randomly initialized HVs $\boldsymbol{a}$ and $\boldsymbol{b}$ are pseudo-orthogonal to each other, i.e., the similarity is approximately equal to 0.5:*

$$s(\boldsymbol{a}, \boldsymbol{b}) \approx \frac{1}{2}$$

2.2.2. Superposition

Superposition + is performed as the component-wise addition of HVs in binary HDC to obtain a compositional $\mathbf{X}$, e.g., $\mathbf{X} = \mathbf{a}+\mathbf{b}+\mathbf{c}$, after which the compositional $\mathbf{X}$ is binarized into the HV $\mathbf{x}$ using the majority rule $[\,]$ according to:

$$\mathbf{x}[k] = [\mathbf{X}[k]] = \begin{cases} 1 & \text{if} \quad \mathbf{X}[k] > \frac{n}{2} \\ 0 & \text{if} \quad \mathbf{X}[k] < \frac{n}{2} \\ Bernoulli\left(\frac{1}{2}\right) & \text{if} \quad \mathbf{X}[k] = \frac{n}{2} \end{cases} \tag{2}$$

with n being the number of HVs in the compositional $\mathbf{X}$ and $Bernoulli(\frac{1}{2})$ the Bernoulli distribution with probability $\frac{1}{2}$ to break ties when n is an even number. One of the properties of superposition useful for this work is Property 3.[3]

Property 3 (Unstructured similarity). *The compositional HV $\boldsymbol{x} = [\boldsymbol{a} + \boldsymbol{b} + \boldsymbol{c}]$ is similar to each of its components $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ corresponding to a similarity higher than 0.5:*

$$s(\boldsymbol{x}, \boldsymbol{a}) > 0.5, \qquad s(\boldsymbol{x}, \boldsymbol{b}) > 0.5 \qquad \textit{and} \qquad s(\boldsymbol{x}, \boldsymbol{c}) > 0.5$$

2.2.3. Binding

Binary HDC employs the component-wise XOR operation $\oplus$ for the binding/association of two vectors, e.g., $\mathbf{a} \oplus \mathbf{b}$. Some properties of binding useful for this work are Properties 4–6.[3]

Property 4 (Distributivity). *The binding operation distributes over the superposition operation:*

$$\boldsymbol{d} \oplus [\boldsymbol{a} + \boldsymbol{b} + \boldsymbol{c}] = [\boldsymbol{d} \oplus \boldsymbol{a} + \boldsymbol{d} \oplus \boldsymbol{b} + \boldsymbol{d} \oplus \boldsymbol{c}]$$

Property 5 (Invertibility). *Since the XOR operation is its own inverse, i.e., $\boldsymbol{a} \oplus \boldsymbol{a} = \boldsymbol{0}$, the binding $\boldsymbol{c} = \boldsymbol{a} \oplus \boldsymbol{b}$ is invertible:*

$$\boldsymbol{a} \oplus \boldsymbol{c} = \boldsymbol{a} \oplus \boldsymbol{a} \oplus \boldsymbol{b} = \boldsymbol{b}$$

Property 6 (Structured similarity). *The binding operation preserves structured similarity, i.e., given that $\boldsymbol{a}$ and $\boldsymbol{b}$ are two random HVs, and $\boldsymbol{a}$ is similar to $\boldsymbol{a}$' and $\boldsymbol{b}$ is similar to $\boldsymbol{b}$', then $\boldsymbol{d} = \boldsymbol{a} \oplus \boldsymbol{b}$ is similar to $\boldsymbol{d}' = \boldsymbol{a}' \oplus \boldsymbol{b}'$ [47]. Considering the special case of $\boldsymbol{b} = \boldsymbol{b}'$, the similarity of $\boldsymbol{d}$ to $\boldsymbol{d}'$ is equal to the similarity between $\boldsymbol{a}$ and $\boldsymbol{a}'$ [47]:*

$$s(\boldsymbol{d}, \boldsymbol{d}') = s(\boldsymbol{a} \oplus \boldsymbol{b}, \boldsymbol{a}' \oplus \boldsymbol{b}) = s(\boldsymbol{a}, \boldsymbol{a}')$$

[1] A list of symbols used in this article is provided in Appendix A (Table A.8).

[2] The XOR operation results in a value of 1 if exactly one of its operands is 1.

[3] For a full list of properties, refer to [48].

2.3. Data transformation to hypervectors

The transformation of a sample having d features $x_i = [x_{i,1}, x_{i,2}, \dots, x_{i,d}]$, with $x_{i,j}$ being the real value of the jth feature of the ith sample, to an HV $\mathbf{v}_{x_i} \in \{0,1\}^D$ can be divided into three steps: representation of feature IDs, representation of feature values and representation of feature vectors.

2.3.1. Representation of feature IDs

All d features are represented by feature ID HVs that are randomly initialized, resulting in HVs that are pseudo-orthogonal to each other (Property 2), denoted as

$$\mathbf{v}_{ID_j} \quad \text{for} \quad j = 1 \dots d \tag{3}$$

2.3.2. Representation of feature values

Continuous feature values are ideally represented by HVs that preserve similarity. For this, the feature value range $[l_j, u_j]$ is quantized into Q quantization levels following the quantization function:

$$\phi(x_{i,j}) = \left\lceil \frac{x_{i,j}}{u_j - l_j} \cdot (Q-1) + \frac{1}{2} \right\rceil \quad \text{for} \quad i = 1 \dots N \quad \text{and} \quad j = 1 \dots d \tag{4}$$

with $\lceil\,\rceil$ the ceiling function. The Q quantization levels will be represented by the HVs

$$\mathbf{v}_q \quad \text{for} \quad q = 1 \dots Q \tag{5}$$

These HVs are created by first initializing a random HV to represent the lowest quantization level $\mathbf{v}_1$ after which an HV for each subsequent quantization level is obtained by flipping $\frac{D/2}{Q-1}$ random components from the HV of the previous quantization level where each flipped component cannot be flipped back. As such, the HVs of the lowest and highest quantization levels, i.e., $\mathbf{v}_1$ and $\mathbf{v}_Q$, will be $\frac{D}{2}$ components apart and thus pseudo-orthogonal.

2.3.3. Representation of feature vectors

An HV for the entire feature vector x_i is obtained by combining the association of each feature ID HV $\mathbf{v}_{ID_j}$ (Eq. 3) with the corresponding quantized feature value HV $\mathbf{v}_{\phi(x_{i,j})}$ (Eq. 5):

$$\begin{aligned}\mathbf{v}_{x_i} &= [\mathbf{v}_{ID_1} \oplus \mathbf{v}_{\phi(x_{i,1})} + \mathbf{v}_{ID_2} \oplus \mathbf{v}_{\phi(x_{i,2})} + \dots + \mathbf{v}_{ID_d} \oplus \mathbf{v}_{\phi(x_{i,d})}] \\ &= \left[\sum_{j=1}^{d} (\mathbf{v}_{ID_j} \oplus \mathbf{v}_{\phi(x_{i,j})})\right] \quad \text{for} \quad i = 1 \dots N\end{aligned} \tag{6}$$

with $[\,]$ the binarization function, i.e., majority rule (Eq. 2) and $\phi(x_{i,j})$ the quantized feature value of the ith sample and jth feature (Eq. 4). This type of encoding is often referred to as record-based, value-ID, key-value, compositional, etc., encoding [49–51].

2.4. HD classifier

A class centroid or prototype is created for each class in the data set. More specifically, a class bundle is obtained by combining all training sample HVs that belong to that class:

$$\mathbf{C}_l = \sum_{i=1}^{N_{tr}} \{\mathbf{v}_{x_i} \mid y_i = l\} \quad \text{for} \quad l = 1 \dots C \tag{7}$$

with N_{tr} the number of training samples, C the number of classes and y_i the ith sample's class. This class bundle is binarized with the majority rule (Eq. 2) into a class prototype $\mathbf{c}_l$. For the classification of a sample, the similarity between the considered sample's HV $\mathbf{v}_{x_i}$ and each class prototype $\mathbf{c}_l$ is computed (Eq. 1). The sample will then be classified to the class with the highest similarity.

An iterative training procedure is performed to improve the class prototypes by using wrongly classified training samples. Namely, a training sample that has been misclassified will be subtracted from the class bundle of the wrong class and added again to the class bundle of the correct class:

$$\forall \mathbf{v}_{x_i} \in \{\mathbf{v}_{x_i} \mid i = 1 \dots N_{tr}, \hat{y}_i \neq y_i\} : \mathbf{C}_{y_i} = \mathbf{C}_{y_i} + \mathbf{v}_{x_i} \quad \wedge \quad \mathbf{C}_{\hat{y}_i} = \mathbf{C}_{\hat{y}_i} - \mathbf{v}_{x_i} \tag{8}$$

with $\hat{y}_i = \arg\max_{l=1\dots C} s(\mathbf{v}_{x_i}, \mathbf{c}_l)$. Optionally, an extended procedure can be used, in which besides the wrongly classified samples, correctly classified training samples with a confidence $c(\mathbf{v}_{x_i})$[4] lower than a predefined threshold α are also used to update the class bundles, as defined by [52]:

$$\begin{aligned}\forall \mathbf{v}_{x_i} \in \{\mathbf{v}_{x_i} \mid i = 1 \dots N_{tr}, \hat{y}_i = y_i \wedge c(\mathbf{v}_{x_i}) < \alpha\} : \mathbf{C}_{y_i} = \mathbf{C}_{y_i} + \mathbf{v}_{x_i} \\ \wedge \quad \mathbf{C}_{y_i'} = \mathbf{C}_{y_i'} - \mathbf{v}_{x_i}\end{aligned} \tag{9}$$

with y_i' the class with the second highest similarity to the sample. When all training samples are passed, the updated class bundles are binarized again to obtain the updated class prototypes. This iterative training procedure can be performed for a predefined number of iterations or until a predefined accuracy on the training set or a validation set is reached.

3. Related work

Two major types of explanation methods exist: feature attribution methods that identify the importance of each feature, pixel, or image region, and counterfactual explanation methods.

In the former category, there are two well-studied explanation methods applicable to tabular data. The first is LIME, i.e., Local Interpretable Model-agnostic Explanations [5]. This method randomly creates samples around the instance to be explained, which are weighted by their distance to the considered instance, and fits an approximate linear decision model to obtain importance values for all features. SHAP, i.e., SHapley Additive exPlanations [2], is the second explanation method for tabular data based on the Shapley regression values. Each feature is assigned an importance value representing the effect on the model prediction when including the considered feature, while considering feature dependencies.

The second category of methods produces counterfactual explanations. These reveal which and how features of an instance should be changed to obtain a different outcome.

Although the explanation of HDC classification outcomes has not yet been explored explicitly in the literature, several studies have examined logical reasoning with HDC which can be seen as the foundation for explaining HDC outcomes [8,30,36–39]. These studies employ the disassociation operation and the clean-up procedure of the HDC framework, which are also used in our proposed explanation and interpretation methods.

The studies of [33,53] are closest to this article. They identify the most meaningful electrodes of the EEG signal for the binary classification of error-related potentials with HDC by measuring the similarity between the two class prototypes created with the data from only one electrode at a time. The smaller this similarity, the better the electrode discriminates between the two classes. For this, new class prototypes are created, whereas this article proposes explanation and interpretation methods that rely solely on the already constructed class prototypes.

The proposed explanation and interpretation methods can be classified as feature attribution methods, and the explanation method will be compared to the LIME explanation method. The methodology of SHAP and input modification methods will be used as coherence checks to

[4] The confidence is defined as the difference between the similarity of the sample vector to the class vector with the highest similarity and the similarity to the class vector with the second highest similarity, i.e., $c(\mathbf{v}_{x_i}) = s(\mathbf{v}_{x_i}, \mathbf{c}_{\hat{y}_i}) - max_{l \neq \hat{y}_i} s(\mathbf{v}_{x_i}, \mathbf{c}_l)$ [52].

identify the faithfulness of the proposed methods. Feature importance values are obtained by moving backward through the model with the disassociation operation which can be seen as similar to deconvolution methods that move backward through the network with gradients and/or backpropagation.

4. Proposed framework

The proposed framework first decodes the class prototypes from the HD space to the original input feature space. The explanation method (Fig. 1(a)) compares these decoded class prototypes to the sample to-be-explained to determine each feature's contribution to the prediction. In the interpretation method (Fig. 1(b)), the decoded class prototypes are compared to each other to rank the features based on their discriminative power.

4.1. Decoding the class prototypes

To decode the class prototypes into the original input feature space, we write the binarized class prototypes in a similar way as the samples (Eq. 6) such that:

$$\mathbf{c}_l = [\mathbf{v}_{ID_1} \oplus \mathbf{v}_{\phi(x_{c_l,1})} + \mathbf{v}_{ID_2} \oplus \mathbf{v}_{\phi(x_{c_l,2})} + \cdots + \mathbf{v}_{ID_d} \oplus \mathbf{v}_{\phi(x_{c_l,d})}] \quad \text{for} \quad l = 1 \dots C \tag{10}$$

Each feature ID HV $\mathbf{v}_{ID_j}$ is dissociated from the class prototype $\mathbf{c}_l$ one at a time:

$$\begin{aligned}
\mathbf{v}_{ID_j} \oplus \mathbf{c}_l &= \mathbf{v}_{ID_j} \oplus [\mathbf{v}_{ID_1} \oplus \mathbf{v}_{\phi(x_{c_l,1})} + \cdots + \mathbf{v}_{ID_j} \oplus \mathbf{v}_{\phi(x_{c_l,j})} + \cdots + \mathbf{v}_{ID_d} \\
&\quad \oplus \mathbf{v}_{\phi(x_{c_l,d})}] \\
&= [\mathbf{v}_{ID_j} \oplus \mathbf{v}_{ID_1} \oplus \mathbf{v}_{\phi(x_{c_l,1})} + \cdots + \mathbf{v}_{ID_j} \oplus \mathbf{v}_{ID_j} \oplus \mathbf{v}_{\phi(x_{c_l,j})} + \cdots \\
&\quad + \mathbf{v}_{ID_j} \oplus \mathbf{v}_{ID_d} \oplus \mathbf{v}_{\phi(x_{c_l,d})}] \qquad \text{(Property 4)} \\
&= [\mathbf{v}_{ID_j} \oplus \mathbf{v}_{ID_1} \oplus \mathbf{v}_{\phi(x_{c_l,1})} + \cdots + \mathbf{v}_{\phi(x_{c_l,j})} + \cdots + \mathbf{v}_{ID_j} \oplus \mathbf{v}_{ID_d} \\
&\quad \oplus \mathbf{v}_{\phi(x_{c_l,d})}] \qquad \text{(Property 5)} \\
&= \left[\mathbf{v}_{\phi(x_{c_l,j})} + \sum_{f=1}^{d} \{\mathbf{v}_{ID_j} \oplus \mathbf{v}_{ID_f} \oplus \mathbf{v}_{\phi(x_{c_l,f})} \mid f \neq j\}\right] \\
&\quad \text{for} \quad j = 1 \dots d \quad \text{and} \quad l = 1 \dots C
\end{aligned} \tag{11}$$

The similarity of the disassociated class prototype (Eq. 11) to the stored HVs of all Q quantization levels (Eq. 5) is computed obtaining a vector of Q similarity values:

$$s_{j,l} = [s(\mathbf{v}_{ID_j} \oplus \mathbf{c}_l, \mathbf{v}_1), \dots, s(\mathbf{v}_{ID_j} \oplus \mathbf{c}_l, \mathbf{v}_Q)] \quad \text{for} \quad j = 1 \dots d \quad \text{and} \quad l = 1 \dots C \tag{12}$$

Following the so-called clean-up procedure [47], the quantization level with the highest similarity to the dissociated class prototype can be selected as the quantized feature value of the class prototype in the original space:

$$\hat{q}_{j,l} = \phi(x_{c_l,j}) = \arg\max_q s(\mathbf{v}_{ID_j} \oplus \mathbf{c}_l, \mathbf{v}_q) \quad \text{for} \quad j = 1 \dots d, \quad l = 1 \dots C \quad \text{and} \quad q = 1 \dots Q \tag{13}$$

4.2. Model explanation

Explaining a model includes determining which features contributed the most to the classification of one particular sample and which features contributed the least, i.e., local instance-level explanation [3]. For one feature at a time, the similarity between the test sample's HV $\mathbf{v}_{x_i}$ disassociated with the corresponding feature ID HV $\mathbf{v}_{ID_j}$, and all stored quantization level HVs (Eq. 5) is calculated as in Eq. (12):

$$s_{j,i} = [s(\mathbf{v}_{ID_j} \oplus \mathbf{v}_{x_i}, \mathbf{v}_1), \dots, s(\mathbf{v}_{ID_j} \oplus \mathbf{v}_{x_i}, \mathbf{v}_Q)] \quad \text{for} \quad j = 1 \dots d \quad \text{and} \quad i = 1 \dots N_{ts} \tag{14}$$

The Spearman rank correlation coefficient is computed between the test sample's similarity vector (Eq. 14) and each class' similarity vector (Eq. 12) resulting in C correlation coefficients per feature. An important feature for the classification of that sample can then be defined as one that has a larger correlation between the sample's similarity vector and the similarity vector of the predicted class than the correlation between the sample's similarity vector and the similarity vector of all the other classes. This is equivalent to maximizing the minimum difference in correlation between the similarity of the sample to the predicted class and the similarity of the sample to the other classes. As such, the contribution of a feature j to the classification of a test sample i can be quantified as:

$$\triangle r_{j,i} = \min\{r(s_{j,i}, s_{j,\hat{y}_i}) - r(s_{j,i}, s_{j,l}) \mid l = 1 \dots C \neq \hat{y}_i\} \quad \text{for} \quad j = 1 \dots d \quad \text{and} \quad i = 1 \dots N_{ts} \tag{15}$$

with $r(\cdot,\cdot)$ being the Spearman rank correlation coefficient.

4.3. Model interpretation

The interpretation or global model-level explanation of a model, i.e., understanding what the model has learned [3], is achieved by determining which features are the most discriminative for the classification task. For this, the pairwise linear correlation between the C class similarity vectors (Eq. 12) is computed using Spearman rank correlation coefficient for each feature separately, resulting in $0.5 \cdot C \cdot (C-1)$ correlation coefficients per feature. These correlation coefficients are averaged across all class pairs for each feature, and should ideally be as small as possible indicating that the class prototypes have different feature values for the considered feature. Hence, the features can be ranked based on this average correlation coefficient:

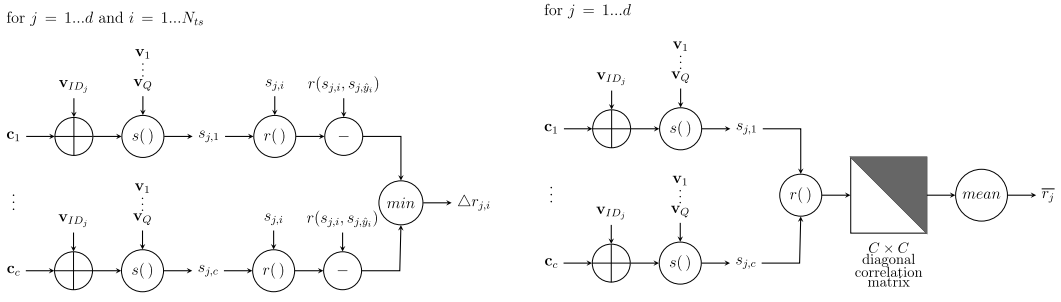


Fig. 1. Proposed framework. An overview of the proposed explanation (a, Section 4.2) and interpretation (b, Section 4.3) method. Both methods start with decoding the class prototypes (Section 4.1).

$$\overline{r_j} = \frac{1}{0.5 \cdot C \cdot (C-1)} \sum_{l=1}^{C} \sum_{k=l+1}^{C} r(s_{j,l}, s_{j,k}) \quad \text{for} \quad j = 1 \dots d \tag{16}$$

with $r(\cdot,\cdot)$ being the Spearman rank correlation coefficient.

5. Experimental evaluation

5.1. Experimental setup

An HDC classification model is trained and tested on three data sets with a varying number of samples, classes, and features:

- The cardiotocography (CTG) data set [54] includes 2126 fetal CTGs, i.e., 1701 training and 425 test samples that are classified according to their fetal state into three categories: normal (N), suspect (S), and pathologic (P) using 21 features.
- The UCIHAR data set [54,55] is obtained by recording the acceleration and velocity with an accelerometer and gyroscope of a smartphone attached to the chest of 30 subjects who performed six activities, i.e., classes, during the recording. The data set includes 7352 training and 2947 test samples, and 561 features.
- The ISOLET data set [54] is obtained by extracting 617 features from speech signals collected from 150 subjects speaking each letter of the alphabet twice. There are 6238 training and 1559 test samples, and 26 classes.

All feature values are scaled to a range of values $[0, 100]$ by obtaining scaling parameters, i.e., the minimum and maximum, for each feature using the training set and applying them to both the training and the test set. Afterward, the feature values are quantized using Eq. (4) with $l_j = 0$, $u_j = 100$ and $Q = 21$. The feature vectors are then transformed to HVs with dimension $D = 10,000$ following Eq. (6).

The confidence threshold α is set to 4.0, 0.75, and 1.0 for the CTG, UCIHAR, and ISOLET data set, respectively [52]. The iterative training procedure is performed for 100 iterations, after which the classifier with the highest classification accuracy on the training set is selected. For this best model, the class prototypes are decoded into feature vectors (Eqs. 11–13), and the feature importances are determined with the proposed explanation and interpretation methods. Since the HDC classification model relies on random initialization of HVs, all experiments are performed for five different initializations of the classification model for each data set to account for this randomness. Table 1 reports the average classification accuracy on the training and test of the three data sets.

Table 1
The classification accuracy (in %) on the training and test sets of the CTG, UCIHAR, and ISOLET data set, averaged over all five random initializations.

CTG		UCIHAR		ISOLET	
train	test	train	test	train	test
88.40 ± 0.26	74.68 ± 1.33	91.09 ± 0.78	86.40 ± 2.78	98.05 ± 1.32	90.46 ± 1.65

Data are mean ± standard deviation across five random initializations.

5.2. Model explanation validation

5.2.1. Coherence checks

Adebayo et al.[56] propose two randomization tests, i.e., the model parameter and the data randomization test, to evaluate the adequacy of explanation methods. Both randomization tests are used to validate the quality of the explanations of the proposed approach. In addition, the results of the proposed explanation method are compared to feature importances that are completely randomly assigned.

Model parameter randomization test. A model with random, untrained class prototypes is created. More specifically, the model is generated by assigning each class prototype $\mathbf{c}_l$ a random HV which is then used to classify the samples. The proposed explanation method is applied to this random classifier, after which its output is compared to the output of the explanation method applied to the original HD classifier. Ideally, the output of the explanation method differs between the two models, indicating that the proposed method depends on the learned parameters of the model. Otherwise, it can be concluded that the explanation method is insensitive to the model's parameters. A quantitative comparison between the two outputs is established with the Spearman rank correlation averaged across all test samples of each data set, which should ideally be close to zero to indicate a dependence on the learned parameters of the model [56].

Data randomization test. Akin to the model parameter randomization test, the data randomization test compares the output of the explanation method of the original classifier to the one of another classifier. In this case, the second classifier is also a trained model following Section 2.4. More specifically, the classifier is trained using a copy of the data set where the labels of the samples are randomly permuted. Again, the explanation output should ideally differ between the original classifier and this second classifier, meaning that the proposed method is sensitive to the labeling of the data. The Spearman rank correlation averaged across all test samples of each data set is computed to quantitatively compare the two outputs, which should again be as close to zero as possible [56].

Random importance test. In contrast to the model parameter and data randomization test that still constructs an HD classifier, the random importance test does not rely on a classification model. Instead, it generates the model explanation output by randomly assigning values between -1 and 1 to indicate the importance of features as a minimum difference in correlation. These random importances are again compared to the output of the explanation method applied to the original classifier with the Spearman rank correlation averaged across all test samples of each data set.

Results. Fig. 2 shows the decoded class prototypes for the original classifier and the randomization tests for CTG and one random

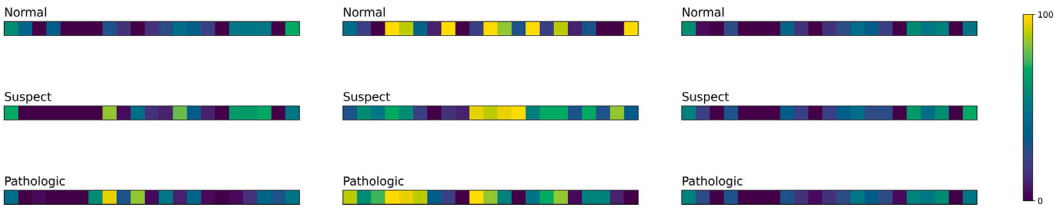


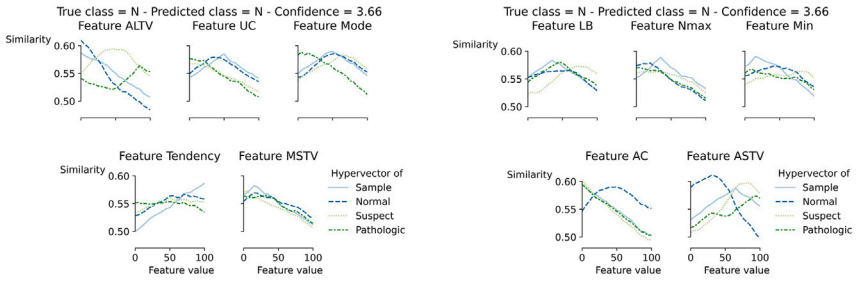
Fig. 2. Decoding the class prototypes. The class prototypes of the considered classifiers decoded to their quantized feature vector in the original space, i.e., $[0, 100]^{21}$, following Eq. (13) for the CTG data set and one random initialization of each model.

initialization of the model. It is clear that the class prototypes of the model parameter randomization test have random values for each of the 21 features, while all three class prototypes of the data randomization test appear to be quite similar to each other and to the prototype of the *Normal* class of the original classifier. The latter may be because the CTG data set is highly imbalanced, i.e., 77.37 % of the training set belongs to the *Normal* class. Therefore, when the labels are permuted in the data randomization test, all three class prototypes will be constructed with and dominated by the many training samples of the *Normal* class.

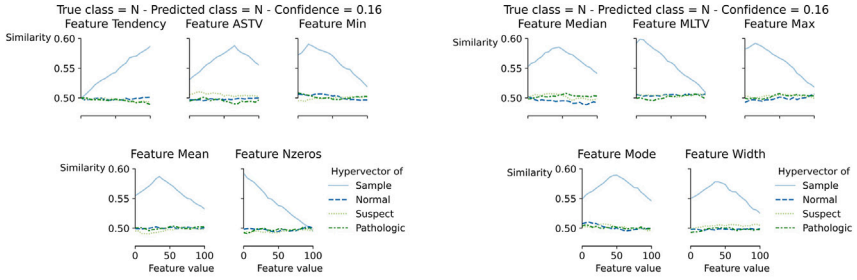
The importance of each feature and each test sample is computed for the original classification model and the coherence check models for the three data sets and the five different initializations (Figs. B.5 and B.6 in Appendix B.1). Fig. 3(A) and (B) show the similarity between each class prototype and the stored HVs of all quantization levels (Eq. 12), and between one test sample HV of the CTG data set and the stored HVs of all quantization levels (Eq. 14) for the five most and five least discriminative features, respectively. The sample's similarity curve of the original classifier is closest to the similarity curve of the *Normal* class (i.e., the predicted class) for the five most discriminative features. In contrast, the five least discriminative features tend to show a sample's similarity curve that is closer to the similarity curve of the *Pathologic* class. Hence, these features did not contribute to classifying the sample to the *Normal* class. It is trivial that the sample's similarity vector is highly dissimilar from those of the random class prototypes of the model parameter randomization test. The similarity curves of the data randomization test are all very alike, which is expected since the considered test sample belongs to the *Normal* class and all class prototypes are highly represented by samples from that class. It can also be seen that the original classifier is more confident in its classification of the sample compared to the two randomization tests, which is measured as the difference in similarity between the two highest-ranked classes.

We can see from Fig. 3(A,a) that the sample has been classified into the *Normal* class since it has a low percentage of time with abnormal long term variability (*ALTV*), an average number of uterine contractions per second (*UC*), an average mode of the fetal heart rate (FHR) histogram (*Mode*), a high FHR histogram tendency (*Tendency*) and a low mean value of short term variability (*MSTV*).

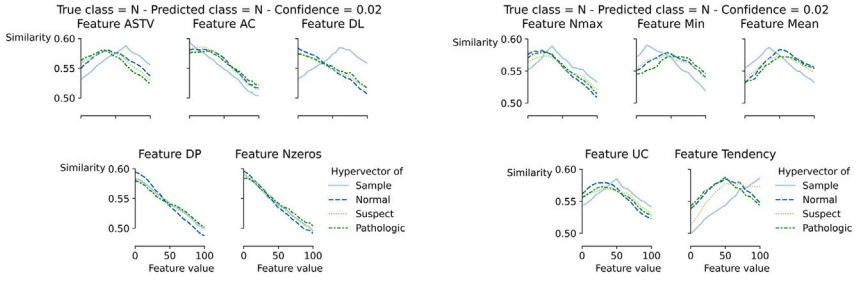
Table 2 contains the average Spearman rank correlation between the explanation output of the original classifier and the output of the coherence check models for the three data sets. The table highlights that



(a) Original classifier.



(b) Model parameter randomization test.



(c) Data randomization test.

(A) The five most discriminative features.

(B) The five least discriminative features.

Fig. 3. Model explanation. The similarity vector for the five most discriminative (A) and five least discriminative (B) features of one test sample (Eq. 14) and the three class prototypes (Eq. 12) of the CTG data set for the considered classifiers, obtained for one random initialization of the models.

Table 2
Model explanation. The average Spearman rank correlation between the explanation output of the original classifier and the output of the coherence check models and the LIME explanation method [5] for the test samples of the CTG, UCIHAR, and ISOLET data set, averaged over all five random initializations.

	CTG	UCIHAR	ISOLET
Model rand.[1]	0.024 ± 0.078	−0.022 ± 0.015	−0.018 ± 0.007
Data rand.[2]	−0.080 ± 0.082	0.077 ± 0.010	0.205 ± 0.011
Rand. importance[3]	−0.005 ± 0.007	0.000 ± 0.001	0.000 ± 0.001
LIME [5]	0.560 ± 0.037	0.159 ± 0.012	0.133 ± 0.020

Data are mean ± standard deviation across five random initializations.
[1] Model parameter randomization test.
[2] Data randomization test.
[3] Random importance test.

the correlation coefficients concentrate around zero, indicating that the explanations of the proposed method are adequate.

5.2.2. Deletion and insertion metric

Petsiuk et al.[4] propose two automatic evaluation metrics to validate explanations: deletion and insertion. For this, features are removed from or added to a test sample one by one[5] while keeping track of the model's ability to classify that test sample correctly. The removal or inclusion of features is established by replacing the feature-ID bound vector ($\mathbf{v}_{ID_j} \oplus \mathbf{v}_{\phi(x_{i,j})}$) of the considered feature(s) with a randomly generated HV in Eq. (6). Features are removed or added based on their importance predicted with the proposed explanation method with decreasing importance, i.e., the most important feature is removed or added first while investigating the effect on the classification performance. In the original article [4], the classification performance is measured through the probability of the predicted class. The higher this probability, the lower the probability of the other classes, because they sum up to one. In contrast, the similarities between the test sample's HV and the class prototypes do not sum up to one. Therefore, a relative similarity is needed as a measure of classification performance which can be defined as the confidence with which the class of the considered test sample is predicted [52]:

$$c(\mathbf{v}_{x_{i,f}}) = \begin{cases} s(\mathbf{v}_{x_{i,f}}, \mathbf{c}_{y_i}) - s(\mathbf{v}_{x_{i,f}}, \mathbf{c}_{\hat{y}_i}), & \text{if} \quad \hat{y}_i \neq y_i \\ s(\mathbf{v}_{x_{i,f}}, \mathbf{c}_{y_i}) - s(\mathbf{v}_{x_{i,f}}, \mathbf{c}_{y'_i}), & \text{if} \quad \hat{y}_i = y_i \end{cases} \quad \text{for} \quad i = 1 \dots N_{ts} \tag{17}$$

with $\mathbf{v}_{x_{i,f}}$ the sample's HV excluding f features, $\hat{y}_i = \arg\max_{l=1\dots C} s(\mathbf{v}_{x_{i,0}}, \mathbf{c}_l)$ and y'_i the class with the second highest similarity to the sample. As such, the confidence will be positive when the sample has been correctly classified, i.e., the similarity of the sample to the prototype of the correct class $s(\mathbf{v}_{x_{i,f}}, \mathbf{c}_{y_i})$ is higher than the one to the other classes, and vice versa. Hence, a higher confidence value indicates better classification performance.

This deletion and insertion procedure is performed for all test samples and five different initializations using the explanation output obtained not only for the original classifier but also for the coherence check models. As such, the deletion and insertion metric of the feature importances of the original classifier can be compared to those of the coherence check models. This study provides the first application of the deletion and insertion metric to the coherence check models.

A quantitative comparison of the different models' deletion and insertion curves is performed by calculating the AUC [4]. For this, the confidence values are min-max scaled to a range of values $[0, 1]$. A sharp drop or increase in confidence is expected at the beginning of the gradual removal or inclusion of features, respectively, for a sample initially correctly classified by the classifier before removing any of the features, i.e., the AUC should ideally be as small or as high as possible, respectively. In contrast, the deletion or insertion curve of a sample that has been wrongly classified by the initial classifier should ideally have an AUC value as large or as small as possible, respectively, corresponding to a sharp increase or drop in confidence at the beginning of the deletion or insertion procedure, respectively.

Results. The deletion and insertion curves for a correctly and a wrongly classified test sample of the CTG data set can be found in Figs. B.7 and B.8 in Appendix B.1. All average AUC values of the test samples of CTG, UCIHAR, and ISOLET for the various models are given in Table 3. The test samples are split into two groups based on whether they are correctly or wrongly classified by the original classifier. From the table it can be seen that the AUC values of the original classifier are lower (and higher) than those of the coherence checks for the correctly (and wrongly) classified samples. Consequently, the proposed explanation method can predict the feature's degree of contribution to the classification of a test sample faithfully. Please refer to Appendix B.1 (Fig. B.9) for additional results.

5.2.3. Alignment with LIME explanations

Although there are no HDC explanation methods available, Ribeiro et al. [5] proposed LIME as an explanation method suitable for any classifier. Therefore, the output of the proposed purely HDC-based explanation method is compared to the output of LIME explaining an HDC classifier where the class similarities are min-max scaled to obtain probabilities. For this, the LIME explanation method is applied to the five different initializations of the trained HDC classification model following their publicly available tutorial[6] and code.[7] More specifically, 5000 samples are sampled around the sample to be explained with Gaussian sampling, a Ridge regression model is learned on this neighborhood data, and explanations are computed for the one class with the highest probability.

Results. The obtained feature contributions (Fig. B.6(e) in Appendix B.1) are compared to those obtained with the proposed explanation method using a Spearman rank correlation and the deletion and insertion metrics. Results are included in Tables 2 and 3 and Figs. B.7–B.9. The Spearman rank correlation is larger than zero indicating an alignment of the proposed explanation method with LIME explanations. However, the correlation for UCIHAR and ISOLET is low due to the large number of features. More specifically, the chance that a feature is ranked at exactly the same importance position decreases with increasing number of features. The average AUC values of the deletion and insertion curves are slightly worse than those of the proposed explanation method. Table 4 compares the average time needed by the proposed and the LIME explanation methods to explain the outcome of a test sample of the three data sets. The table highlights that the proposed explanation method, compared to the LIME explanation method, is roughly 34 times faster for the CTG data set, 17 times faster for the UCIHAR data set, and 4 times faster for the ISOLET data set.

5.3. Model interpretation validation

5.3.1. Coherence checks

The random importance test described in Section 5.2.1 is employed to validate the output of the proposed interpretation method.

Results. The importance of each feature is computed for the classification model of the three data sets and five initializations (Fig. B.10 in Appendix B.2). Fig. 4(A) and (B) show the similarity between each class prototype and the stored HVs of all quantization levels (Eq. 12) for

[5] Because of the large number of features in the UCIHAR and ISOLET data sets, the deletion/insertion of features is done in groups of 50 features.

[6] https://marcotcr.github.io/lime/tutorials/Tutorial-continuousandcategoricalfeatures.html.
[7] https://github.com/marcotcr/lime/blob/master/lime/lime_tabular.py.

Table 3
Deletion and insertion metrics. The average area under the deletion (a) and insertion (b) curves using feature importance values obtained with the different models for the test samples split into two groups (i.e., those that are correctly and wrongly classified with the original classifier) of the CTG, UCIHAR, and ISOLET data set, averaged over all five random initializations.

(a) Deletion metric.

	CTG		UCIHAR		ISOLET	
	Correct	Wrong	Correct	Wrong	Correct	Wrong
original classifier	0.313 ± 0.006	0.639 ± 0.004	0.314 ± 0.005	0.586 ± 0.018	0.322 ± 0.007	0.443 ± 0.026
model rand.[1]	0.500 ± 0.012	0.502 ± 0.011	0.546 ± 0.007	0.558 ± 0.014	0.587 ± 0.006	0.590 ± 0.011
data rand.[2]	0.445 ± 0.024	0.489 ± 0.023	0.592 ± 0.018	0.600 ± 0.060	0.627 ± 0.006	0.630 ± 0.007
rand. importance[3]	0.478 ± 0.011	0.531 ± 0.013	0.450 ± 0.007	0.509 ± 0.007	0.499 ± 0.001	0.492 ± 0.010
LIME [5]	0.432 ± 0.011	0.570 ± 0.008	0.441 ± 0.010	0.511 ± 0.003	0.493 ± 0.003	0.494 ± 0.005

(b) Insertion metric.

	CTG		UCIHAR		ISOLET	
	Correct	Wrong	Correct	Wrong	Correct	Wrong
original classifier	0.643 ± 0.016	0.423 ± 0.017	0.618 ± 0.012	0.443 ± 0.012	0.663 ± 0.006	0.554 ± 0.022
model rand.[1]	0.491 ± 0.024	0.493 ± 0.029	0.459 ± 0.010	0.451 ± 0.023	0.419 ± 0.002	0.417 ± 0.007
data rand.[2]	0.508 ± 0.015	0.499 ± 0.036	0.454 ± 0.013	0.454 ± 0.037	0.388 ± 0.004	0.380 ± 0.009
rand. importance[3]	0.478 ± 0.010	0.534 ± 0.006	0.470 ± 0.007	0.507 ± 0.003	0.520 ± 0.003	0.505 ± 0.006
LIME [5]	0.518 ± 0.012	0.504 ± 0.012	0.478 ± 0.005	0.500 ± 0.005	0.527 ± 0.003	0.504 ± 0.003

Data are mean ± standard deviation across five random initializations.
[1] Model parameter randomization test.
[2] Data randomization test.
[3] Random importance test.

Table 4
Model explanation. The average time (in sec/sample) needed to explain the outcome of one test sample of the CTG, UCIHAR, and ISOLET data set for the proposed and the LIME explanation method [5], averaged over all five random initializations.

	CTG	UCIHAR	ISOLET
Proposed	$38.74 \cdot 10^{-3} \pm 2.58 \cdot 10^{-3}$	9.30 ± 0.39	32.22 ± 0.85
LIME [5]	1.33 ± 0.07	160.24 ± 0.84	124.33 ± 3.71

Data are mean ± standard deviation across five random initializations.

Table 5
Model interpretation. The time (in sec) needed to interpret the classifier of the CTG, UCIHAR, and ISOLET data set for the proposed interpretation method, averaged over all five random initializations.

	CTG	UCIHAR	ISOLET
Proposed	0.07 ± 0.03	15.05 ± 1.20	242.34 ± 5.98

Data are mean ± standard deviation across five random initializations.

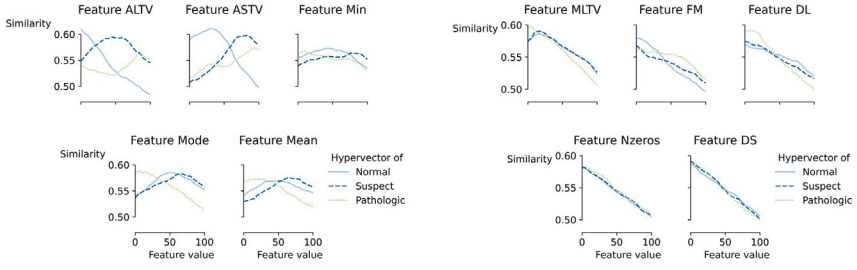
the five most and five least discriminative features, respectively, for the original classifier trained on the CTG data set for one random initialization of the model. It can immediately be seen that the similarity curves of the three classes of the original classifier are distinct for the five most discriminative features, while these are highly similar for the five least discriminative features.

The output of the proposed interpretation method reveals that the three most discriminative features (Fig. 4(A)) are the percentage of time with abnormal long term variability (*ALTV*) which is low for *Normal*, average for *Suspect* and high for *Pathologic*; the percentage of time with abnormal short term variability (*ASTV*) which is low for *Normal* and high

Table 6
Model interpretation. The Spearman rank correlation between the interpretation output of the original classifier and the output of the random importance test for the CTG, UCIHAR, and ISOLET data set, averaged over all five random initializations.

	CTG	UCIHAR	ISOLET
random importance test	−0.112 ± 0.323	0.009 ± 0.043	0.016 ± 0.027

Data are mean ± standard deviation across five random initializations.



(A) The five most discriminative features.

(B) The five least discriminative features.

Fig. 4. Model interpretation. The similarity vector for the five most discriminative (A) and five least discriminative (B) features of the three class prototypes (Eq. 12) of the CTG data set for the original classifier, obtained for one random initialization of the models.

Table 7
Feature ablation study. The area under the feature ablation curve of the training and test sets using feature importance values obtained with the different models, averaged over all five random initializations.

	CTG		UCIHAR		ISOLET	
	train	test	train	test	train	test
original classifier	0.589 ± 0.026	0.610 ± 0.024	0.499 ± 0.011	0.478 ± 0.005	0.532 ± 0.011	0.505 ± 0.010
rand. importance[1]	0.775 ± 0.011	0.739 ± 0.038	0.643 ± 0.012	0.612 ± 0.021	0.660 ± 0.005	0.623 ± 0.007

Data are mean ± standard deviation across five random initializations.
[1] Random importance test.

for *Suspect* and *Pathologic*; and the minimum of the FHR histogram (Min) which is average for *Normal*, high for *Suspect* and low for *Pathologic*.

Table 5 includes the time needed to interpret the classifier for the three data sets, highlighting a fast execution of the proposed interpretation method.

Table 6 contains the average Spearman rank correlation between the interpretation output of the original classifier and the output of the random importance test for the three data sets. All Spearman correlation values tend to be close to zero, demonstrating the quality of the proposed interpretation method.

5.3.2. Feature ablation study

The feature ablation study is highly similar to the deletion metric (Section 5.2.2) with the only difference being that the classification performance is quantified using the classification accuracy for the training and test sets. The study is performed on the original classifier and the random importance test. The classification accuracy ideally shows a sharp drop at the beginning of the feature ablation study because most important features are ablated first. This corresponds to an AUC value that is as small as possible.

Results. Fig. B.11 in Appendix B.2 shows the feature ablation curves for the different models. The AUC values of the feature ablation curves for the different models for both the training and test sets of the three data sets are given in Table 7. The AUC of the original model is lower than that of the random importance test for all data sets such that the proposed interpretation method can be considered faithful.

6. Conclusion

This work proposes an explanation and an interpretation method using hyperdimensional computing and its efficient operations. This method has been evaluated successfully on three tabular data sets with a diverse number of samples, features, and classes. Its quality is validated with randomization tests, and a feature ablation study and the deletion and insertion metric have highlighted the faithfulness of the method. The explanations obtained with the proposed method are shown to be aligned with the well-known LIME explanations. A natural progression of this work is to analyze how the operations of hyperdimensional computing can be employed to produce counterfactual explanations. Another further study could extend and/or adjust the proposed methods to be applied to other data types such as images.

CRediT authorship contribution statement

Laura Smets: Writing – original draft, Visualization, Software, Methodology, Conceptualization. **Werner Van Leekwijck:** Writing – review & editing, Methodology, Conceptualization. **Steven Latré:** Writing – review & editing, Supervision, Funding acquisition. **José Oramas:** Writing – review & editing, Methodology, Conceptualization.

Funding

This work was supported by the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" program.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Notation

A summary of notation can be found in Table A.8.

Table A.8
List of symbols (HV = hypervector).

Symbol	Definition	Symbol	Definition
x_i	sample i in feature space	d	number of features ($j = 1 \dots d$)
C	number of classes ($l = 1 \dots C$)	N	number of samples ($i = 1 \dots N$)
N_{tr}	number of training samples	N_{ts}	number of test samples
l_j	minimum value of jth feature	u_j	maximum value of jth feature
y_i	class label of ith sample	$\hat{y}_i$	predicted class of ith sample
y'_i	class with second highest similarity		
$s()$	similarity function	$H()$	Hamming distance
$Bernoulli()$	Bernoulli distribution	$[\,]$	binarization function, i.e., majority rule
$+$	component-wise addition operator	$\oplus$	component-wise XOR operator
$\phi()$	quantization function	$\lceil\,\rceil$	ceiling function
D	number of components in HVs ($k = 1 \dots D$)	Q	number of quantization levels ($q = 1 \dots Q$)
$\mathbf{v}$, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$, $\mathbf{x}$	dense binary HVs	$\mathbf{X}$	non-binary, integer HV, compositional HV
$\mathbf{v}_{x_i}$	HV of ith sample	$\mathbf{v}_{ID_j}$	HV of jth feature
$\mathbf{v}_q$	HV of qth quantization level		
$\mathbf{C}_l$	bundle of lth class	$\mathbf{c}_l$	prototype of lth class
$c(\mathbf{v}_{x_i})$	confidence of ith sample	α	confidence threshold
$s_{j,l}$	vector of similarity values for jth feature and lth class	$\hat{q}_{j,l}$	estimated quantized value of jth feature and lth class
r	Spearman rank correlation coefficient	$\overline{r_j}$	model-level importance value of jth feature
$\triangle r_{j,i}$	sample-level importance value of jth feature for ith sample		

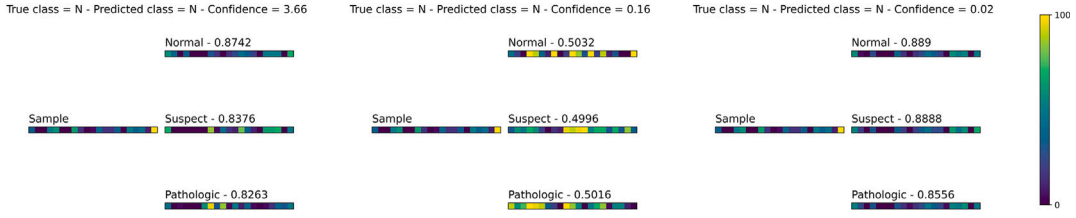
Appendix B. Additional results

B.1. Model explanation

Fig. B.5 shows a CTG test sample's feature vector and the class prototypes of the considered classifiers mapped to the original feature space where the feature vectors contain the quantized feature values determined with Eq. (13). The similarities of the sample's HV to each of the class prototypes are also mentioned. Given that the class prototypes of the model parameter randomization test are randomly generated, it is trivial that the sample's HV and each class prototype are pseudo-orthogonal, i.e., similarities are around 0.5. Since all class prototypes of the data randomization test are highly similar and the sample belongs to the *Normal* class, the similarities between each class prototype and the considered test sample's HV are also comparable.

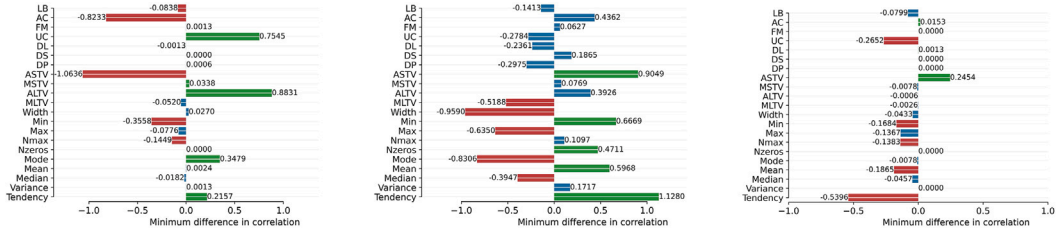
Fig. B.6 illustrates the contribution of each feature to the classification of one CTG test sample for one random initialization of the model as the minimum difference in correlation coefficient calculated with Eq. (15); the higher the minimum difference in correlation of a feature, the more the feature contributed to the classification of the considered sample. There is no resemblance between the importance values obtained with the coherence checks and those of the original model. In contrast, the output of the LIME explanation method and the proposed method on the original model roughly indicate the same features as most and least discriminative.

The deletion and insertion curves of one correctly classified test sample of the CTG data set using the model explanation output obtained with the original classifier, the coherence check models, and the LIME explanation method are shown in Fig. B.7. The figure also indicates to which class the test sample created with fewer features is classified, with the true class highlighted in green. Looking at the deletion curves, the confidence values of the original classifier decrease drastically when removing the first most important features after which the confidence values become negative. In other words, the model classifies the sample incorrectly when more than nine important features are not considered. In contrast, the deletion curves of the data randomization test and the random importance test show a more gradual decrease in confidence values. The model parameter randomization test shows a random behavior regarding confidence values and class prediction. When comparing the deletion curve of the original classifier to those of the coherence checks, it can be concluded that the former shows the most desired behavior and that the proposed explanation method can determine which

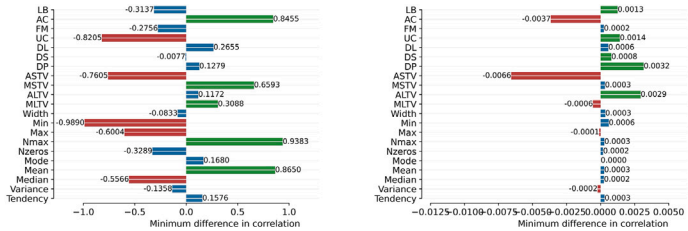


(a) Original classifier. (b) Model parameter randomization test. (c) Data randomization test.

Fig. B.5. Model explanation. A test sample's feature vector and the class prototypes of the considered classifiers decoded to their quantized feature vector in the original space, i.e., $[0, 100]^{21}$, following Eq. (13) for the CTG data set and one random initialization of each model. The similarity of the test sample HV to each of the class prototypes is also mentioned.

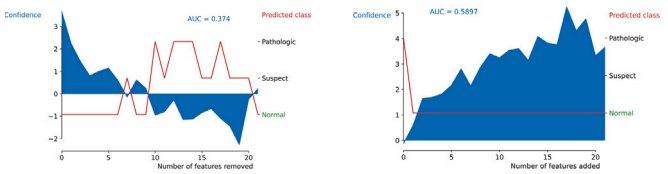


(a) Original classifier. (b) Model parameter randomization test. (c) Data randomization test.

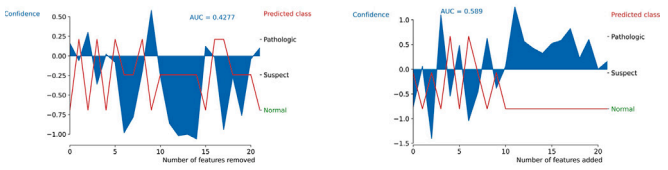


(d) Random importance test. (e) LIME explanations.

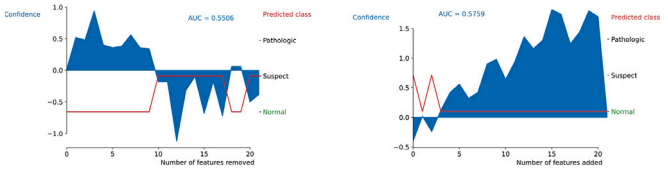
Fig. B.6. Model explanation. The minimum difference in correlation coefficients (Eq. 15) of each feature for one test sample for the considered models, obtained for one random initialization of each model for the CTG data set. The five most discriminative and the five least discriminative features are highlighted in green and red, respectively. The other features are shown in blue.



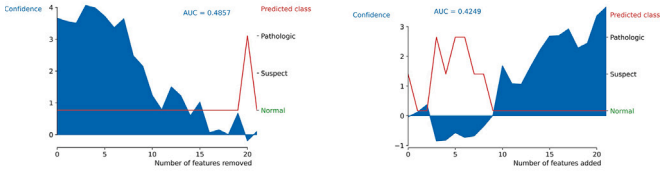
(a) Original classifier.



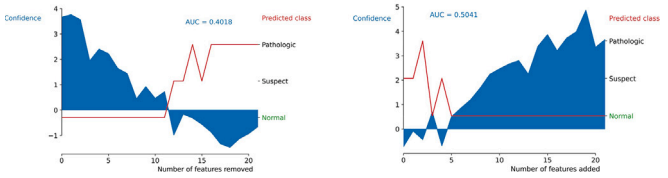
(b) Model parameter randomization test.



(c) Data randomization test.



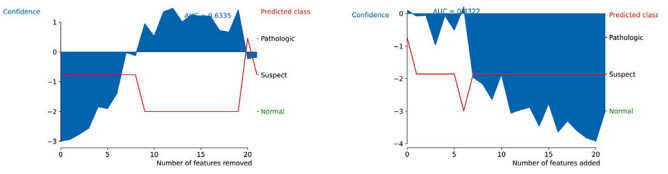
(d) Random importance test.



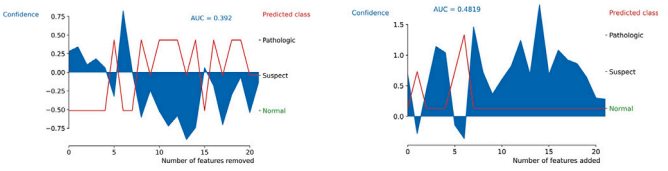
(e) LIME explanations.

(A) Deletion metric. (B) Insertion metric.

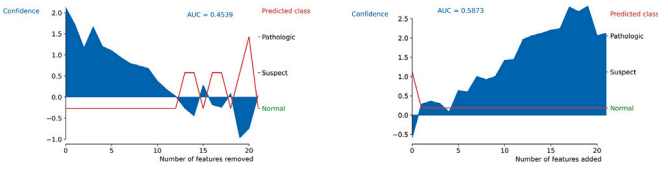
Fig. B.7. Deletion and insertion metric. Features are deleted (A) or inserted (B) with decreasing importance obtained with the different models for one correctly classified test sample of the CTG data set and one random initialization of the models. The confidence of prediction is shown in function of the number of features removed or added.



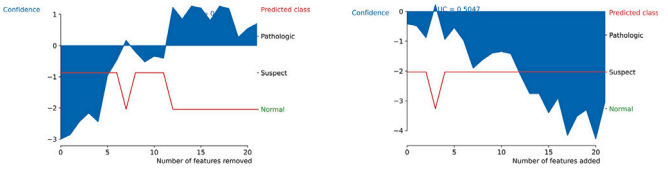
(a) Original classifier.



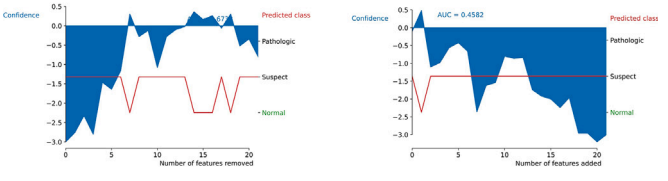
(b) Model parameter randomization test.



(c) Data randomization test.



(d) Random importance test.



(e) LIME explanations.

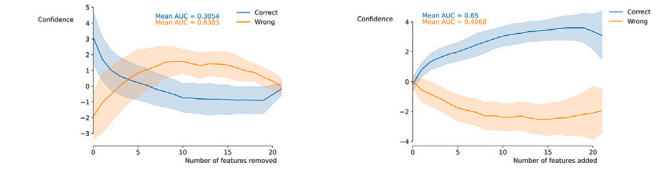
(A) Deletion metric. (B) Insertion metric.

Fig. B.8. Deletion and insertion metric. Features are deleted (A) or inserted (B) with decreasing importance obtained with the different models for one wrongly classified test sample of the CTG data set and one random initialization of the models. The confidence of prediction is shown in function of the number of features removed or added.

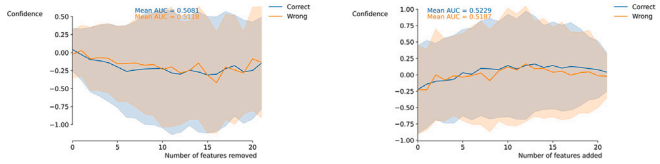
features contributed most to the classification of the considered test sample. When comparing the insertion curves of the coherence checks to the one of the original classifier, it is seen that the latter has an increase in confidence values and correctly predicts the sample after including the most important feature, while the former shows either no clear increase or a slower increase in confidence values. The deletion and insertion curves of the LIME explanations show a slower decrease and increase compared to the proposed explanation method, respectively.

The deletion and insertion curves of one wrongly classified test sample of the CTG data set using the model explanation output obtained with the original classifier, the coherence check models, and the LIME explanation method are shown in Fig. B.8. The figure also indicates to which class the test sample created with fewer features is classified, with the true class highlighted in green. Looking at the deletion curves, the confidence values of the original classifier increase drastically when removing the first most important features after which the confidence values stabilize to a positive value. In the meantime, it is seen that the sample is constantly correctly classified when more than eight important features are not considered. In contrast, the deletion curves of the random importance test show a slower increase in confidence values, and the one of LIME explanations does not stabilize to positive confidence values and fluctuates between a correct and wrong prediction. The randomization tests show either a random behavior regarding confidence values

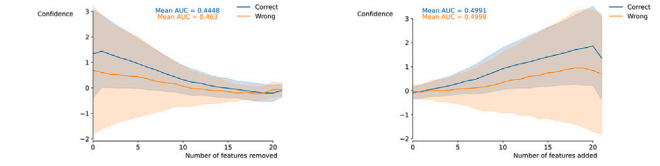
and class prediction or a decrease in confidence values. When comparing the deletion curve of the original classifier to those of the coherence check models, it can be concluded that the former shows the most desired behavior and that the proposed explanation method can determine which features contributed most to the classification of the considered test sample. When comparing the insertion curves of the randomization tests to the one of the original classifier, it is seen that the latter has the desired decrease in confidence values and never correctly predicts the sample after including the most important feature, while the former shows random behavior or even an increase in confidence values. The insertion curves of the random importance test and LIME explanations also show a decrease in confidence values, but these decreases are slightly slower than those of the original classifier.



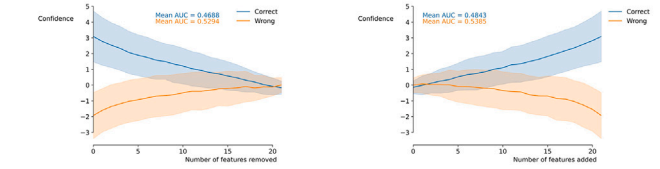
(a) Original classifier.



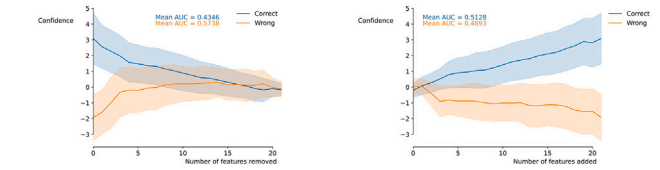
(b) Model parameter randomization test.



(c) Data randomization test.



(d) Random importance test.



(e) LIME explanations.

(A) Deletion metric. (B) Insertion metric.

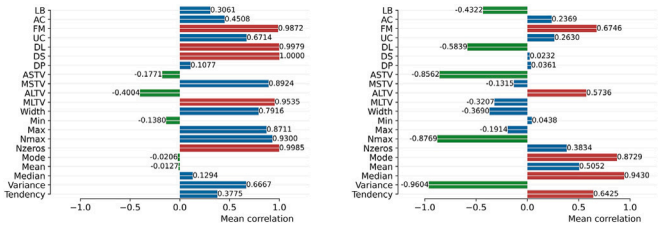
Fig. B.9. Deletion and insertion metric. The deletion (A) and insertion (B) curves averaged across the CTG test samples that are correctly or wrongly classified with the original classifier for one random initialization for the considered models. The average confidence of prediction is shown in function of the number of features removed or added.

Fig. B.9 demonstrates the deletion and insertion curves averaged across all test samples of the CTG data set split into two groups, i.e., the samples that are correctly classified with the original classifier and those that are wrongly classified, for one random initialization of the models. These average curves are obtained by taking the average confidence value across all correctly and wrongly classified test samples at each feature removal step. The average AUC values are also shown in the figure. The AUC values of the original classifier's deletion curves are lower (and higher) than those of the coherence checks for the correctly (and wrongly) classified samples. Also, the insertion curves of the original classifier show better AUC values compared to those of the coherence checks. Consequently, the proposed explanation method can predict the degree of contribution of features to the classification of a test sample faithfully.

B.2. Model interpretation

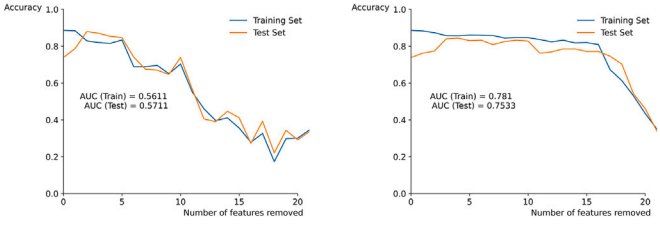
Fig. B.10 shows the discriminative power of each feature as the mean correlation coefficient (Eq. 16) for the considered models, obtained for one random initialization of each model for the CTG data set. The lower the mean correlation of a feature, the better the feature discriminates between the classes. The feature importance values of the two different models appear highly different from one another.

Fig. B.11 shows the feature ablation curves for the different models. The desired sharp drop at the beginning of the ablation is not seen for the original classifier. Instead, the decrease is rather gradual. This might be due to the property of structured similarity of the binding operation (Property 6). Namely, even if only one of the 21 features is removed, the remaining 20 feature value-ID bound pairs in the sample's HV will stay



(a) Original classifier. (b) Random importance test.

Fig. B.10. Model interpretation. The mean correlation coefficients of each feature (Eq. 16) for the considered models, obtained for one random initialization of each model for the CTG data set. The five most discriminative features and the five least discriminative features are highlighted in green and red, respectively. The other features are shown in blue.



(a) Original classifier. (b) Random importance test.

Fig. B.11. Feature ablation study. Features are ablated with decreasing importance obtained with the different models for the CTG data set and one random initialization of the models. The classification accuracy for the training and test set is shown in function of the number of features ablated.

similar to the corresponding 20 feature value-ID pairs in the class prototypes. Hence, classification is not strongly influenced. Nevertheless, the feature ablation curve of the original classifier shows more desirable behavior in appearance and the AUC value than those obtained with the random importance test. Therefore, we can conclude that the proposed interpretation method faithfully identifies the features' importance.

Data availability

The data sets are publicly available.

References

[1] R.C. Fong, A. Vedaldi, Interpretable explanations of black boxes by meaningful perturbation, in: IEEE International Conference on Computer Vision (ICCV), 2017, https://doi.org/10.1109/ICCV.2017.371

[2] S. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Conference on Neural Information Processing Systems, 2017, https://doi.org/10.48550/arXiv.1705.07874

[3] J. Oramas, K. Wang, T. Tuytelaars, Visual explanation by interpretation: improving visual feedback capabilities of deep neural networks, in: International Conference on Learning Representations (ICLR), 2019, https://doi.org/10.48550/arXiv.1712.06302

[4] V. Petsiuk, A. Das, K. Saenko, Rise: randomized input sampling for explanation of black-box models, in: British Machine Vision Conference, 2018, https://doi.org/10.48550/arXiv.1806.07421

[5] M.T. Ribeiro, S. Singh, C. Guestrin, "why should i trust you?" Explaining the predictions of any classifier, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, vol. 13-17-August-2016, Association for Computing Machinery, 2016, pp. 1135–1144, https://doi.org/10.1145/2939672.2939778

[6] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: visual explanations from deep networks via gradient-based localization, in: IEEE International Conference on Computer Vision (ICCV), 2017, https://doi.org/10.1007/s11263-019-01228-7

[7] R. Guidotti, Counterfactual explanations and how to find them: literature review and benchmarking, Data Min. Knowl. Discov. (2022) https://doi.org/10.1007/s10618-022-00831-6

[8] P. Kanerva, Hyperdimensional computing: an introduction to computing in distributed representation with high-dimensional random vectors, Cogn. Comput. 1 (2009) 139–159, https://doi.org/10.1007/s12559-009-9009-8

[9] 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, https://doi.org/10.48550/arXiv.cs/0412059

[10] I. Misuno, D. Rachkovskij, S. Slipchenko, A. Sokolov, Searching for text information with the help of vector representations, Probl. Prog. 4 (2005) 50–59.

[11] D.A. Rachkovskij, Linear classifiers based on binary distributed representations, Inf. Theor. Appl. 14 (2007) 270–274.

[12] A. Rahimi, P. Kanerva, 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, 2016, pp. 64–69, https://doi.org/10.1145/2934583.2934624

[13] M. Imani, D. Kong, A. Rahimi, T. Rosing, VoiceHD: hyperdimensional computing for efficient speech recognition, in: IEEE International Conference on Rebooting Computing (ICRC), 2017, pp. 1–8, https://doi.org/10.1109/ICRC.2017.8123650

[14] Y. Kim, M. Imani, 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, 2018, pp. 1–6, https://doi.org/10.1145/3277593.3277617

[15] A. Moin, A. Zhou, A. Rahimi, A. Menon, S. Benatti, G. Alexandrov, S. Tamakloe, J. Ting, N. Yamamoto, Y. Khan, F. Burghardt, L. Benini, A.C. Arias, J.M. Rabaey, A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition, Nat. Electron. 4 (2021) 54–63, https://doi.org/10.1038/s41928-020-00510-8

[16] A. Zhou, R. Muller, J. Rabaey, Memory-efficient, limb position-aware hand gesture recognition using hyperdimensional computing, in: TinyML Research Symposium, 2021, pp. 1–8, https://doi.org/10.48550/arXiv.2103.05267

[17] D.A. Rachkovskij, Shift-equivariant similarity-preserving hypervector representations of sequences, Cogn. Comput. 16 (2024) 909–923, https://doi.org/10.1007/s12559-024-10258-4

[18] K. Schlegel, P. Neubert, P. Protzel, HDC-MiniROCKET: explicit time encoding in time series classification with hyperdimensional computing, in: International Joint Conference on Neural Network, 2022, pp. 1–8, https://doi.org/10.48550/arXiv.2202.08055

[19] D. Kleyko, S. Khan, E. Osipov, 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, 2017, pp. 1053–1056, https://doi.org/10.1109/ISBI.2017.7950697

[20] N. Watkinson, T. Givargis, V. Joe, A. Nicolau, 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, https://doi.org/10.1109/EMBC46164.2021.9630898

[21] E.M. Kussul, L.M. Kasatkina, D.A. Rachkovskij, D.C. Wunsch, Application of random threshold neural networks for diagnostics of micro machine tool condition, 1998. https://doi.org/10.1109/IJCNN.1998.682270

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

[23] A. Goltsev, D. Rachkovskij, Combination of the assembly neural network with a perceptron for recognition of handwritten digits arranged in numeral strings, Pattern Recognit. 38 (2005) 315–322, https://doi.org/10.1016/j.patcog.2004.09.001

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

[25] D.A. Rachkovskij, Representation of spatial objects by shift-equivariant similarity-preserving hypervectors, Neural Comput. Appl. 34 (2022) 22387–22403, https://doi.org/10.1007/s00521-022-07619-1

[26] L. Smets, W.V. Leekwijck, I.J. Tsang, S. Latré, An encoding framework for binarized images using hyperdimensional computing, Front. Big Data 7 (2024) https://doi.org/10.3389/fdata.2024.1371518

[27] P. Neubert, S. Schubert, P. Protzel, An introduction to hyperdimensional computing for robotics, KI - Künstl. Intell. 33 (2019) 319–330, https://doi.org/10.1007/s13218-019-00623-z

[28] S. Aygun, M.S. Moghadam, M.H. Najafi, M. Imani, Learning from hypervectors: a survey on hypervector encoding, arXiv:2308.00685, 2023, pp. 1–14.

[29] L. Ge, K.K. Parhi, Classification using hyperdimensional computing: a review, IEEE Circ. Syst. Mag. 20 (2020) 30–47, https://doi.org/10.1109/MCAS.2020.2988388

[30] D. Kleyko, D.A. Rachkovskij, E. Osipov, A. Rahimi, A survey on hyperdimensional computing aka vector symbolic architectures, part II: applications, cognitive models, and challenges, ACM Comput. Surv. 55 (2023) 1–52, https://doi.org/10.1145/3558000

[31] P. Vergés, M. Heddes, I. Nunes, T. Givargis, A. Nicolau, Classification using hyperdimensional computing: a review with comparative analysis, Artif. Intell. Rev. (2023) 1–49, https://doi.org/10.21203/rs.3.rs-3425561/v1

[32] H. Lee, J. Kim, H. Chen, A. Zeira, N. Srinivasa, M. Imani, Y. Kim, Comprehensive integration of hyperdimensional computing with deep learning towards neuro-symbolic AI, in: Proceedings - Design Automation Conference, vol. 2023-July, Institute of Electrical and Electronics Engineers Inc., 2023, https://doi.org/10.1109/DAC56929.2023.10248004

[33] A. Rahimi, P. Kanerva, L. Benini, J.M. Rabaey, Efficient biosignal processing using hyperdimensional computing: network templates for combined learning and classification of ExG signals, Proc. IEEE 107 (2019) 123–143, https://doi.org/10.1109/JPROC.2018.2871163

[34] A. Rahimi, S. Datta, D. Kleyko, E.P. Frady, B. Olshausen, P. Kanerva, J.M. Rabaey, High-dimensional computing as a nanoscalable paradigm, IEEE Trans. Circuits Syst. I Regul. Pap. 64 (2017) 2508–2521, https://doi.org/10.1109/TCSI.2017.2705051

[35] M. Stock, D. Boeckaerts, P. Dewulf, S. Taelman, M.V. Haeverbeke, W.V. Criekinge, B.D. Baets, Hyperdimensional computing: a fast, robust and interpretable paradigm for biological data, PLoS Comput. Biol. (2024) https://doi.org/10.48550/arXiv.2402.17572

[36] R.W. Gayler, S.D. Levy, A distributed basis for analogical mapping, in: Proceedings of the Second International Conference on Analogy, 2009.

[37] M. Hersche, M. Zeqiri, L. Benini, A. Sebastian, A. Rahimi, A neuro-vector-symbolic architecture for solving raven's progressive matrices, Nat. Mach. Intell. 5 (2023) 363–375, https://doi.org/10.48550/arXiv.2203.04571

[38] D. Kleyko, E. Osipov, R.W. Gayler, A.I. Khan, A.G. Dyer, Imitation of honey bees' concept learning processes using vector symbolic architectures, Biol. Inspired Cogn. Archit. 14 (2015) 57–72, https://doi.org/10.1016/j.bica.2015.09.002

[39] G. Montone, J.K. O'Regan, A.V. Terekhov, Hyper-dimensional computing for a visual question-answering system that is trainable end-to-end, in: 31st Conference on Neural Information Processing Systems, 2017, https://doi.org/10.48550/arXiv.1711.10185

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

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

[42] M. Laiho, J.H. Poikonen, P. Kanerva, 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, 2015, pp. 1–4, https://doi.org/10.1109/BioCAS.2015.7348414

[43] D.A. Rachkovskij, Representation and processing of structures with binary sparse distributed codes, IEEE Trans. Knowl. Data Eng. 13 (2001) 261–276, https://doi.org/10.1109/69.917565

[44] 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.

[45] T.A. Plate, Holographic reduced representations, IEEE Trans. Neural Netw. 6 (1995) 623–641, https://doi.org/10.1109/72.377968

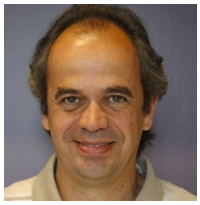
[46] T.A. Plate, Holographic Reduced Representation: Distributed Representation for Cognitive Structures, CSLI Publications, 2003.

[47] D. Kleyko, D.A. Rachkovskij, E. Osipov, A. Rahimi, A survey on hyperdimensional computing aka vector symbolic architectures, Part I: models and data transformations, ACM Comput. Surv. (2022) 1–27, https://doi.org/10.1145/3538531

[48] D. Kleyko, M. Davies, E.P. Frady, P. Kanerva, S.J. Kent, B.A. Olshausen, E. Osipov, J.M. Rabaey, D.A. Rachkovskij, A. Rahimi, F.T. Sommer, Vector symbolic architectures as a computing framework for nanoscale hardware, Comput. Res. Repository (corr) (2021) https://doi.org/10.1109/JPROC.2022.3209104
[49] M. Imani, C. Huang, D. Kong, T. Rosing, Hierarchical hyperdimensional computing for energy efficient classification, in: ACM/ESDA/IEEE Design Automation Conference (DAC), vol. Part F137710, IEEE, 2018, pp. 1–6, https://doi.org/10.1145/3195970.3196060
[50] E. Kussul, 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.
[51] D.A. Rachkovskij, 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.
[52] L. Smets, W.V. Leekwijck, I.J. Tsang, S. Latre, Training a hyperdimensional computing classifier using a threshold on its confidence, Neural Comput. 35 (2023) 2006–2023, https://doi.org/10.1162/neco_a_01618
[53] A. Rahimi, A. Tchouprina, P. Kanerva, J.D.R. Millán, J.M. Rabaey, Hyperdimensional computing for blind and one-shot classification of eeg error-related potentials, Mob. Netw. Appl. 25 (2020) 1958–1969, https://doi.org/10.1007/s11036-017-0942-6
[54] D. Dua, C. Graff, UCI machine learning repository, 2019. http://archive.ics.uci.edu/ml
[55] D. Anguita, A. Ghio, L. Oneto, X. Parra, J.L. Reyes-Ortiz, A public domain dataset for human activity recognition using smartphones, in: 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013, 2013.
[56] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, B. Kim, Sanity checks for saliency maps, in: Conference on Neural Information Processing Systems (NeurIPS), 2018, https://doi.org/10.48550/arXiv.1810.03292

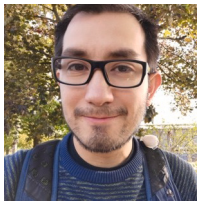
Author biography

Laura Smets received the B.S. and M.Sc. degrees in Bioscience Engineering: Human Health Engineering from the KU Leuven (Belgium) in 2018 and 2020. She is currently pursuing the Ph.D. degree in Computer Science from the University of Antwerp (Belgium). Her research interests include energy-efficient, neuro-inspired algorithms for machine learning tasks.



Werner Van Leekwijck received the B.S. and M.Sc. degrees in Electrical Engineering from the KU Leuven (Belgium), in 1988 and 1990, and the M.Sc. Degree in Knowledge and Information Technology from the University of Ghent (Belgium) in 1994. He is currently a senior researcher at imec, where he leads the group on novel alternative computing systems.

Steven Latré received the B.S., M.Sc. and Ph.D. degrees in Computer Science from the University of Ghent (Belgium) in 2004, 2006 and 2011. He is currently Professor at the Internet Data Lab (IDLab), a joint research lab between the University of Antwerp (Belgium) and imec.



José Oramas received the Ph.D. degree in Electrical Engineering from the KU Leuven (Belgium) in 2015. He is currently Associate Professor at the Internet Data Lab (IDLab), a joint research lab between the University of Antwerp and imec. There, he leads the Interpretable Representation Learning (sqIRL) Lab. For his service to the scientific community, he has been a recipient of several Outstanding Reviewer recognitions at different top-tier venues.