mathematics

Article

Data Classification Methodology for Electronic Noses Using Uniform Manifold Approximation and Projection and Extreme Learning Machine

Jersson X. Leon-Medina [1,2], Núria Parés [3,*], Maribel Anaya [4], Diego A. Tibaduiza [5] and Francesc Pozo [1,6,*]

1 Control, Modeling, Identification and Applications (CoDAlab), Department of Mathematics, Escola d'Enginyeria de Barcelona Est (EEBE), Campus Diagonal-Besòs (CDB), Universitat Politècnica de Catalunya (UPC), Eduard Maristany 16, 08019 Barcelona, Spain; jersson.xavier.leon@upc.edu
2 Ingeniería Mecánica Investigación-USTA (IMECI-USTA), Faculty of Mechanical Engineering, Universidad Santo Tomás-Seccional Tunja, Calle 19 # 11-64, Tunja 150001, Colombia
3 Laboratori de Càlcul Numèric (LaCàN), Department of Mathematics, Escola d'Enginyeria de Barcelona Est (EEBE), Campus Diagonal-Besòs (CDB), Universitat Politècnica de Catalunya (UPC), Eduard Maristany 16, 08019 Barcelona, Spain
4 MEM (Modelling-Electronics and Monitoring Research Group), Faculty of Electronics Engineering, Universidad Santo Tomás, Bogota 110231, Colombia; maribelanaya@usantotomas.edu.co
5 Departamento de Ingeniería Eléctrica y Electrónica, Universidad Nacional de Colombia, Cra 45 No. 26-85, Bogota 111321, Colombia; dtibaduizab@unal.edu.co
6 Institute of Mathematics (IMTech), Universitat Politècnica de Catalunya (UPC), Pau Gargallo 14, 08028 Barcelona, Spain
* Correspondence: nuria.pares@upc.edu (N.P.); francesc.pozo@upc.edu (F.P.)

Abstract: The classification and use of robust methodologies in sensor array applications of electronic noses (ENs) remain an open problem. Among the several steps used in the developed methodologies, data preprocessing improves the classification accuracy of this type of sensor. Data preprocessing methods, such as data transformation and data reduction, enable the treatment of data with anomalies, such as outliers and features, that do not provide quality information; in addition, they reduce the dimensionality of the data, thereby facilitating the tasks of a machine learning classifier. To help solve this problem, in this study, a machine learning methodology is introduced to improve signal processing and develop methodologies for classification when an EN is used. The proposed methodology involves a normalization stage to scale the data from the sensors, using both the well-known min − max approach and the more recent mean-centered unitary group scaling (MCUGS). Next, a manifold learning algorithm for data reduction is applied using uniform manifold approximation and projection (UMAP). The dimensionality of the data at the input of the classification machine is reduced, and an extreme learning machine (ELM) is used as a machine learning classifier algorithm. To validate the EN classification methodology, three datasets of ENs were used. The first dataset was composed of 3600 measurements of 6 volatile organic compounds performed by employing 16 metal-oxide gas sensors. The second dataset was composed of 235 measurements of 3 different qualities of wine, namely, high, average, and low, as evaluated by using an EN sensor array composed of 6 different sensors. The third dataset was composed of 309 measurements of 3 different gases obtained by using an EN sensor array of 2 sensors. A 5-fold cross-validation approach was used to evaluate the proposed methodology. A test set consisting of 25% of the data was used to validate the methodology with unseen data. The results showed a fully correct average classification accuracy of 1 when the MCUGS, UMAP, and ELM methods were used. Finally, the effect of changing the number of target dimensions on the reduction of the number of data was determined based on the highest average classification accuracy.

Keywords: electronic nose (EN); data transformation; data reduction; manifold learning; mean-centered unitary group-scaling (MCUGS); uniform manifold approximation and projection (UMAP); extreme learning machine (ELM); odor recognition

check for updates

Citation: Leon-Medina, J.X.; Parés, N.; Anaya, M.; Tibaduiza, D.A.; Pozo, F. Data Classification Methodology for Electronic Noses Using Uniform Manifold Approximation and Projection and Extreme Learning Machine. *Mathematics* **2022**, *10*, 29. https://doi.org/10.3390/math10010029

Academic Editors: Bo-Hao Chen and Cornelio Yánez Márquez

Received: 4 November 2021
Accepted: 17 December 2021
Published: 22 December 2021

1. Introduction

An electronic nose (EN), or e-nose, is an electronic device that is used as an artificial olfactory system. In general terms, it is used to identify gases having a wide spectrum of odor patterns by utilizing the interaction with a sensor array. As a system, it takes advantage of its various properties for classification tasks and is composed of an array of sensors, a data acquisition system, and a pattern recognition approach [1]. These components, including both software and hardware, require the application of odor capturing by a sensor array, signal conditioning, signal processing, data gathering, data preprocessing, and pattern recognition techniques to perform odor classification [2].

Different approaches to model the response of the chemical sensors in electronic noses have been developed [3,4]. Firstly, deterministic models have been developed to reproduce the sensing mechanism in the metal oxide gas sensors based on the adsorption and desorption process carried out in the sensing material [5–7]. Some physicochemical parameters have been used to improve the selectivity of a metal-oxide gas sensor. For example, a time-domain characterization technique is developed in [8], searching to improve the discrimination ability of an electronic nose fitting a fully analytical model of the electric resistance transient sensor response. Secondly, numerical approaches have been developed to simulate the chemical reaction in the sensing material upon contact with the analyte studied [9,10]. Finally, stochastic models [11,12] have also been developed to reproduce the sensor response and have been linked in the process of classification of substances using electronic nose.

Pattern recognition in an EN enables the treatment of signals based on the identification of features that can be used in classification tasks. The signal processing methodology of an EN is composed of different stages, including the application of dimensionality reduction techniques, such as feature extraction or feature selection, to identify the best information, suppress outliers and noises in the collected signals, and decrease the false detection/classification rates [13]. Although a dimensionality reduction technique can filter redundant information by denoising, some useful information in the original data may be lost, thus affecting the final results of the classification. In addition, the classification method can automatically remove the useless components during the learning process of the sample [14].

Although several studies have been conducted with this aim, as described in the following section, this remains an open research topic because of the option to include new techniques to improve the classification process. To solve this problem, a classification methodology for classifying data from an EN-type sensor is proposed in this study. This methodology consists of several stages developed for EN-type sensor arrays using machine learning and signal processing techniques. Uniform manifold approximation and projection (UMAP) is used as a type of manifold learning to achieve dimensionality reduction, and an extreme learning machine (ELM) is used as a classifier. The methodology is evaluated by using three different EN datasets, and it achieves excellent results.

The overall contributions of this study can be summarized as follows:

1. A methodology for classifying data obtained by using a sensor array was developed for ENs. A crucial part of this methodology is data preprocessing, including a data arrangement using an unfolding procedure, $min - max$ or mean-centered unitary group-scaling (MCUGS) for dealing with the different magnitudes of the sensors (data normalization), a reduction of the manifold learning data by applying the supervised UMAP algorithm.
2. The data classification process in the methodology is performed through an ELM classifier, which is a fast and accurate method for discriminating the different classes. A two-stage validation/verification evaluation process is carried out. First, in the training stage, 5-fold cross-validation (CV) is used to prevent an overfitting of 75% of the data, and second, a testing set is formed with the remaining 25% of the data and is used to verify the performance of the methodology with unseen data.

3. The methodology was validated using three different datasets for classification: (i) Six distinct gases (dataset #1), (ii) three different qualities of wine (dataset #2), and (iii) three classes of gases (dataset #3). The average classification accuracy was used as a performance measure of the proposed process. The high average accuracy achieved in both training and testing sets indicated the effectiveness of the proposed methodology.
4. The methodology can be used in imbalanced multi-class classification problems, as evidenced on datasets #2 and #3, which exhibited an imbalanced behavior in the number of samples per class.

The remainder of this paper is organized as follows. Section 2 presents a brief review of the related studies on the various uses of ENs and the available methods established in the literature for data analysis and processing of this type of sensor. Section 3 describes the methodology needed to improve the classification of data derived from EN sensors, along with the classification methodology applied and its step-by-step processing. Next, in Section 4, three EN datasets used to validate the classification methodology are described and a brief overview of the data acquisition and transformation methods used in the proposed EN data classification methodology is presented. Next, the data reduction stage using UMAP is described in Section 5. The data classification stage using the ELM method is then detailed in Section 6. Section 7 outlines the validation/verification procedure and describes the training/test data split. The classification performance measures are then defined. In Section 8, the experiment results and a discussion are provided. In addition, the main results after the application of the developed methodology for the three datasets are presented, including the normalization, dimensionality reduction, confusion matrix, average classification performance metrics, and tuning parameters for each method. Finally, some concluding remarks are provided in Section 9.

2. Related Work

Data preprocessing can improve the classification of different analytes in data from several sensors, including the sensors used by ENs. Different linear and nonlinear methods can be used in the data reduction stage. Although principal component analysis (PCA) [15] is used as a linear method, in most cases, data have nonlinear characteristics that cannot be identified through a linear method. For this reason, different nonlinear data reduction methods, which are members of the group of manifold learning algorithms [16,17], can be used to deal with the data reduction stage. In recent years, different studies related to the use of manifold learning algorithms in ENs have been conducted. For instance, the modified unsupervised discriminant projection [18] and Laplacian eigenmaps (LEs) [19] were used as data reduction methods for the rapid determination of the freshness of Chinese mitten crab, similarly the supervised locality-preserving projection was used to process the feature matrix before applying it to the classifier to improve the performance of an EN [20].

Different perspectives in the literature have been developed to deal with classification problems in ENs [21]. One such perspective is the machine learning approach, which can be divided in supervised, semi-supervised, or unsupervised methods. It delivers qualitative answers by which different types of analytes are identified [13]. Different supervised learning algorithms have been used to solve classification tasks in ENs, including support vector machines (SVMs) [22], artificial neural networks [23], and a new kernel discriminant analysis [14]. In addition, the *k*-nearest neighbor (KNN) algorithm with one neighbor and the Euclidean distance has proven to be efficient [24] solving multi-class problems, and obtaining high classification rates.

There are a significant number of datasets available in the literature related to the classification of EN signals. One such dataset was used in the study by Vergara et al. (2012) [25], who measured six volatile organic compounds during a 3-year period under strongly controlled operating conditions and using a series of 16 metal-oxide gas sensors [26]. In that study, the authors used a classifier ensemble as a supervised learning method, to treat the sensor drift problem in ENs.

Leon-Medina et al. [21] developed an EN machine learning classification methodology. In their study, four nonlinear data reduction algorithms were compared in an EN classification task: A kernel PCA, LEs, locally linear embedding, and Isomap. A KNN algorithm was used as the classification model. The results when applying these algorithms reached a classification accuracy of 98.33% after performing a holdout CV.

In 2019, Vallejo et al. [27] reviewed some models for classification and regression tasks usually determined by human perception, such as smell, and they called these measurement techniques soft metrology. In their study, the classification process in soft metrology is composed of several stages such as a database construction and preprocessing, construction of an effective representation space, model choice training and validation, and model maintenance.

Different preprocessing techniques for data reduction in big data have been developed; for example, wavelet packet decomposition (WPD) is used in [28] for selecting features that have maximal separability according to the Fisher distance criterion. The WPD method was compared with fast Fourier transform and autoregressive (AR) models for data transformation. The best classification accuracy was obtained using the WPD method, reaching a value of 90.8%.

Several studies on the use of machine learning in sensor arrays such as ENs have recently been conducted. Some drift problems in ENs are normally found owing to the deterioration of the sensors over time. A cross-domain discriminative subspace learning (CDSL) method was developed in [29]. This domain adaptation-based odor recognition method allows solving recognition tasks in systems of master and slave ENs. In this way, two different configurations of the source and target domains were solved. The maximum average accuracies reached by the CDSL method for the two configurations of the source and target domains were 78.96% and 80.17%, respectively.

In 2014, Zhang and Zhang [30] developed the domain adaptation ELM (DAELM) method to solve the drift problem. This method learns a robust classifier by leveraging a limited number of labeled data from a target domain for drift compensation in ENs. The developed DAELM method was validated using a dataset of an EN sensor array captured during a 36-month period, and two different settings were validated using this method, reaching average accuracies of 91.8% under both settings. In 2020, Kumar and Ghosh [31] developed a system identification approach for data transformation and reduction in sensor arrays. The equivalent circuit parameter method was compared against a discrete wavelet transform and the neighborhood component analysis, which found a significant reduction in the number of features in the machine learning tasks. The best average prediction accuracy was reached by the neural network regression method, with a value of 98%. A novel bio-inspired neural network [32] processes the raw data of an EN without any signal preprocessing, feature selection, or reduction. This significantly simplifies the data processing procedure in ENs. The results showed a classification accuracy of 100% in the task of recognizing seven classes of Chinese liquors. It is worth mentioning that although a classification accuracy of 100% is reached, this was only tested for the signals acquired with the electronic nose developed in [32], in addition, the proposed olfactory neural network has a problem with respect to the parameter tuning.

A portable EN instrument was developed in [33] for qualitative discrimination among different gas samples. A multivariate data analysis of this instrument was conducted using a PCA. Good results were obtained in the qualitative and quantitative tests conducted using this instrument for three industrial gases: Acetone, chloroform, and methanol. Krutzler et al. [34] showed that the determination of gas concentration is improved through the implementation of sensors with smaller resistance variations when the reference library from one sensor is also used for other sensor elements. In 2012, Brudzewski et al. [35] used a nonlinear classifier in the form of a Gaussian kernel SVM to classify EN data. A total of 11 classes of coffee brand mixtures were correctly classified with an average error of 0.21%

using a differential nose system containing two arrays of semiconductor sensors composed of pairs of the same sensors. An incremental-learning fuzzy model for the classification of black tea using an EN is described in Tudu et al. [36]. With the use of the approach they developed, a universal computational model can evolve incrementally to automatically include the newly presented patterns in the training dataset without affecting the class integrity of the previously trained system. After a 10-fold CV procedure, an average classification rate of 80% with a standard deviation of 4.969% was obtained.

In 2014, Zhang and Tian [37] developed a multilayer perceptron (MLP) based on a multiple-input, single-output approach for an estimation of the simultaneous concentration of multiple types of chemicals in an EN. The lowest average mean square error of prediction was 3.33%, obtained by a particle swarm optimization method based on the bacterial chemotaxis backpropagation (BP) method. A study describing the extension of neuromorphic methods for artificial olfaction is presented in [38]. The neuromorphic engineering aims to overcome data processing challenges and reduce the output latency. The neuromorphic approach was applied in [39] for gas recognition of three target gases, namely, ethanol, methane, and carbon monoxide, with a 100% classification accuracy. A hybrid approach using both convolutional and recurrent neural networks (CRNNs), based on the long short-term memory module, was proposed in [40]. A deep learning method is well-suited for extracting the valuable transient feature contained at the very beginning of the response curve. The reported accuracy dramatically outperformed the previous algorithms, including gradient tree boosting, random forest (RF), SVM, KNN, and linear discriminant analysis. The CRNN approach reached an average classification accuracy of 98.28% when 4-s signals were used.

In 2019, Liu et al. [41] proposed a multi-task model based on the BP neural network (MBPNN) for EN classification problems. The approach they developed was compared against RF and SVM. Their study showed that the EN is effective for the classification and evaluation of organic green tea. The MBPNN method was satisfactorily used in two datasets of ENs for classification tasks, reaching accuracies of 99.83% and 99.67%. A multi-task learning-long-short-term memory (MLSTM) recurrent network was proposed in [42] for gas classification and concentration estimation (regression) in ENs. The best classification accuracies obtained with the MLSTM method for each of the three datasets presented were 99.99%, 95.17%, and 100%. Cheng et al. [43] proposed a solution to the problem of dynamically growing odor datasets while both the training samples and number of classes increase over time. The solution uses a deep nearest class mean (DNCM) model based on a deep learning framework and the nearest class mean method. The results showed that the DNCM model is extremely efficient for incremental odor classification, particularly for new classes with only a few training examples. The best average recognition accuracy was 78.09%, obtained using the DNCM method.

In 2020, an AR process associated with an observation of zero-mean Gaussian noise was used to solve drift issues in EN-type sensor arrays [44]. In this approach, each sensor response passes through a separate Kalman filter and a regression technique is used to predict the sensor response. In 2018, Zhang et al. [45] developed a subspace learning methodology for classifying data originating from a sensor array. A sliding window-based smooth filter was employed for denoising and feature selection, a local discriminant preservation projection approach was applied for dimensionality reduction, and a kernel ELM was used as a classifier. The best results showed a classification accuracy of 98.22% on a training set using 5-fold CV.

In summary, some limitations of the existing methods in the literature that lead to the development of the methodology proposed in this study are: The low flexibility of the models, the need to make the dimensionality reduction model each time a new data is obtained, and the inability to perform a quick classification. To treat these limitations, the methodology for electronic nose signal classification developed in this study includes: (i) The correct verification in three different electronic nose datasets, (ii) the use of the supervised variant of the UMAP method for data reduction allowing for mapping new data to a low dimensional space using the model saved in the training, and (iii) the use of the ELM classifier algorithm with all its capabilities related with an extremely fast learning speed and good generalization performance.

3. Methodology Description

This study presents a methodology for classifying signals acquired by EN systems that can yield high classification rates. The classifier is constructed using four stages (see Figure 1).

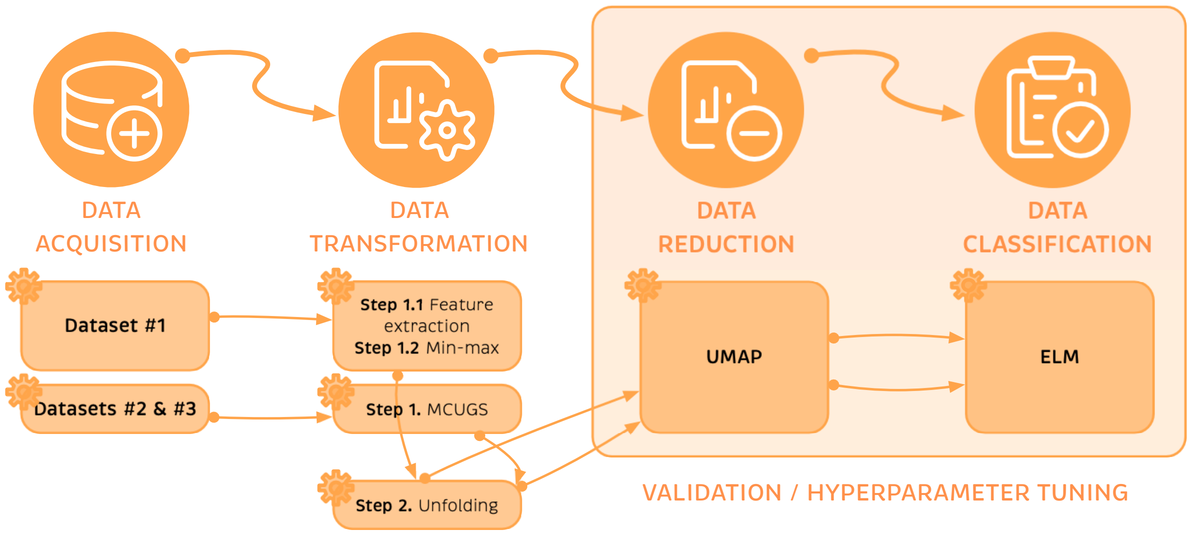

Figure 1. The data classification methodology for ENs(Electronic Noses) is divided into four stages: Data acquisition, data transformation, data reduction, and data classification. MCUGS, mean-centered unitary group-scaling; UMAP, uniform manifold approximation and projection; ELM, extreme learning machine.

The first stage is data acquisition, where the EN (as a multi-sensor system) collects data in a three-dimensional matrix: The rows and columns of the matrix contain the data acquired during each experiment/test per time instant for a specific sensor, whereas the third dimension stores the information of each sensor. In the second stage, these data are first properly transformed to consider whether the data captured by each sensor or their associated extracted features can present significant differences in their magnitudes. Owing to the different nature of the considered data, two different normalization approaches have been considered: The $min - max$ normalization and MCUGS. Later, with the use of an unfolding process, the three-dimensional matrix containing the data is stored as a two-dimensional matrix. The third step concerns data reduction, whereas the UMAP method is applied to the high-dimensional EN-transformed data for discarding features that are irrelevant to the classification problem. After data reduction, the low-dimensional data serve as an input for training a machine learning classifier. In this case, the ELM algorithm is used owing to its capabilities, such as better generalization ability, robustness, controllability, and fast learning. Once the classifier is set, given a new experimental sample, the classification algorithm can predict its class. However, before the classifier is used to predict new class samples, it is crucial to evaluate its performance. This evaluation is a challenging task because the available data must be used to both define the classifier and estimate its performance. Here, a two-stage validation/verification evaluation is considered (see Figure 2). During the validation step, a standard 5-fold CV is performed by using a training set containing 75% of the total data (see [15]). This allows for the suitable tuning of the number of extracted features of the UMAP method and the hyperparameters of the ELM classifier using a GridSearch algorithm, while providing standard estimates of the overall performance of the model. However, although validation performance measures are usually used in the literature to assess the quality of a classification strategy, an extra verification step is added to avoid an overfitting. Once the UMAP model and the ELM classifier are defined, the strategy is tested using the remaining 25% of the data. The key point here is that these test data are completely independent of the training step because they are not involved in the supervised feature extraction step nor in the tuning of the classifier parameters.

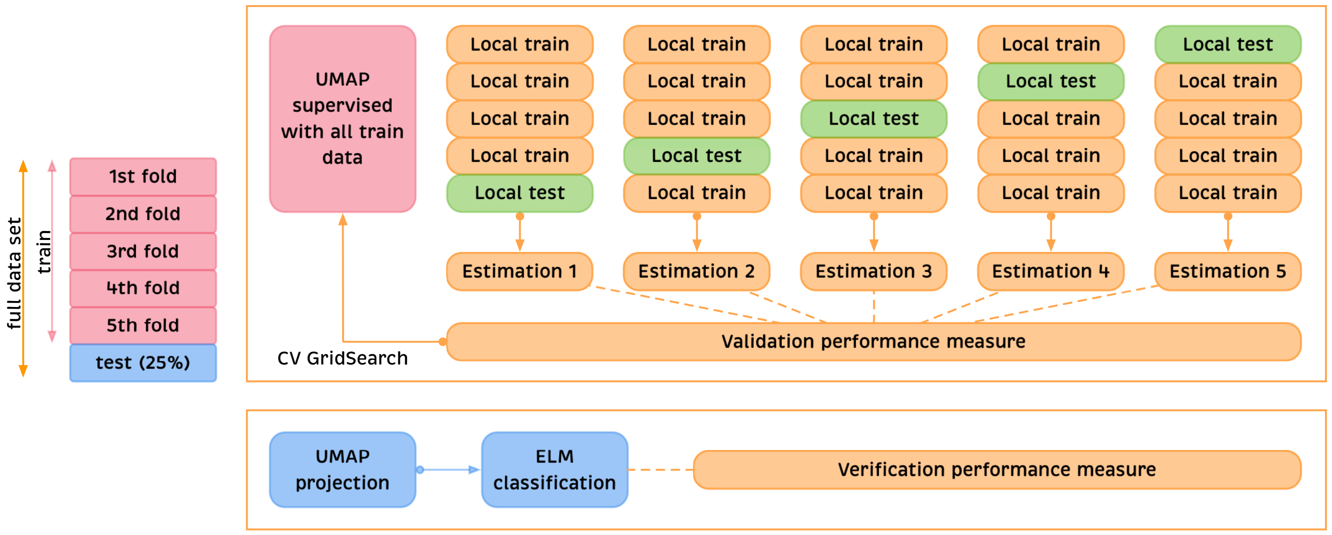

Figure 2. Model validation/verification using 75% of the data in the 5-fold CV used to tune the hyperparameters and 25% of the data for the final verification.

Consequently, two different performance measures are obtained, one for the training set (associated with the final hyperparameters after applying the CV GridSearch) and one for the test set.

4. Datasets Description, Data Acquisition, and Transformation

The contributions of this study include the approaches to data preprocessing: How the data are arranged, scaled, transformed, and reduced. In this section, data preprocessing is presented in adequate detail, along with the three different datasets used for the validation of the proposed approach. More precisely, the preprocessing is divided into three phases: Data acquisition, data transformation (feature extraction and $\min - \max$ normalization for dataset #1, and MCUGS for datasets #2 and #3), and data reduction (Appendix A). Figure 1 illustrates these three phases.

4.1. Dataset #1

The first dataset was created by Vergara et al. [25] and is composed of 3600 measurements of six volatile organic compounds (acetaldehyde, ethanol, toluene, ammonia, ethylene, and acetone) under strongly controlled operating conditions recorded from 16 metal-oxide semiconductor gas sensors manufactured by Figaro, Inc. (figarosensor.com, accessed on 10 October 2021).

4.1.1. Data Acquisition

The EN employed to collect dataset #1 used a test chamber linked to a computer-controlled continuous flow system [21]. The total flow rate through the detection chamber was set to 200 mL/min. Synthetic dry air was used as the background for all measurements to keep the humidity level constant at 10% relative humidity (measured at 25 ± 1 °C). The operating temperature of the sensors was set to 400 °C. The dataset was obtained under laboratory conditions with a controlled atmosphere, which favors those materials of the sensors that interact with the gas in a reversible manner [25].

The experiments were carried out using an EN for measuring the behavior of 6 analytes during a 36-month period. The current study considered the 10th batch of the measurements, corresponding to the 36th month. More precisely, batch number 10 corresponded to the last batch of recordings in the study by Vergara et al. [25]. This batch contained 3600 measurements from the analytes captured during month 36 of the author's study and was collected 5 months after batch 9. During this 5-month period between the collection of measurements from batch 9 to batch 10, the sensors were turned off and their response capacity was affected owing to a lack of operating temperature [46], making the data collected during the 10th batch extremely relevant for validating the classification strategies. As previously stated, there were a total of 3600 experiments, divided into six classes. Table 1 describes the number of samples per class (gas) in this dataset. The duration, T, of the experiments was not constant, spanning at least 300 s (comprising three phases of measurement: Baseline measurements injecting only pure air, test gas measurements injecting the gas, and a recovery phase) under a sampling frequency of 100 Hz. Therefore, during each experiment, at least 300 s $\times$ 100 Hz $=$ 30,000 resistance measurements per sensor were acquired. These collected data were arranged into a three-dimensional matrix with a size of $n \times M \times N$, where $n = 3600$ indicates the total number of experiments, $M > 30{,}000$ represents the number of time measures, and $N = 16$ shows the number of sensors. For the data transformation, the sensor measures were grouped into $n \times M$ matrices, $\mathbf{X}^1_{\text{raw}}, \mathbf{X}^2_{\text{raw}}, \ldots, \mathbf{X}^N_{\text{raw}}$ (see Figure 3).

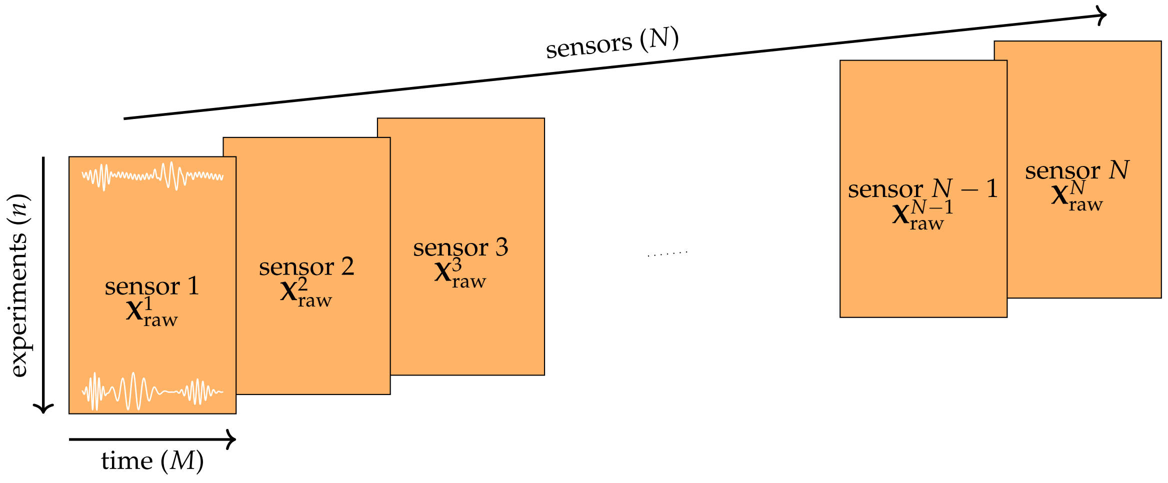
sensors (N)
experiments (n)
time (M)
sensor 1 $\mathbf{X}^1_{\text{raw}}$
sensor 2 $\mathbf{X}^2_{\text{raw}}$
sensor 3 $\mathbf{X}^3_{\text{raw}}$
sensor $N-1$ $\mathbf{X}^{N-1}_{\text{raw}}$
sensor N $\mathbf{X}^N_{\text{raw}}$

Figure 3. Initial arrangement of the collected data.

Table 1. Number of measures in batch #10 of the EN dataset #1.

Index	Class Name	Number of Measures
1	Acetaldehyde	600
2	Ethanol	600
3	Toluene	600
4	Ammonia	600
5	Ethylene	600
6	Acetone	600
	Total	3600

4.1.2. Data Transformation

Feature Extraction: Feature extraction usually plays an important role in data preprocessing for chemo-sensory applications, transforming the raw sensor responses while preserving the most meaningful portion of the information contained in the original sensor signal. For this dataset, two distinct types of features were considered:

(a) Two steady-state responses of the sensor element; and
(b) Three rising and three decaying transient portions of the sensor response.

Therefore, from the $M > 30{,}000$ resistance measurements per sensor stored in the columns of matrix $\mathbf{X}^k_{\text{raw}}$, $k = 1, \ldots, N$, only $F = 8$ features were extracted.

Specifically, for a given experiment $i = 1, \ldots, n$ and sensor $k = 1, \ldots, N$, let $\mathbf{r}^k_i = \text{row}_i(\mathbf{X}^k_{\text{raw}}) \in \mathbb{R}^M$ be the ith row of matrix $\mathbf{X}^k_{\text{raw}}$ collecting all the time profiles of the resistance measures for the given sensor associated with the given experiment. Note that the jth component of this vector r^k_{ij}, $j = 1, \ldots, M$, corresponds to the resistance acquired at time $t = j/100 \in [0, T]$. Therefore, $\mathbf{X}^k_{\text{raw}} = (r^k_{ij})$ is the matrix having a raw resistance value of r^k_{ij} as the (i, j)th entry.

The two features corresponding to the steady-state response of the sensor are the difference between the maximum and minimum/baseline resistance measurements,

$$\Delta R^{ik} = \max(\mathbf{r}^k_i) - \min(\mathbf{r}^k_i),$$

and its normalized version,

$$\Delta_n R^{ik} = \frac{\Delta R^{ik}}{\min(\mathbf{r}^k_i)},$$

where the $\min(\cdot)$ and $\max(\cdot)$ functions return the minimum and maximum values of a vector, respectively.

The second set of six features reflecting the sensor dynamics of the increasing/decaying transient portions of the sensor response are computed by first introducing the exponential moving average (EMA) of the signal. Given a smoothing factor $\alpha \in [0, 1]$, the EMA of the sensor resistance with respect to the smoothing factor α is a new vector $\text{ema}_\alpha(\mathbf{r}^k_i) \in \mathbb{R}^M$ defined as $\text{ema}_\alpha(\mathbf{r}^k_i)_1 = 0$ and for $l = 2, \ldots, M$ is defined as:

$$\text{ema}_\alpha(\mathbf{r}^k_i)_l = (1 - \alpha)\text{ema}_\alpha(\mathbf{r}^k_i)_{l-1} + \alpha(r^k_{il} - r^k_{i,l-1}),$$

where $\text{ema}_\alpha(\mathbf{r}^k_i)_l$ is the lst component of the EMA signal. The maximum and minimum values of the EMA signal, $M^{ik}_\alpha = \max(\text{ema}_\alpha(\mathbf{r}_i))$ and $m^{ik}_\alpha = \min(\text{ema}_\alpha(\mathbf{r}^k_i))$, respectively, are characteristic features of the rising and decaying portion of the sensor response. Therefore, the second set of six features is obtained by computing the minimum and maximum values of the EMA signals for $\alpha = 0.001, 0.01$, and 0.1.

Thus, each vector $\mathbf{r}^k_i \in \mathbb{R}^M$ is mapped into vector $\tilde{\mathbf{r}}^k_i \in \mathbb{R}^8$:

$$\begin{aligned}\tilde{\mathbf{r}}^{ik} = (&\Delta R^{ik}, \Delta_n R^{ik}, \\ &M^{ik}_{0.001}, m^{ik}_{0.001}, M^{ik}_{0.01}, m^{ik}_{0.01}, M^{ik}_{0.1}, m^{ik}_{0.1}).\end{aligned}$$

By computing the abovementioned $F = 8$ features for each of the experiments and each of the $N = 16$ sensors in the prerecorded time series, we can map the $n \times M$ matrices $\mathbf{X}^1_{\text{raw}}, \mathbf{X}^2_{\text{raw}}, \ldots, \mathbf{X}^N_{\text{raw}}$ to the $n \times F$ matrices $\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^N$, where $\text{row}_i(\mathbf{X}^k) = \tilde{\mathbf{r}}^k_i$.

Figure 4 illustrates a reference time profile of the sensor resistance where the three phases (baseline, test gas, and recovery) can be clearly distinguished along with their associated EMA signals and extracted features.

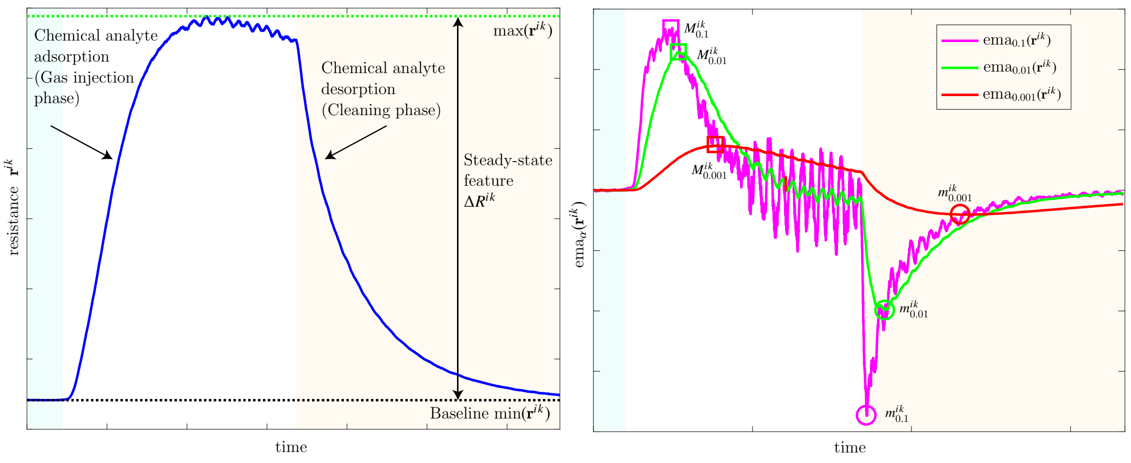

Figure 4. Illustration of the response of a sensor of an EN and its EMA signals. Data taken from dataset #2 resistance values.

Data Normalization: Both because the nature of the eight features is extremely diverse (steady-state raw and normalized, and dynamic) and because different types of sensors are used to acquire the signals, significant differences are appreciated in the magnitudes of the data. To make the data comparable, we apply a data normalization procedure. For each sensor $k = 1, \ldots, N$ and feature $j = 1, \ldots, F$, let:

$$m_j^k = \min(\mathrm{col}_j(\mathbf{X}^k)) = \min_{1\le i\le n} x_{ij}^k,$$
$$M_j^k = \max(\mathrm{col}_j(\mathbf{X}^k)) = \max_{1\le i\le n} x_{ij}^k,$$

be the minimum and maximum values of feature j computed by sensor k, respectively, where $\mathbf{X}^k = (x_{ij}^k)$, i.e., x_{ij}^k is the (i,j)th element entry of matrix $\mathbf{X}^k$ and $\mathrm{col}_j(\mathbf{X}^k)$ is the jth column of matrix $\mathbf{X}^k$. The features, x_{ij}^k, in these matrices are scaled to lie between $m = -1$ and $M = 1$ by applying the min $-$ max normalization:

$$\check{x}_{ij}^k = \frac{x_{ij}^k - m_j^k}{M_j^k - m_j^k}(M - m) + m.$$

Therefore, elements x_{ij}^k of matrix $\mathbf{X}^k$, $k = 1, \ldots, N$, are scaled to create a new matrix, $\check{\mathbf{X}}^k = (\check{x}_{ij}^k)$. For simplicity, the min $-$ max normalized matrices, $\check{\mathbf{X}}^k$, are renamed simply as $\mathbf{X}^k$.

4.2. Dataset #2

In 2019, Rodriguez Gamboa et al. [47,48] developed a wine-quality detection system using an EN. The EN wine-quality detection system consists of an array of six metal-oxide gas sensors. These sensors, manufactured by Hanwei Sensors, were chosen because of their high sensitivity to alcohol and low sensitivity to benzine, high sensitivity to methane and natural gas, and high sensitivity to the liquefied petroleum gases isobutane and propane. The detection system uses a deep MLP neural network with online detection through a rising window method to process an early portion of the sensor signals.

4.2.1. Data Acquisition

The total number of experiments on dataset #2 was 235, which was divided into three classes (high-quality, average-quality, and low-quality wine). Table 2 lists the number of samples per class in this dataset and the ranges of the acetic acid detected and the volatile acidity according to the wine spoilage thresholds. Each experiment in the data acquisition had a duration of $T = 180$ s using a sample frequency of 18.5 Hz. The collected data were arranged, as in dataset #1, into a three-dimensional matrix of size $n \times M \times N$, where $n = 235$ indicates the total number of experiments, $M = 180\ \text{s} \times 18.5\ \text{Hz} = 3330$ represents the number of time measures, and $N = 6$ indicates the number of sensors. For data transformation purposes, sensor measures were also grouped into $n \times M$ matrices $\mathbf{X}^1_{\text{raw}}, \mathbf{X}^2_{\text{raw}}, \ldots, \mathbf{X}^N_{\text{raw}}$ (see Figure 3).

4.2.2. Data Transformation

Feature Extraction: Data transformation for dataset #1 in Section 4.1 is devised as a two-step procedure: (i) Feature extraction and (ii) data normalization. In the original study [47], a similar approach was conducted for the feature extraction and selection of dataset #2, where 23 features captured the dynamic and static behavior of each sensor. However, in the current study, because the raw data are available [48], the full set of sensor values were maintained. Therefore, the number of considered features coincided with the number of time samples, $F = M$.

Mean-Centered Unitary Group-Scaling: This method, which has been recently applied to the detection and classification of structural damage in wind turbines [15], is considered to be an alternative data normalization procedure. The MCUGS is formulated as a two-step normalization technique. The columns are modified by subtracting the mean of each column (column-wise scaling), and the mean-centered data are scaled by using the standard deviation of the scaled matrix. More precisely, for each sensor $k = 1, \ldots, N$ and time instant $j = 1, \ldots, M$, we define:

$$\mu^k_j = \text{mean}(\text{col}_j(\mathbf{X}^k_{\text{raw}})) = \frac{1}{n}\sum_{i=1}^{n} r^k_{ij},$$

to be the mean of all measures acquired by sensor k at time instant j, where $\mathbf{X}^k_{\text{raw}} = (r^k_{ij})$. Therefore, the mean-centered matrices, $\mathbf{X}^k = \left(x^k_{ij}\right)$, are defined as:

$$x^k_{ij} = r^k_{ij} - \mu^k_j.$$

Then, because:

$$\sum_{i=1}^{n}\sum_{j=1}^{M} x^k_{ij} = 0,$$

the standard deviation of the mean-centered matrices, $\mathbf{X}^k$, is computed as follows:

$$\sigma^k = \sqrt{\frac{1}{nM}\sum_{i=1}^{n}\sum_{j=1}^{M}\left(x^k_{ij}\right)^2}.$$

Therefore, matrix $\mathbf{X}^k$ is scaled by dividing all the data by the standard deviation σ^k:

$$\check{x}^k_{ij} = \frac{x^k_{ij}}{\sigma^k},$$

to create the MCUGS matrix $\check{\mathbf{X}}^k = \left(\check{x}^k_{ij}\right)$. As described in Section 4.1, for simplicity, the normalized matrices $\check{\mathbf{X}}^k$ are renamed simply as $\mathbf{X}^k$.

Table 2. Number of measures in wine of the EN dataset #2.

Index	Class Name	Number of Measures	Volatile Acidity in g/L	Acetic Acid in g/L
1	High-Quality Wine	51	[0.15, 0.3]	not detected or [0, 0.23]
2	Average-Quality Wine	43	[0.31, 0.41]	[0.24, 0.34]
3	Low-Quality Wine	141	[0.8, 3]	[0.74, 2.75]
	Total	235		

4.3. Dataset #3

The third dataset used to validate the methodology for classifying EN signals was used by Yin et al. [49], who proposed a temperature-modulated EN system using two sensors from Figaro, Inc. The reference models of these two sensors are TGS2620 and TGS2602, which were selected because of their wider detection range and higher sensitivity for the monitoring of multiple gases.

4.3.1. Data Acquisition

The total number of experiments in dataset #3 was 309, divided into three classes (carbon monoxide (CO), formaldehyde (HCHO), and nitrogen oxide (NO_2)) . Table 3 shows the number of samples per class in this dataset. Each experiment in the data acquisition had a duration of $T = 50$ s using a sample frequency of 100 Hz. The collected data were arranged, as in datasets #1 and #2, into a three-dimensional matrix of size $n \times M \times N$, where $n = 309$ represents the total number of experiments, $M = 50\text{ s} \times 100\text{ Hz} = 5000$ indicates the number of time measures, and $N = 2$ represents the number of sensors. For data transformation purposes, sensor measures were also grouped into $n \times M$ matrices $\mathbf{X}^1_{\text{raw}}$ and $\mathbf{X}^2_{\text{raw}}$ (see Figure 3).

4.3.2. Data Transformation: MCUGS

To obtain the scaled matrices $\check{\mathbf{X}}^1$ and $\check{\mathbf{X}}^2$, we normalize the raw matrices $\mathbf{X}^1_{\text{raw}}$ and $\mathbf{X}^2_{\text{raw}}$ using the MCUGS, as described in Section 4.2.2. These matrices are renamed simply as $\mathbf{X}^1$ and $\mathbf{X}^2$. For uniformity in the notations in Section 4.4, as in dataset #2, let $F = M$.

Table 3. Number of measures of the EN dataset #3.

Index	Class Names	Number of Measures
1	Carbon monoxide (CO)	96
2	Formaldehyde (HCHO)	100
3	Nitrogen oxide (NO_2)	113
	Total	309

4.4. Data Unfolding

Before the data reduction, the normalized matrices $\mathbf{X}^k \in \mathbb{R}^{n \times F}, k = 1, \ldots, N$, are concatenated horizontally such that the data are collected into a single $n \times D$ matrix $\mathbf{X}$ as follows:

$$\mathbf{X} = \left(\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^N\right), \tag{1}$$

where $D = F \cdot N$.

Each row of matrix $\mathbf{X}$ is a $D-$dimensional vector that contains the data associated with a particular experiment. The ith row of matrix $\mathbf{X}$ collecting the information of all sensors for the ith experiment is:

$$\text{row}_i(\mathbf{X}) = (x^1_{i1}, \ldots, x^1_{iF}, x^2_{i1}, \ldots, x^2_{iF}, \ldots, x^N_{i1}, \ldots, x^N_{iF}).$$

For clarity, and as a summary, note that the dimensions of matrix $\mathbf{X}$ are as follows:

- $n \times (F \cdot N) = 3600 \times (8 \cdot 16) = 3600 \times 128$, for dataset #1.
- $n \times (F \cdot N) = 235 \times (3330 \cdot 6) = 235\times$ 19,980, for dataset #2.
- $n \times (F \cdot N) = 309 \times (5000 \cdot 2) = 309\times$ 10,000, for dataset #3.

The rows of matrix $\mathbf{X}$ (for each of the different datasets) are the input samples for the data reduction technique described in Appendix A. The new samples, with a reduced dimension, are the input of the data-driven classification methodology described in Section 6. The process of the data arrangement, normalization, and unfolding in the case of datasets #2 and #3 is illustrated in Figure 5.

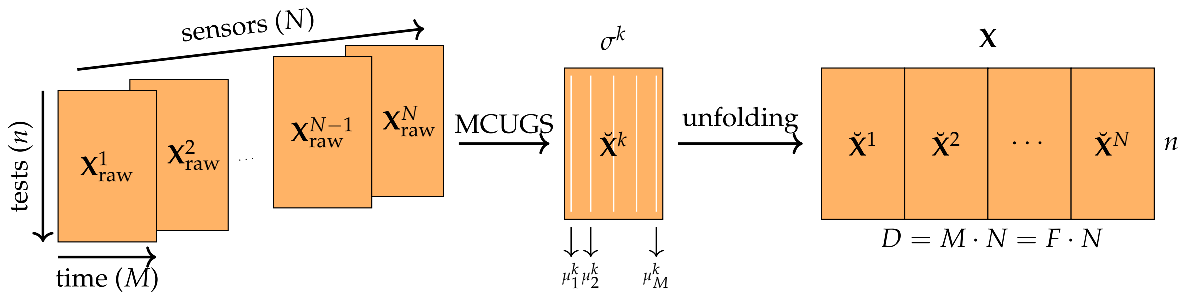

Figure 5. Initial data arrangement, MCUGS and unfolding dataset #2 and dataset #3, where $F = M$.

5. Data Reduction: UMAP

A proper selection of the data reduction approach and the number of target dimensions can have a significant impact on the power of the classification [50]. Dimensionality reduction should be used not only for visualization or as a preprocessing step for extremely high dimensional data but also as a general preprocessing technique for datasets that contain many variables not related to the target parameter to increase the classification accuracy. However, the difficult selection of both the dimensionality reduction technique and the reduced number of dimensions should be based directly on the effects of the classification power, using performance metrics such as accuracy [21].

When the input patterns lie on or near a low-dimensional submanifold of the input space, the structure of the dataset may be highly nonlinear, and therefore, linear methods are bound to fail [51]. To solve this issue, researchers have developed some manifold learning algorithms that serve as data reduction methods. This type of algorithm can be classified into two categories: Graph-based and kernel methods. The graph-based methods construct a sparse graph in which the nodes represent the input patterns and the edges represent the neighborhood relations [51]. Some graph-based methods include UMAP, Isomap, LEs, and locally linear embedding. The kernel-based methods are based on a kernel trick, which is used to generalize a linear method in statistical learning to nonlinear settings [52]. One example of a linear method that can be generalized to a nonlinear version is PCA, which can be generalized to its kernel version, i.e., a kernel PCA [53].

A detailed explanation of the UMAP method is beyond the scope of this study; see [54] for further details. To describe the background and motivation of the proposed methodology, we provide a practical computational description of the UMAP method in Appendix A.

A brief description of the main UMAP parameters used in this study is provided in Table 4. It is important to note that, in this study, any parameter that is not defined is set to its default value.

Table 4. Description of the key hyperparameters of the UMAP method.

Parameter	Value	Default	Description
n_components	$\mathbb{N}$	2	dimension of the low-dimensional space (d)
n_neighbors	$\mathbb{N}$	15	number of neighbors of the k-neighbor graph (k)
metric		Euclidean	metric to compute the distances in the high dimensional space $\mathbb{R}^D$ (d)
n_epochs	$\mathbb{N}$	none	number of training epochs of the SGD method (by default 200 or 500 for large and small datasets, respectively)
min_dist	$[0, +\infty)$	0.1	minimum distance between points in the low dimensional representation (determining a and b)
spread	$(0, +\infty)$	1	effective scale of the embedded points, where min_dist> spread (determining a and b)

Supervised UMAP and Metric Learning

Two extensions of the standard UMAP method briefly outlined in Appendix A have been recently introduced (see [55,56]). The key ideas of these extensions are highlighted in this section.

The first extension is useful when a dimensionality reduction technique is used for classification purposes. In this case, a supervised manifold learning algorithm is available, where class labels are applied during the feature selection process to better describe the low-dimensional data in terms of class information. In short, this extension of the UMAP method takes the labels of each sample, and after a metric space is considered to describe the categorical distance, it includes this information in the similarity function p_{ij}. The effect of the metrics provided by the data and the labels is controlled using the hyperparameter (target_weight $\in [0, 1]$) controlling the relative weight of the two metrics in the final similarity function (a value of 0 favors the data, whereas a value of 1 puts almost all of the weight onto the label distance).

The second extension is crucial to classifying new unseen data. The original UMAP method can be classified as a nonparametric algorithm that transforms the full data matrix $\mathbf{X} \in \mathbb{R}^{n\times D}$ into a reduced low-order d-dimensional manifold described by matrix $\mathbf{Y} \in \mathbb{R}^{n\times d}$ when minimizing the cross-entropy using a SGD method. However, if new data are available, it is not possible to recover their associated d-reduced features for use in the classification step. Fortunately, an extension of the UMAP method [57] incorporates a transformation mapper that makes it possible to transform new unseen data, albeit more slowly than some other transformers. Although this simple approach is used in the present study, the use of parametric UMAP, which introduces a deep neural network during the optimization procedure for learning the parametric relationship between the original data, $\mathbf{X}$, and the final embedding, $\mathbf{Y}$, can also be considered.

6. Data Classification: Extreme Learning Machine (ELM)

An ELM is a single hidden layer feedback neural network proposed by Huang et al. [58]. The input weight and the biases of the hidden layer nodes of the ELM are generated at random, and the output weight can be determined automatically by the input data without repeated adjustments. Unlike traditional neural networks, such as a BP neural network, the ELM has a simpler structure and higher training speed, and its approximation capability is comparable to that of traditional neural networks [59].

A detailed description of the ELM algorithm is beyond the scope of this study. For more information on it, please refer to [60]. However, to present the background and motivation for the proposed methodology, a summary of the ELM method is provided in Appendix B. The recap is based on [59].

A brief description of the main ELM parameters used in this study is provided in Table 5. It is important to note that, in this study, all undefined parameters are set to their default values.

Table 5. Description of the key hyperparameters of the ELM method.

Parameter	Value	Default	Description
n_hidden	$\mathbb{N}$	20	number of nodes in hidden layer ($\tilde{d}$)
activation_func	string [a]	tanh	activation function of the hidden layer (g)

[a] tanh, sine, tribas, inv_tribase, sigmoide, hardlim, softlim, gaussian, multiquadric, inv_multiquadrice, callable function. tanh: Hyperbolic tangent. triangular activation function $g(x) = \min\{1, \max\{0, 1 - |x|\}\}$. hardlim: Hard limit activation function (Heaviside step function $g(x) = 1$ if $x > 0$ and 0 otherwise).

7. Validation/Verification: Train/Test Data Split and Performance Measures

Assessing the performance of the machine learning tool on unseen data is crucial. Here, a two-stage approach is proposed on the basis of the initial validation step followed by a final verification on the completely unseen data. Therefore, this study split the data into the training and test sets with a 3:1 ratio, where 75% of the data were used for the training/validation of the model and the 25% test data, which remained hidden during the model training and model performance evaluation stage, were used for the final performance verification of the presented approach. Moreover, in the training and validation step, a 5-fold CV procedure was used to avoid an overfitting. That is, the training dataset containing 75% of the data was, in turn, split into five folds (or subsets) containing 15% of the data to generate five different validation models (each one taking the data from four folds as the local training data for the algorithm and the remaining local test data for the performance evaluation) (see Figure 2). For a fixed number of extracted UMAP features d and fixed values of the ELM hyperparameters, the UMAP method was applied to the entire training dataset, and the five ELM classifiers were trained using the local training data. These classifiers were validated using the associated local test dataset.

To define the performance/accuracy measure during the validation step, because we are dealing with a multiclass classification problem where each sample from the local test dataset has a known class label, we incorporate the correctness/incorrectness of the predicted classes into a multiclass confusion matrix (see [15]). Specifically, for a problem with l classes or labels $\{C_1, C_2, \ldots, C_l\}$, the number of samples belonging to class C_i that have been classified as belonging to class C_j, herein denoted by C_{ij}, can be summarized in the confusion matrix, as shown in Table 6, for the specific case of $l = 3$.

Table 6. Multi-class confusion matrix. The colored cells correspond to the true positives (green), true negatives (cyan), false negatives (orange), and false positives (magenta) associated with the C_3 class.

		Predicted Class		
		Class C_1	**Class C_2**	**Class C_3**
Actual Class	Class C_1	C_{11}	C_{12}	C_{13}
	Class C_2	C_{21}	C_{22}	C_{23}
	Class C_3	C_{31}	C_{32}	C_{33}

Then, for a given class C_i, denote by tp_i, tn_i, fn_i, and fp_i, the number of samples that, with respect to class C_i, are true positives, true negatives, false negatives, and false positives, respectively, namely:

$$\mathsf{tp}_i = C_{ii}, \qquad \mathsf{tn}_i = \sum_{\substack{k=1 \\ k\neq i}}^{l} \sum_{\substack{j=1 \\ j\neq i}}^{l} C_{kj},$$

$$\mathsf{fn}_i = \sum_{\substack{j=1 \\ j\neq i}}^{l} C_{ij}, \qquad \mathsf{fp}_i = \sum_{\substack{k=1 \\ k\neq i}}^{l} C_{ki}.$$

This allows for introducing the following average performance measures:

$$\text{average accuracy} = \frac{1}{l}\sum_{i=1}^{l}\frac{\mathsf{tp}_i + \mathsf{tn}_i}{\mathsf{tp}_i + \mathsf{tn}_i + \mathsf{fn}_i + \mathsf{fp}_i}, \tag{2}$$

$$\text{average precision} = \frac{1}{l}\sum_{i=1}^{l}\frac{\mathsf{tp}_i}{\mathsf{tp}_i + \mathsf{fp}_i}, \tag{3}$$

$$\text{average recall} = \frac{1}{l}\sum_{i=1}^{l}\frac{\mathsf{tp}_i}{\mathsf{tp}_i + \mathsf{fn}_i}, \tag{4}$$

$$\text{average specificity} = \frac{1}{l}\sum_{i=1}^{l}\frac{\mathsf{tn}_i}{\mathsf{tn}_i + \mathsf{fp}_i}, \tag{5}$$

$$\text{F}_1\text{-score} = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \tag{6}$$

which consider the imbalance problem presented in the EN dataset #2 and the existing differences between the samples of the wine classes (see [61] for more details).

As shown in Figure 2, the 5-fold CV produces five different confusion matrices providing five performance measures that are averaged to obtain the final validation performance measure. However, despite the performance measures being nonlinear, the common practice of computing the total performance measure by first adding the five confusion matrices associated with each fold and then computing the performance of the total added confusion matrix is used.

8. Experimental Results and Discussion

The results of the developed manifold learning classification methodology for ENs are shown in this section for the three described datasets. These results were obtained using the available online implementations of the UMAP [55] and ELM [62] methods.

8.1. Data Transformation and Unfolding

8.1.1. Dataset #1

The EN used to acquire dataset #1 is composed of 16 sensors. Specifically, four Figaro sensors of each of the following types were used: TGS2600, TGS2602, TGS2610, and TGS2620. After conducting the experiments, acquiring the signals by each sensor, and selecting the corresponding eight features per sensor, we applied the min-max normalization. The data were then unfolded to obtain the two-dimensional data matrix $\mathbf{X} \in \mathbb{R}^{3600\times128}$, where each row contains the information regarding all 16 sensors of a particular experiment (see Section 4.4).

Figure 6 shows the 128 feature vectors for a particular experiment before and after applying the min-max normalization. As shown in the figure, before the normalization, the non-normalized steady-state feature, ΔR^{ik} (first feature per sensor), stands out among the other features, i.e., both the normalized steady-state feature (second feature per sensor) and the dynamic features (features 3–8). After the values are scaled to lie between -1 and 1, the data are comparable and ready to be used for classification.

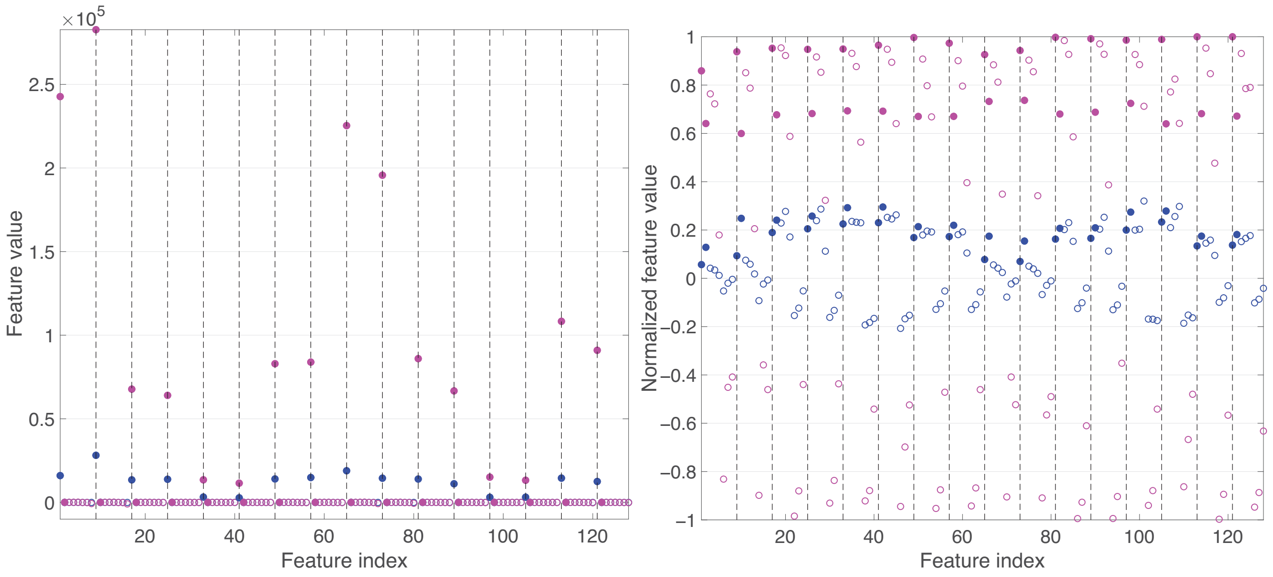

Figure 6. Two unfolded samples in dataset #1 (ethanol compound in blue and ethylene compound in magenta). Initial feature values, i.e., 8 features per 16 sensors (**left**) and min-max normalized values (**right**). The steady-state features are highlighted by solid circles.

8.1.2. Dataset #2

As mentioned in Section 4.2, in this study, all of the sensor values of the full time profile were maintained. Figures 7 and 8 show the data collected in matrix $\mathbf{X} \in \mathbb{R}^{235 \times 19{,}980}$ before and after applying MCUGS normalization. Recall that the non-preprocessed data contain the responses in terms of resistivities over time for each sensor (the values per sensor are placed one after another, with a total of 6 blocks $\times$ 3300 values per sensor). As can be seen in the figure, there are clear differences in the magnitudes associated with each of the six sensors; specifically, the values of the fourth sensor are significantly larger. These differences can be eliminated by applying the MCUGS method. The resulting MCUGS normalized data have zero mean (per feature) and both a unitary block and a global deviation.

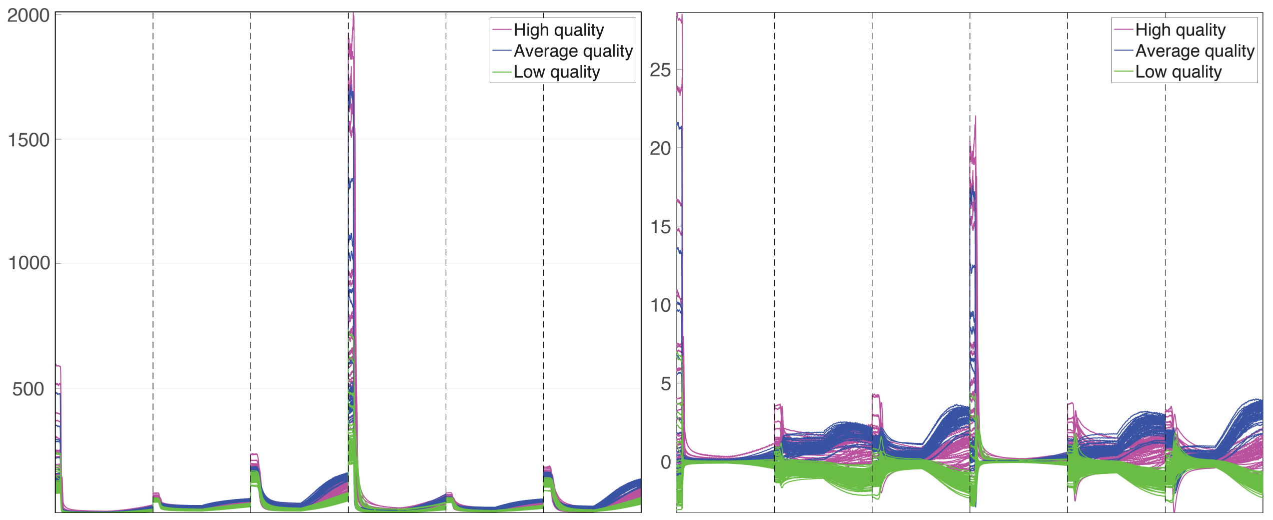

Figure 7. Unfolded data in dataset #2. Initial values (**left**) and MCUGS signal (**right**). All experiments shown are superposed (3330 values per 6 sensor signals for 235 experiments).

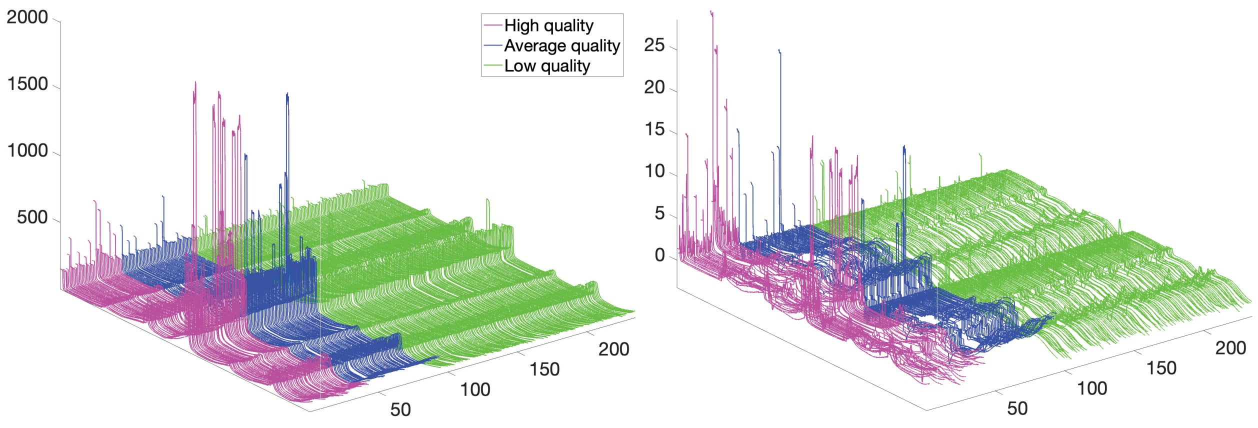

Figure 8. Three-dimensional plot of the unfolded data in dataset #2. Initial values (**left**) and MCUGS signal (**right**). All experiments are shown superposed (3330 values per 6 sensor signals for 235 experiments).

8.1.3. Dataset #3

Figure 9 shows the data collected in matrix $\mathbf{X} \in \mathbb{R}^{309\times 10{,}000}$ before and after applying MCUGS normalization. The non-preprocessed data contain the measures for each sensor (5000 values for 2 sensors) for all 309 experiments. It can be seen in the figure that, in the MCUGS signals, the differences between the three classes are accentuated and, therefore, are better suited for classification purposes.

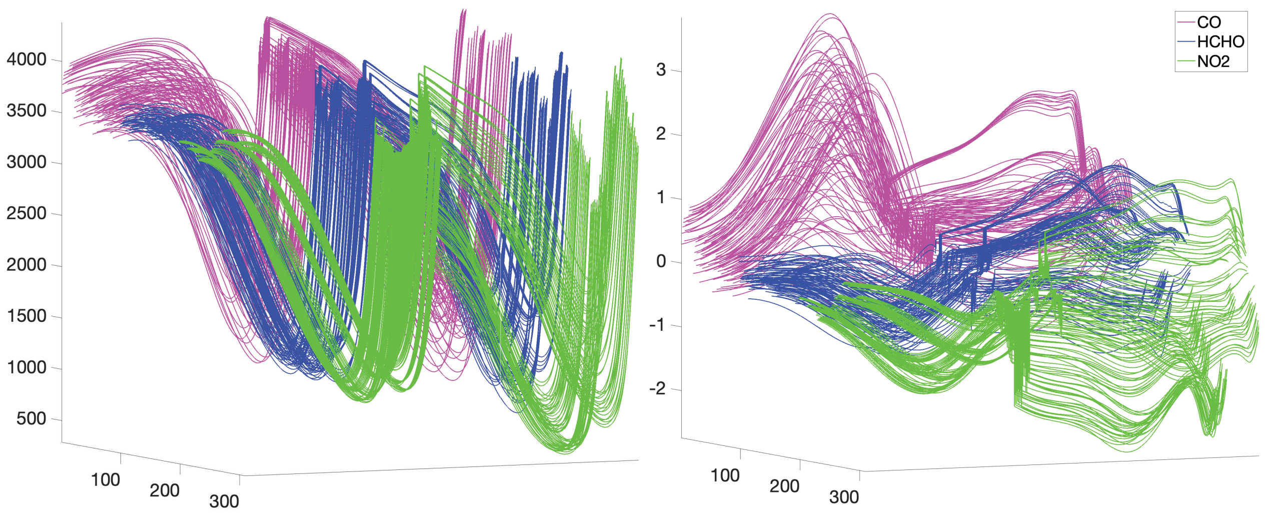

Figure 9. Unfolded data in dataset #3. Initial values (**left**) and MCUGS signal (**right**). All experiments are shown superposed (5000 values per 2 sensor signals for 309 experiments).

8.2. Validation Step: Tuning the UMAP/ELM Parameters

As mentioned in Section 3, a 5-fold CV using a training set containing 75% of the total data was applied to properly tune the number of extracted features of the UMAP method and the hyperparameters of the ELM classifier using a GridSearch algorithm (see Figure 2). A standard method used to find the optimal parameters applies a simple grid search combined with a CV to evaluate the accuracy of the model for each set of candidate hyperparameters. However, a simultaneous exhaustive grid search within the entire grid of the joined UMAP and ELM hyperparameters is prohibitively expensive because we have numerous parameters to be tuned (see Tables 4 and 5). Therefore, a simplified strategy is used. First, the set of hyperparameters to be tuned is decreased by fixing some of the hyperparameters to their default values (the UMAP metric (d) is set to Euclidean, n_epochs is set to none, and the spread is set to 1), and therefore, only five parameters have to be tuned: The UMAP n_neighbors k, the UMAP min_dist, the UMAP n_components d, the number of hidden layer nodes of the ELM $\tilde{d}$, and the ELM activation function g. Then, the five-dimensional space is separated into five separate one-dimensional spaces, and sequential one-dimensional grid searches are carried out, while keeping the values of the other hyperparameters fixed. Although this approach is generally not optimal, because the hyperparameters are typically interdependent, in the current scenario it provides excellent classification results and the use of more involved techniques is not considered (see [63]). The optimal one-dimensional searches are conducted using the GridSearchCV function of scikit learn [64], where the accuracy of the model for each set of parameters is measured using the average accuracy (see Equation (2)).

Table 7 presents the sequential GridSearch conducted for datasets #1 and #3 along with the final selected parameters. Since the results obtained are shown for the three selected activation functions (tanh, tribas, and hardlim), the GridSearch strategy is not applied to determine the optimal ELM activation function. To determine the other parameters, the ELM activation function is set to its default (tanh).

Table 7. One-dimensional sequential GridSearch of the UMAP and ELM hyperparameters (the other parameters are set to their default: Metric (d) is set to the Euclidean, n_epochs is set to none, and the spread is set to 1) for datasets #1 and #3. The range for the one-dimensional searches is highlighted in bold.

	Dataset #1					**Dataset #3**				
	k	**min_dist**	d	$\tilde{d}$	g	k	**min_dist**	d	$\tilde{d}$	g
UMAP n_neighbors k	$[\mathbf{6}, \mathbf{55}]$	0.5	8	100	tanh	$[\mathbf{6}, \mathbf{55}]$	0.5	8	100	tanh
UMAP min_dist	16	$[\mathbf{0.1}, \mathbf{0.9}]$	8	100	tanh	6	$[\mathbf{0.1}, \mathbf{0.9}]$	8	100	tanh
UMAP n_components d	16	0.1	$[\mathbf{2}, \mathbf{20}]$	100	tanh	6	0.5	$[\mathbf{2}, \mathbf{20}]$	100	tanh
ELM hidden layer nodes $\tilde{d}$	16	0.1	8	$[\mathbf{10}, \mathbf{200}]$	tanh	6	0.5	8	$[\mathbf{10}, \mathbf{200}]$	tanh
Final Values	16	0.1	8	60	tanh	6	0.5	8	50	tanh

The results obtained for dataset #2 are not detailed because a perfect classification was obtained for all the tested parameters. The final parameters for dataset #2 were set to n_neighbors $= k = 16$, min_dist $= 0.1$, n_components $= d = 8$, and $\tilde{d} = 100$.

As can be seen in Table 7, the first parameter that has been tuned is the number of neighbors k of the UMAP neighboring graph when the remaining parameters are fixed. Figure 10 shows the effectivity index associated to the average accuracy, defined as $\rho = 1 -$ average accuracy, obtained when varying k from 6 to 55 for datasets #1 and #3. Since a perfect classification corresponds to an average unitary accuracy, the best results are obtained when ρ is close to zero. As can be seen in the figure, a jump in the quality of the classifier is obtained when using 16 neighbors for dataset #1 and 6, 34, or 35 neighbors for dataset #3, where the average accuracies obtained are 0.99901 and 0.99711, respectively. It can also be seen that, for parameter values larger than 40, an increase in the accuracy can be obtained (even yielding perfect classification results); however, considering the trade-off between computational cost and accuracy, and because the accuracy is also improved by tuning the remaining parameters, settings of $k = 16$ and 6 are the best option. The results for dataset #2 are not shown because all values of k yielded a perfect classification ($\rho = 0$). In this case, $k = 16$ is considered.

After the optimal number of neighbors k is fixed, GridSearch is applied with respect to the min_dist parameter that controls the minimum distance between the points in the low-dimensional representation (for the same fixed remaining parameters). Figure 11 shows the effectivity index obtained when varying min_dist within the interval $[0.1, 0.9]$ for datasets #1 and #3.

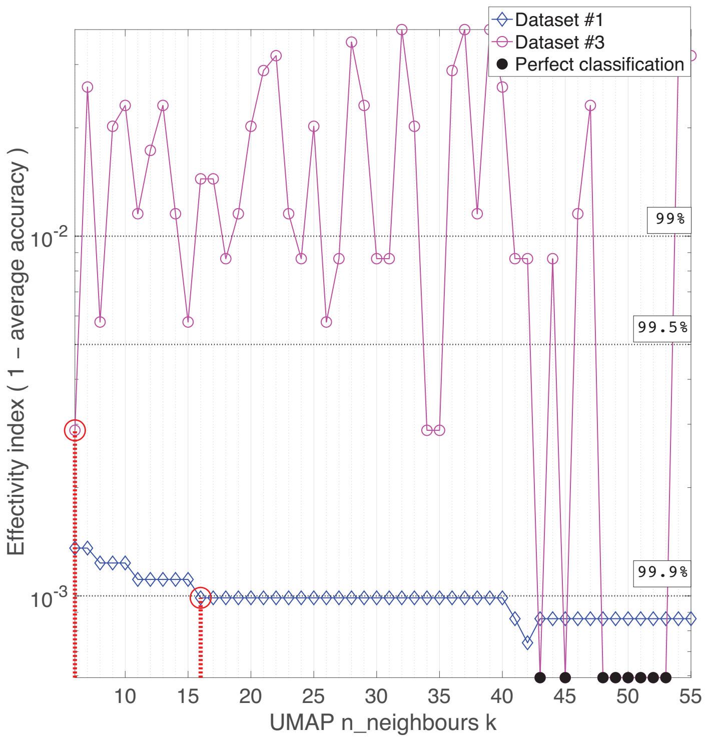

Figure 10. Average accuracy vs. the number of neighbors in the UMAP graph k for datasets #1 and #3 (semi-logarithmic plot in the vertical axis). The vertical axis represents the effectivity index, i.e., one minus the average accuracy. Perfect classifications ($\rho = 0$) are marked with solid black circles. The optimal selected parameters are highlighted in red. The accuracy range for dataset #1 is from 99.864% to 99.925% and that for dataset #3 is from 96.248% to 100%.

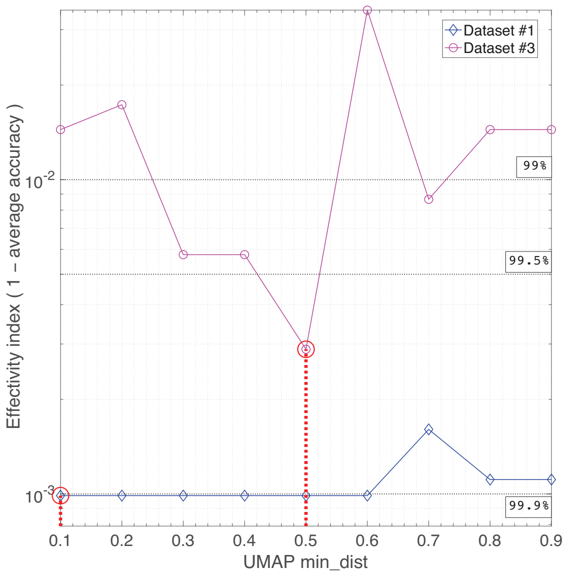

Figure 11. Average accuracy versus the minimum distance between points in the low-dimensional UMAP representation min_dist for datasets #1 and #3 (semi-logarithmic plot in the vertical axis). The vertical axis represents the effectivity index. The optimal selected parameters are highlighted in red. The accuracy range for dataset #1 is from 99.839% to 99.901% and for dataset #3 is from 96.536% to 99.711%.

Again, the results for dataset #2 are not shown because a perfect classification was obtained for all values, taking min_dist $= 0.1$. Regarding dataset #1, the highest average accuracy of 0.99901 is obtained for values of min_dist of up to 0.6, and the default value of 0.1 is maintained. It is worth noting that the results in this case are not overly sensitive to changes in the parameter value. Finally, for dataset #3, changing the initially prescribed value of min_dist = 0.5 provides worse results, and therefore, the value is maintained.

The final hyperparameter of the UMAP method to be tuned is n_components, i.e., the dimension of the low-dimensional space, d (number of retained features used for classification). It is worth noting that this parameter, which is the dimension of the feature vector inserted into the ELM classifier, is key to the accuracy of the final classification results (see [65]). Figure 12 shows the values of the effectivity index when varying the n_components within the interval $[2, 20]$ for datasets #1 and #3. For dataset #3, a jump in quality is obtained for n_components, = 8. Regarding dataset #1, excellent results are obtained for nearly all parameter values and for a value of $d = 8$. As can be seen in the figure, the supervised version of the UMAP method obtains excellent accuracies for the training dataset even for an extremely low number of retained features (an accuracy of 0.99901 is obtained even for $d = 2$ in dataset #1, and an accuracy of 0.98845 is obtained for $d = 4$ in dataset #3, instead of 0.99711 in the case of $d = 8$). However, because the classifier is needed to classify new unseen data, the value of $d = 8$ is retained in both cases as a tradeoff between computation cost and accuracy.

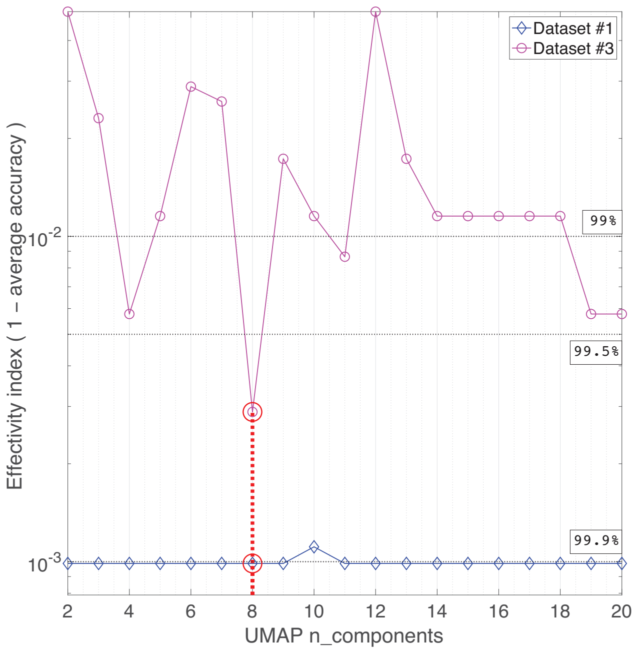

Figure 12. Average accuracy vs. the dimension of the UMAP low-dimensional space d n_components for datasets #1 and #3 (semi-logarithmic plot in the vertical axis). The vertical axis represents the effectivity index. The optimal selected parameters are highlighted in red. The accuracy range for dataset #1 is from 99.888% to 99.901% and that for dataset #3 is from 95.093% to 99.711%.

Finally, a GridSearch was conducted to set the number of hidden layer nodes $\tilde{d}$ of the ELM method. It is well known that the number of hidden nodes is a key factor for achieving a good performance of the ELM method. The optimal value of the parameter usually depends on the size of the training set, n, the number of input features, d, and the number of output classes, L. Figure 13 shows the sensitivity analysis of the accuracy of the combined UMAP-ELM strategy with respect to the number of hidden nodes of the ELM classifier. The selected parameters are $\tilde{d} = 60$ and 50 for datasets #1 and #3, respectively, with associated final accuracies of 99.901% and 99.711%, respectively. As depicted in the figure, the combined approach allows excellent accuracies to be obtained even for an extremely low number of hidden nodes as opposed to the standard large number of hidden nodes required in other existing approaches (see for instance [45]). It is also worth noting that, for both datasets, all reported accuracies are above 90% (excluding the value $\tilde{d} = 1$ for dataset #1, where an accuracy of 88% was reached).

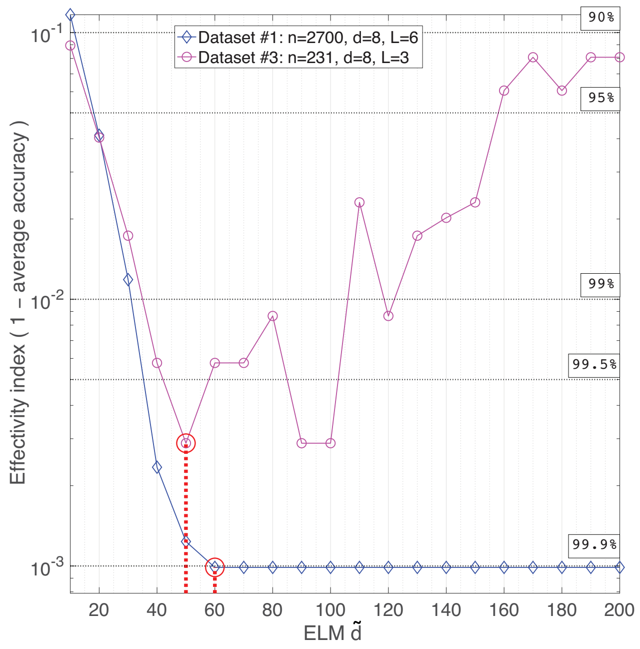

Figure 13. Average accuracy vs. the number of hidden layer nodes ELM $\tilde{d}$ for datasets #1 and #3 in the training set (semi-logarithmic plot in the vertical axis). The vertical axis represents the effectivity index. The optimal selected parameters are highlighted in red. The accuracy range for dataset #1 is from 88.333% to 99.901% and that for dataset #3 is from 91.053% to 99.711%.

Although it has been reported that the selection of the UMAP and ELM methods has a significant impact on the performance, it is worth noting that the combined approach presented here is extremely robust. Figures 10–13 show the results of the sensitivity analysis of the combined UMAP-ELM strategy proposed, showing the accuracies with respect to the variation of the parameters. It can be seen in the figures that, for the first three parameters studied, the accuracies are all above 95%, and regarding the number of hidden nodes, all values ranging from 20 to 150 also ensure an accuracy of 95%.

8.3. Classification Results

This section shows the validation and verification performance measures obtained for the three considered datasets (see Figure 2).

8.3.1. Dataset #1

This dataset contains 3600 samples from six different analytes, where each sample has 128 initial features. After normalizing and unfolding all of the data, we applied a 3:1 partition of the total samples: 2700 samples were selected for the initial training (450 for each gas) and 900 samples were kept for the final testing (150 for each gas). The training samples were used in the supervised UMAP manifold learning algorithm to reduce the number of features to be used for classification to only $d = 8$ (see Table 7 for the details of the UMAP hyperparameters used).

Figure 14 illustrates the first three features extracted using the UMAP method in dataset #1 for the training samples and the test samples, separately. As can be seen in the figure, the supervised UMAP allows the successful clustering of most samples of the training set, even though a clear visual separation cannot be achieved. A careful look at the eight features provided by the UMAP method provides similar results to those shown in Figure 14. Most of the samples seem to be clustered however, some remaining overlapping occurs with no clear visual separation. Therefore, the use of a machine learning classifier algorithm is crucial to be able to properly classify all of the samples. When the learned UMAP mapper is used to extract the features of the test dataset, it can be seen that most new points also follow the same pattern, even though the samples do not cluster as clearly.

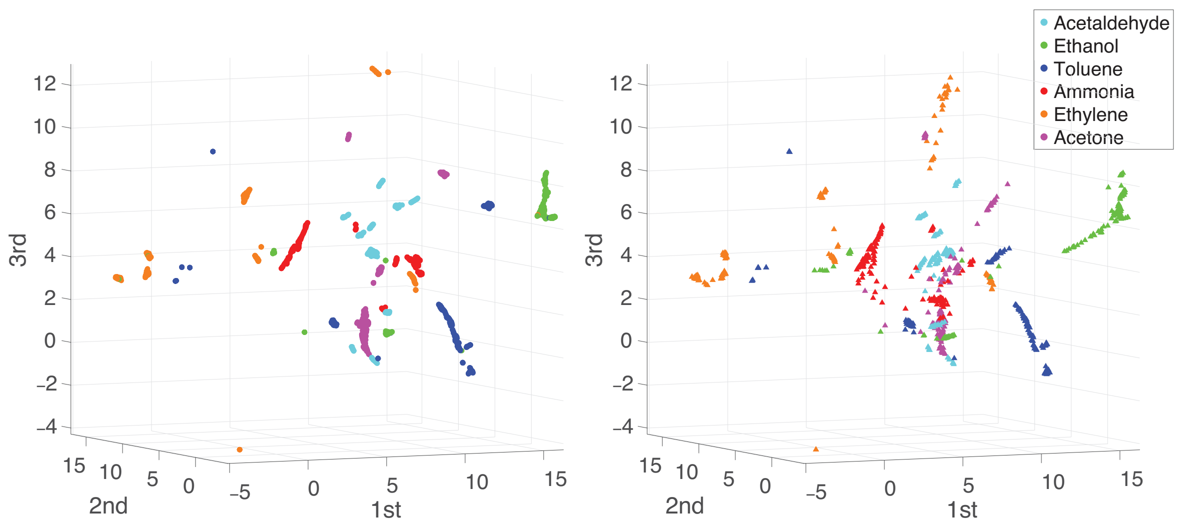

Figure 14. Scatter plot of the first three UMAP features of the data in dataset #1 (the training set is shown on the **left** and the test set is shown on the **right**). The six different analytes are shown in different colors.

The classification results obtained by using a 5-fold CV on the training dataset with the hyperparameters reported in Table 7 are summarized in the total added confusion matrix shown in Table 8. This table also shows the classification results on the test dataset for the same value of the parameters.

As can be seen in the table, all the training samples from acetaldehyde, ammonia, and acetone are perfectly recognized and only eight samples from among the 2700 are misclassified (two ethylene samples, three ethanol samples, and three toluene samples), yielding an accuracy of 99.90%. Regarding the classification of the 900 samples of the test data, in this case, the toluene and ethylene samples are perfectly recognized and only 17 samples are misclassified, yielding an accuracy of 99.37%.

Tables 9 and 10 show all the performance metrics associated with both the validation and verification steps, and the effect of the choice of the activation function of the ELM classifier.

As expected, because the parameters of the methods were trained using these data, excellent performance metrics were obtained when the combined UMAP-ELM classifier was used in the training set for the hyperbolic tangent (tanh) activation function. However, it can be seen that the same parameters also provided good results for the other two activation functions (tribas and hardlim). In addition, the accuracies obtained in the validation step with the unseen test dataset were excellent.

Table 8. Confusion matrix for the UMAP-ELM training and test sets for dataset #1 (for the ELM activation function tanh).

		Predicted Class (Train Set)						Predicted Class (Test Set)					
		Acet.	**Etha.**	**Tolu.**	**Ammo.**	**Ethy.**	**Acet.**	**Acet.**	**Etha.**	**Tolu.**	**Ammo.**	**Ethy.**	**Acet.**
Actual Class	Acetaldehyde	450	0	0	0	0	0	148	0	0	2	0	0
	Ethanol	0	447	1	0	2	0	0	144	3	0	1	2
	Toluene	0	3	447	0	0	0	0	0	150	0	0	0
	Ammonia	0	0	0	450	0	0	0	3	0	147	0	0
	Ethylene	0	1	0	1	448	0	0	0	0	0	150	0
	Acetone	0	0	0	0	0	450	1	4	0	1	0	144

Table 9. Validation (train) classification performance measures of the ELM classifier for three different transfer functions of the ELM algorithm in dataset #1.

	Validation Performance Measures (Training Set)				
	Accuracy	**Precision**	**Specificity**	**Recall**	**F1 Score**
ELM tanh	0.99901	0.99704	0.99941	0.99704	0.99704
ELM tribas	0.95333	0.88147	0.97200	0.86000	0.87060
ELM hardlim	0.99864	0.99593	0.99919	0.99593	0.99593

Table 10. Verification (test) classification performance measures of the ELM classifier for three different transfer functions of the ELM algorithm in dataset #1.

	Verification Performance Measures (Test Set)				
	Accuracy	**Precision**	**Specificity**	**Recall**	**F1 Score**
ELM tanh	0.99370	0.98117	0.99622	0.98111	0.98114
ELM tribas	0.95741	0.87711	0.97444	0.87222	0.87466
ELM hardlim	0.99333	0.98029	0.99600	0.98000	0.98015

8.3.2. Dataset #2

This dataset contains 235 samples of wine with three different qualities, where each initial sample has 19, 980 features. After the MCUGS was applied and all of the data were unfolded, a 3:1 partition of the total samples was achieved: 176 samples were selected for the initial training (32, 38, and 106 for the high-, average-, and low-quality samples, respectively) and 59 samples were kept for the final testing (11, 13, and 35 for the high-, average-, and low-quality samples, respectively). As mentioned in Section 8.2, the combined UMAP-ELM strategy provides a perfect classification on the training set for all the selected parameters when using the tanh activation function. Note that, although some imbalance in the data is present, this does not indicate a poor predictive performance of the model for the minority classes.

Figure 15 illustrates the first three features extracted using the UMAP method in dataset #2 for both the training and test samples. As can be seen in the figure, the capabilities of the UMAP method for clustering the data are evident owing to the perfect separation of the wine-quality classes as evidenced for both the training and test datasets. This clustering explains the perfect classification results obtained using the ELM classifier.

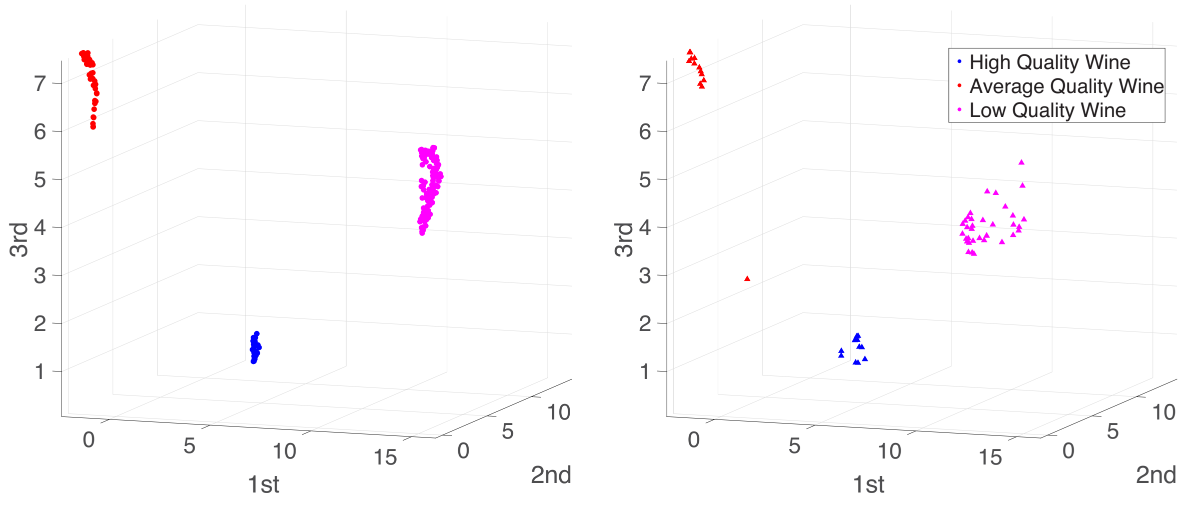

Figure 15. Scatter plot of the first three UMAP features of the data in dataset #2 (the training set is on the left and the test set is on the right). The three qualities of the wine are shown in different colors.

The performance measures obtained for dataset #2 using three different activation functions on the ELM method are given in Tables 11 and 12. As mentioned, the tanh activation function allows to perfectly classify both the training and test datasets. The results for the tribas and hardlim functions were also excellent, yielding only some minor misclassifications (see the associated confusion matrices shown in Table 13).

Table 11. Validation (train) classification performance measures of the ELM classifier for three different transfer functions of the ELM algorithm in dataset #2.

	Validation Performance Measures (Training Set)				
	Accuracy	**Precision**	**Specificity**	**Recall**	**F1 Score**
ELM tanh	1	1	1	1	1
ELM tribas	0.96970	0.93333	0.98148	0.96922	0.95094
ELM hardlim	1	1	1	1	1

Table 12. Verification (test) classification performance measures of the ELM classifier for three different transfer functions of the ELM algorithm in dataset #2.

	Verification Performance Measures (Test Set)				
	Accuracy	**Precision**	**Specificity**	**Recall**	**F1 Score**
ELM tanh	1	1	1	1	1
ELM tribas	1	1	1	1	1
ELM hardlim	0.98870	0.97222	0.99306	0.97436	0.97329

Table 13. Confusion matrices for the UMAP-ELM for dataset #2. Training set using the tribas activation function (left) and test set using the hardlim activation function (right).

		Predicted Class (Train Dataset & Tribas)			Predicted Class (Test Dataset & Hardlim)		
		HQ	**AQ**	**LQ**	**HQ**	**AQ**	**LQ**
Actual Class	High-Quality Wine	32	0	0	11	0	0
	Average-Quality Wine	1	37	0	1	12	0
	Low-Quality Wine	7	0	99	0	0	35

The current combined UMAP-ELM approach improves the classification results presented in the original study [47]. This study also involves a feature extraction method along with the use of two different classifiers (an SVM and a deep MLP neural network) and presents the results of a 5-fold validation in the training and verification test stages. The reported average accuracies for the test stages were 97.34% for the conventional SVM approach, where 69 features were selected for the classification, and 97.68% for the improved approach, where 300 features were selected for the classification. In this case, the use of the combined UMAP-ELM methods allows perfect classifications to be obtained, therefore improving the aforementioned results.

8.3.3. Dataset #3

This dataset consists of 309 nearly balanced samples (92, 100, and 112 samples of carbon monoxide (CO), formaldehyde (HCHO), and nitrogen oxide (NO_2), respectively) split into 231 training samples (72, 75, and 84, respectively) and 78 test samples (24, 25, and 29, respectively). Each sample contained 10,000 initial features that were reduced to 8 using the supervised UMAP method. Figure 16 illustrates the first three features extracted using the UMAP method in dataset #3 for both the training and test samples. As can be seen in the figure, the supervised UMAP seems to properly cluster nearly all the samples of the training dataset however, when the test data were projected, a slight overlapping occurred. The small ambiguities present in the separation of the training dataset were amplified when projecting the test dataset. It is worth noting that, in this case, the characteristics of the features did not allow such an easy clustering as in dataset #2 and that the supervised UMAP method only had 231 samples to train however, in dataset #1, where the global structure was quite complex, 2700 samples were used for training.

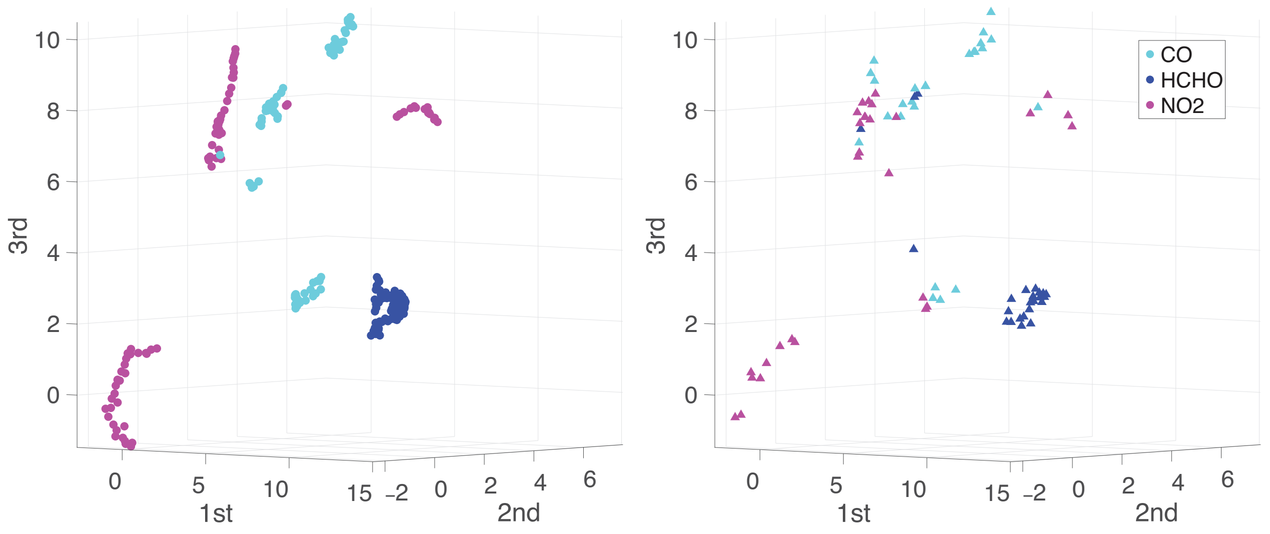

Figure 16. Scatter plot of the first three UMAP features of the data in dataset #3 (training set on the **left** and test set on the **right**). The three different classes are shown in different colors.

The classification results obtained using the hyperparameters reported in Table 7 are shown in Table 14 for both the verification and validation steps.

Table 14. Confusion matrix for the UMAP-ELM training and test sets for dataset #3 (for the ELM activation function tanh).

		Predicted Class (Train Set)			Predicted Class (Test Set)		
		CO	HCHO	NO_2	CO	HCHO	NO_2
Actual Class	carbon monoxide (CO)	72	0	0	20	2	2
	formaldehyde (HCHO)	0	74	1	0	20	5
	nitrogen oxide (NO_2)	0	0	84	0	4	25

In the verification step, the combined UMAP-ELM classifier reached an accuracy of 99.71% with only one misclassified HCHO sample. However, when used in the test set, the accuracy decreased to 88.89%. As mentioned above, although this is still a good classification result, the decrease in accuracy regarding the results obtained in datasets #1 and #3 could be due to the fact that very few samples were available for training. Although these samples were sufficient for clustering the training set using the supervised UMAP method, when the test data were projected, the main features did not allow for accurately separating the data and the ELM classifier introduced some classification errors.

Finally, Tables 15 and 16 show all the performance verification and validation measures for the three activation functions used. As with the previous datasets, excellent accuracies were obtained in all cases, with the tribas option being slightly worst. Among the three cases, the same decrease in accuracy was obtained for the test dataset.

Table 15. Validation (training) classification performance measures of the ELM classifier for three different transfer functions of the ELM algorithm in dataset #3.

	Validation Performance Measures (Training Set)				
	Accuracy	Precision	Specificity	Recall	F1 Score
ELM tanh	0.99711	0.99608	0.99773	0.99556	0.99582
ELM tribas	0.90476	0.86301	0.92805	0.85802	0.86051
ELM hardlim	0.99134	0.98721	0.99346	0.98762	0.98742

Table 16. Verification (test) classification performance measures of the ELM classifier for three different transfer functions of the ELM algorithm in dataset #3.

	Verification Performance Measures (Test Set)				
	Accuracy	**Precision**	**Specificity**	**Recall**	**F1 Score**
ELM tanh	0.88889	0.85016	0.91465	0.83180	0.84088
ELM tribas	0.82051	0.72999	0.86566	0.73433	0.73215
ELM hardlim	0.88034	0.83831	0.90836	0.82031	0.82921

The current results can be partially compared with those reported in the original study [49], where a subsampling strategy to extract 10, 20, 30, and 40 subsampled features from the original 10,000 features was used instead of the UMAP method. Then, a full 2D GridSearch was applied to find the optimal parameters of the regularized ELM classifier. Specifically, they tuned together the number of hidden nodes along with the penalty regularization parameter. All the data were used for training, and therefore, only the performance validation measures were obtained. Perfect per-class classifications were reported for HCHO and CO, whereas a 99.92% accuracy was reported for NO_2. The per-class average accuracies (computed as shown in Equation (2) $(\mathtt{tp}_i + \mathtt{tn}_i)/(\mathtt{tp}_i + \mathtt{tn}_i + \mathtt{fn}_i + \mathtt{fp}_i)$) can also be obtained by employing our approach, as shown in Table 14 (training set); we obtain an average accuracy of 100% for CO and 99.56% (=230/231 $\times$ 100) for HCHO and NO_2, where the total number of samples is 231 instead of 309. Three aspects are worth pointing out. First, as can be seen in Figure 10 and Table 17, a perfect classification for the training set could be obtained using the newly combined UMAP-ELM approach, for instance, taking $k = 43$. However, in the present approach, overfitting is avoided by adding the verification step. Second, reference [49] does not report how the average per-class accuracies are computed. Using the aforementioned standard per-class accuracy, a classification with only one error in one of the five folds (four retained for training and one for testing) yields an approximate per-class accuracy of $308/309 = 0.9967$, from which it is difficult to reproduce the 99.92% accuracy reported for comparison with the present results. Finally, it is worth noting that, in [49], despite the larger number of extracted features, the number of hidden nodes used was kept to $\tilde{d} = 3$. In future studies, the use of the regularized ELM method instead of the standard non-regularized ELM method can be considered to determine whether the use of a penalty regularization parameter allows for decreasing the number of hidden nodes without introducing an overfitting during the validation step.

Table 17. Additional classification performance measures in the training set (validation step) using three different transfer functions of the ELM algorithm in dataset #3 (corresponding to $k = 50$, $\text{min\_dist} = 0.5, d = 8, \tilde{d} = 50$).

	Verification Performance Measures (Test Set)				
	Accuracy	**Precision**	**Specificity**	**Recall**	**F1 Score**
ELM tanh	1	1	1	1	1
ELM tribas	0.98268	0.97382	0.98703	0.97439	0.97410
ELM hardlim	1	1	1	1	1

9. Concluding Remarks

In this study, a data classification methodology for ENs was presented. It involved several stages, namely, data unfolding, data reduction by applying the UMAP algorithm, and machine learning classification using the ELM method, and excellent results were achieved, as evidenced by the high values of the classification performance. Three different EN datasets were used to validate the application of the developed methodology. The first dataset was composed of six different gases, second dataset described three different classes of wine, and third dataset consisted of measurements of three different gases.

The main conclusions of this work are as follows:

- Data transformation and unfolding: A data transformation procedure (feature extraction and $\min - \max$ normalization for dataset #1 and MCUGS for datasets #2 and #3) was conducted to correct the differences in the signal magnitudes acquired by the EN sensor array. In addition, a data unfolding step was employed to arrange the acquired data in a two-dimensional matrix.

- Data reduction: The supervised UMAP algorithm was used as a data reduction method. The capability to maximize the inter-class distance and minimize the intra-class distance was demonstrated. The UMAP data reduction procedure allows for a new representation of the data to be obtained in a low-dimensional space, facilitating the task conducted by the ELM classifier.
- Data classification: The ELM classifier algorithm was used owing to its high training speed and satisfactory behavior in conducting classification tasks. Proof of this is in the high average accuracies obtained on the training and test sets for each of the three datasets tested.
- Validation/verification and hyperparameter tuning: A data splitting procedure was used, which divided 75% of the data as the training set and the remaining 25% as the test set. First, the training procedure performed a 5-fold CV to avoid overfitting and the hyperparameters were tuned using the UMAP and ELM methods. Second, the remaining unseen 25% of the data were used in the verification stage. The results in this test set showed a good behavior after the application of the developed methodology because the UMAP algorithm mapped the new data correctly to the low-dimensional space and the previously trained ELM classifier performed a successful classification.
- Various average classification performance measures were calculated. These measures describe the multi-class classification problem for a single number. High values of these classification performance measures were obtained for the training and test sets in each of the three EN datasets used.
- Two of the three datasets exhibited imbalanced data. The classification results showed high average accuracies, and in some cases, the values reached 100% for the training set.

The overall contributions of this paper can be summarized as follows:

In summary, based on the results and previous conclusions, it is possible to highlight the following elements as differentiating elements compared with other similar works:

The combination of the methods to produce this methodology and its validation with three different datasets.

The use of $\min - \max$ or mean-centered unitary group-scaling (MCUGS) to deal with the different magnitudes of the sensors. This MCUGS normalization method is firstly introduced in the analysis of data from EN sensors, and it is the second time that it has been used in the literature.

The use 5-fold cross-validation (CV) to prevent overfitting, which result in other way to validate the results compared with traditional validations in data analysis.

In future studies, the performance of the methodology will be evaluated using different methods of data normalization and different classification algorithms with an active learning approach.

Author Contributions: Conceptualization, J.X.L.-M., M.A. and D.A.T.; methodology, J.X.L.-M. and N.P.; software, J.X.L.-M. and N.P.; validation, D.A.T., M.A. and F.P.; formal analysis, N.P.; investigation, J.X.L.-M. and N.P.; resources, D.A.T. and F.P.; data curation, J.X.L.-M.; writing—original draft preparation, J.X.L.-M. and N.P.; writing—review and editing, D.A.T., M.A. and F.P.; visualization, N.P. and F.P.; supervision, D.A.T., M.A. and F.P.; project administration, F.P.; funding acquisition, D.A.T. and M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded in part with resources from the Fondo de Ciencia, Tecnología e Innovación (FCTeI) del Sistema General de Regalías (SGR) from Colombia. The authors express their gratitude to the Administrative Department of Science, Technology, and Innovation–Colciencias with the grant 779–"Convocatoria para la Formación de Capital Humano de Alto Nivel para el Departamento de Boyacá 2017" for sponsoring the research presented herein. This study has been partially funded by the Spanish Agencia Estatal de Investigación (AEI)-Ministerio de Economía, Industria y Competitividad (MINECO), and the Fondo Europeo de Desarrollo Regional (FEDER) through research projects DPI2017-82930-C2-1-R and PGC2018-097257-B-C33; and by the Generalitat de Catalunya through research projects 2017-SGR-388 and 2017-SGR-1278.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are available through the links reported in the articles [25] (dataset #1), [47,48] (dataset #2), and [49] (dataset #3).

Acknowledgments: J.X. L.-M. thanks Miryam Rincón Joya from the Department of Physics of the National University of Colombia and Leydi Julieta Cardenas Flechas, who is currently a student, for their introduction to the electronic tongue sensor array field of research.

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

Abbreviations

The following abbreviations are used in this manuscript:

UMAP	Uniform manifold approximation and projection
ELM	Extreme learning machine
ENs	Electronic noses
MCUGS	Mean-centered unitary group-scaling
CV	Cross-validation
PCA	Principal component analysis
CO	Carbon monoxide
HCHO	Formaldehyde
NO_2	Nitrogen oxide
SGD	Stochastic gradient descent
tanh	Hyperbolic tangent
tribas	Triangular activation function
hardlim	Hard limit activation function

Appendix A. Practical Computational Description of the UMAP Method

This section presents a practical computational description of the UMAP method in terms of weighted graphs. Let $\mathbf{X} \in \mathbb{R}^{n\times D}$ be a matrix collecting all the samples for a specific dataset, where n denotes the total number of samples and D denotes the total number of features describing the data. The aim is to determine a reduced low-order d-dimensional manifold, $d \ll D$, described by the transformed matrix, $\mathbf{Y} \in \mathbb{R}^{n\times d}$, that retains most of the information of the original matrix, $\mathbf{X}$. In particular, in the present study, the aim is to obtain a transformed matrix, $\mathbf{Y}$, that provides an accurate classifier when used as an input of the machine learning algorithm. The UMAP transformed matrix, $\mathbf{Y}$, is computed in two phases. In the first phase, namely, graph construction, a weighted k-neighbor graph is constructed, and after proper transformations are applied on the edges of the graph, the data, $\mathbf{X}$, are represented by the adjacency matrix of this graph. Then, a low-dimensional representation of the previous graph preserving the desired characteristics of the graph, i.e., a graph layout, is built. Specifically, let $\mathbf{X} \in \mathbb{R}^{n\times D}$ be the input data set:

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)^T, \tag{A1}$$

where $\mathbf{x}_i = \text{row}_i(\mathbf{X})^T \in \mathbb{R}^D$; in addition, consider the metric $\mathsf{d} : \mathbb{R}^D \times \mathbb{R}^D \to [0, +\infty)$.

To fix these ideas, let d be the Euclidean distance, which is defined as:

$$\mathsf{d}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 = \sqrt{\sum_{l=1}^{D} \left((\mathbf{x}_i)_l - (\mathbf{x}_j)_l\right)^2},$$

where $(\mathbf{x}_i)_l$ denotes the lst element entry of vector $\mathbf{x}_i$, which coincides with x_{il}, i.e., the (i,l)th entry of matrix $\mathbf{X}$. Then, given an input hyperparameter, k, for each point $\mathbf{x}_i$, $i = 1, \ldots, n$, we compute the set $\{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_k}\}$ of the k-nearest neighbors of $\mathbf{x}_i$ under the metric d and define the parameters ρ_i and σ_i such that:

$$\rho_i = \min\{\mathsf{d}(\mathbf{x}_i, \mathbf{x}_{i_j}) \mid 1 \le j \le k,\ \mathsf{d}(\mathbf{x}_i, \mathbf{x}_{i_j}) > 0\},$$

and

$$\sum_{j=1}^{k} \exp\left(-\frac{\mathsf{d}(\mathbf{x}_i, \mathbf{x}_{i_j}) - \rho_i}{\sigma_i}\right) = \log_2(k)$$

(see Figure A1). Note that ρ_i is the distance between point $\mathbf{x}_i$ and its nearest neighbor, whereas σ_i is a smoothed normalization factor.

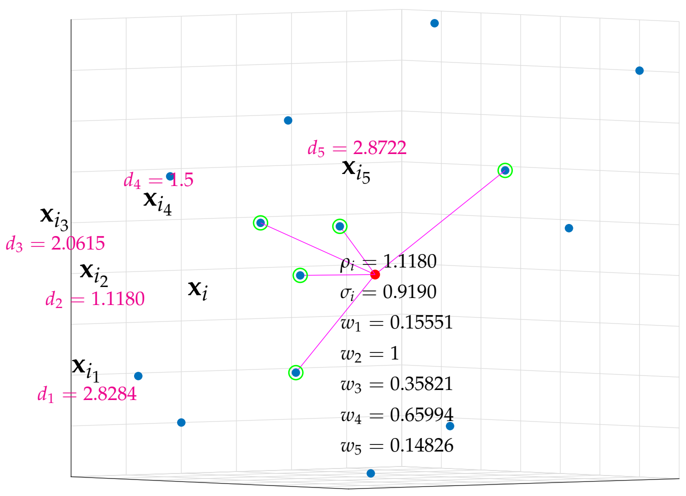

Figure A1. Five nearest neighbors of point $\mathbf{x}_i$ for a particular set of points in $\mathbb{R}^3$, where, for simplicity, $\mathsf{d}_j = \mathsf{d}(\mathbf{x}_i, \mathbf{x}_{i_j})$ and $w_j = w(\mathbf{x}_i, \mathbf{x}_{i_j})$. The metric is the Euclidean distance.

Then, the weighted directed k-neighbor graph is defined as $G = (V, E, w)$, where vertices V are the points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and the set of directed edges is $E = \{(\mathbf{x}_i, \mathbf{x}_{i_j}) \mid 1 \le j \le k,\ 1 \le i \le n\}$ with the corresponding directed weights,

$$p_{\,i|i_j} = w(\mathbf{x}_i, \mathbf{x}_{i_j}) = \exp\left(-\frac{\mathsf{d}(\mathbf{x}_i, \mathbf{x}_{i_j}) - \rho_i}{\sigma_i}\right)$$

(see Figure A2). It is worth noting that the definitions of ρ_i and σ_i ensure that $p_{\,i|i_j} = w(\mathbf{x}_i, \mathbf{x}_{i_j}) \in (0,1]$, at least one edge has a weight of 1, non-existent edges are set to have a weight of 0, and all weights associated with the outcoming edges of every node sum to $\log_2(k)$. Weight $p_{\,i|i_j}$ associated with the edge joining $\mathbf{x}_i$ and $\mathbf{x}_j$ represents the similarity between the high-dimensional points $\mathbf{x}_i$ and $\mathbf{x}_j$, measured using an exponential probability distribution, which can also be seen as the probability that an edge joining $\mathbf{x}_i$ and $\mathbf{x}_j$ exists.

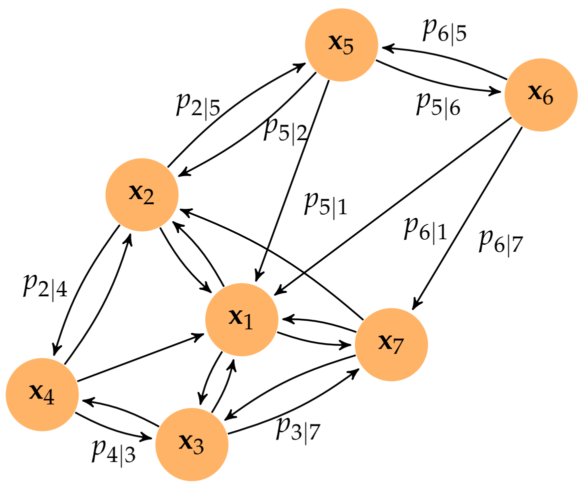

Figure A2. Weighted directed three-neighbor graph for a particular set of points in $\mathbb{R}^2$. For clarity, only some weights $p_{i|i_1} = w(\mathbf{x}_i, \mathbf{x}_{i_1})$ are shown. The metric is the Euclidean distance.

The last step of the graph construction phase is to recover an undirected graph from G. The weights associated with the new undirected edges $(\mathbf{x}_i, \mathbf{x}_j)$ are given by the triangular conorm or the probabilistic sum of the directed weights, namely,

$$p_{ij} = p_{i|j} + p_{j|i} - p_{i|j}p_{j|i}, \tag{A2}$$

which can be interpreted as the probability that at least one of the two directed edges (from $\mathbf{x}_i$ to $\mathbf{x}_j$ and from $\mathbf{x}_j$ to $\mathbf{x}_i$) exists. The adjacency matrix of this undirected graph $\mathbf{P} \in \mathbb{R}^{n\times n}$ containing the weights $p_{ij}, i, j = 1, \ldots, n$ in its corresponding (i, j)th position is the starting point of the UMAP procedure. Note that if $\mathbf{P}_{\text{dir}} \in \mathbb{R}^{n\times n}$ is the adjacency matrix of graph G, then from Equation (A2),

$$\mathbf{P} = \mathbf{P}_{\text{dir}} + \mathbf{P}_{\text{dir}}^T - \mathbf{P}_{\text{dir}} \circ \mathbf{P}_{\text{dir}}^T$$

where '$\circ$' is the Hadamard (or pointwise) matrix product. It is worth noting that, in the initial step, the data matrix, $\mathbf{X} \in \mathbb{R}^{n\times D}$, is associated with a larger, although sparse, $\mathbf{P} \in \mathbb{R}^{n\times n}$ matrix, where the new entries are probability-like measures of the distances between samples.

The aim of the second phase of the UMAP procedure is to compute a low-dimensional representation of the undirected graph preserving the main desired characteristics. That is, the goal is to compute a new transformed matrix, $\mathbf{Y} \in \mathbb{R}^{n\times d}$:

$$\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)^T, \tag{A3}$$

where $\mathbf{y}_i = \text{row}_i(\mathbf{Y})^T \in \mathbb{R}^d$, containing only d features and a new undirected graph with associated weights defined by matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$, which minimizes the cross-entropy between the two graphs, and is defined as follows:

$$\mathcal{C}(\mathbf{P}, \mathbf{Q}) = \sum_{i,j=1}^{n} \left[p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) + (1 - p_{ij}) \log\left(\frac{1 - p_{ij}}{1 - q_{ij}}\right)\right],$$

where p_{ij} and q_{ij} are the weights of the undirected graphs given by the (i, j)th entries of the adjacency matrices $\mathbf{P}$ and $\mathbf{Q}$, (see Figure A3). Since $\mathcal{C}(\mathbf{P}, \mathbf{Q}) = 0$ for $q_{ij} = p_{ij}$, the goal of UMAP is to try to find a low-dimensional representation of the data trying to mimic the similarities p_{ij} in a high-dimensional space.

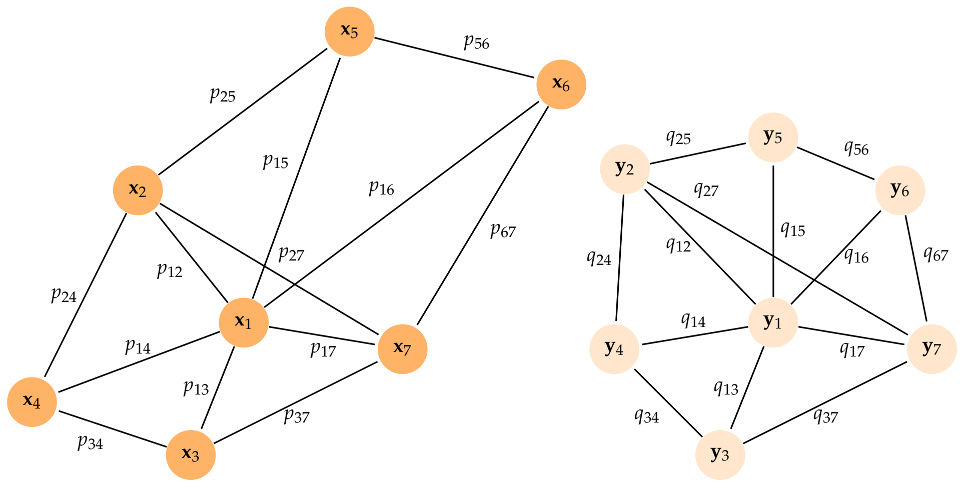

Figure A3. Illustration of a weighted undirected graph for points in $\mathbf{x}_i \in \mathbb{R}^D$ and a low-order representation of the graph where $\mathbf{y}_i \in \mathbb{R}^d$. The weights $p_{ij}(\mathbf{X})$ in the original space are computed from distances measured in the $\mathbb{R}^D$ d-metric of the k-neighbors of each point, whereas the weights $q_{ij}(\mathbf{Y}) = (1 + a\|\mathbf{y}_i - \mathbf{y}_j\|_2^{2b})^{-1}$ are computed from pairwise $\mathbb{R}^d$ Euclidean distances.

Note that the values of p_{ij}, which are computed from $\mathbf{X}$, are fixed and that the weights q_{ij} are computed from the d features of the samples stored in $\mathbf{Y}$. Therefore, after the constant terms in the objective function are removed, the problem at hand translates to finding $\mathbf{Y} \in \mathbb{R}^{n\times d}$ by minimizing:

$$\min_{\tilde{\mathbf{Y}}\in\mathbb{R}^{n\times d}} - \sum_{i,j=1}^{n} \left[p_{ij}\log(q_{ij}(\tilde{\mathbf{Y}})) + (1-p_{ij})\log(1-q_{ij}(\tilde{\mathbf{Y}}))\right]. \tag{A4}$$

To use the stochastic gradient descent (SGD) method in the previous optimization problem, we consider a smooth function describing the similarities between the pairs of points in the low-dimensional space $\mathbb{R}^d$:

$$q_{ij} = (1 + a\|\mathbf{y}_i - \mathbf{y}_j\|_2^{2b})^{-1},$$

where a and b can be user-defined positive parameters or automatically set by solving a nonlinear least squares fitting, once the minimum distance between the points in the low-dimensional representation (*min_dist* hyperparameter) and the effective scale of the embedded points (*spread* hyperparameter) is fixed. Specifically, a and b are found, if not previously given, fitting the real function $(1 + ax^{2b})^{-1}$ to the exponential decay function:

$$\psi(x) = \begin{cases} 1, & x < \text{min\_dist} \\ \exp(-(x - \text{min\_dist})/\text{spread}), & \text{otherwise} \end{cases},$$

for $x \in [0, 3\cdot\text{spread}]$. For instance, for the reference values min_dist $= 0.1$ and spread $= 1$, we have $a = 1.58$ and $b = 0.90$ (see [66] for details).

Therefore, after an initial guess $\mathbf{Y}_{\text{ini}}$ is computed, an iterative procedure is applied to find the minimum value in Equation (A4). The initial guess for the SGD method is the largest eigenvectors of the symmetric normalized graph Laplacian associated with $\mathbf{P}$ (see [67]). Specifically, we denote by $\mathbf{D} \in \mathbb{R}^{n\times n}$ the degree matrix of the undirected graph $\mathbf{P}$ ($\mathbf{D} = \text{diag}(D_{ii})$, where $D_{ii} = \sum_{j=1}^{n} p_{ij}$), and by $\mathbf{L} = \mathbf{D}^{1/2}(\mathbf{D} - \mathbf{P})\mathbf{D}^{1/2}$ the normalized graph Laplacian; then, $\mathbf{Y}_{\text{ini}}$ is the matrix containing only the first d eigenvectors with the largest eigenvalues.

Appendix B. Practical Computational Description of Extreme Learning Machine

Let $\mathbf{Y} \in \mathbb{R}^{n\times d}$ be a matrix collecting the n samples given in Equation (A3) to be classified, where each specific sample is given by $\mathbf{y}_i = \text{row}_i(\mathbf{Y})^T = (y_{i1}, y_{i2}, \ldots, y_{id})^T \in \mathbb{R}^d$

and d indicates the relevant features selected by the UMAP algorithm. In addition, let $\{C_1, C_2, \ldots, C_L\}$ be the classes/labels into which the samples must be classified, where L denotes the total number of classes. The ELM classifier takes as input a particular sample, $\mathbf{y}_i \in \mathbb{R}^d$, and returns the output vector, $\mathbf{o}_i = (o_{i1}, o_{i2}, \ldots, o_{iL})^T \in \mathbb{R}^L$, containing the raw score of the class membership to each of the L classes. By default, the ELM method uses binarized $\{-1, 1\}$ class labels, and thus, if sample $\mathbf{y}_i$ belongs to class C_{c_i}, we expect that $(\mathbf{o}_i)_j = o_{ij} \approx -1 + 2\delta_{j,c_i}$, i.e., we expect the c_ith component of the vector to be close to 1 and all the other components to be close to -1. The class prediction is then computed by selecting the maximum component of $\mathbf{o}_i$:

$$cp_i = \underset{j=1,\ldots,L}{\arg\max}\, o_{ij} \in \{1, 2, \ldots, L\}.$$

The output vector, $\mathbf{o}_i$, is computed from $\mathbf{y}_i$ by introducing a hidden intermediate layer in the neural network represented by vector $\mathbf{h}_i \in \mathbb{R}^{\tilde{d}}$, where $\tilde{d}$ denotes the number of nodes in the hidden layer.

In general, every layer of a neural network that takes as input a feature vector, $\mathbf{x}_{\text{in}} \in \mathbb{R}^{n_{\text{in}}}$, and returns an output vector, $\mathbf{x}_{\text{out}} \in \mathbb{R}^{n_{\text{out}}}$, is characterized by:

(1) An activation function $g : \mathbb{R} \to \mathbb{R}$;
(2) An n_{out} threshold or biases $b_j \in \mathbb{R}, j = 1, \ldots, n_{\text{out}}$, collected in vector $\mathbf{b} \in \mathbb{R}^{n_{\text{out}}}$; and
(3) n_{out} weighting vectors $\mathbf{w}_j \in \mathbb{R}^{n_{\text{in}}}, j = 1, \ldots, n_{\text{out}}$, collected in the weighting matrix $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{n_{\text{out}}})^T \in \mathbb{R}^{n_{\text{out}} \times n_{\text{in}}}$.

Thus, the output vector is computed as follows:

$$\mathbf{x}_{\text{out}} = g^{\circ}(\mathbf{W}\mathbf{x}_{\text{in}} + \mathbf{b}) = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_{\text{in}} + \mathbf{b}) \\ \vdots \\ g(\mathbf{w}_{n_{\text{out}}} \cdot \mathbf{x}_{\text{in}} + \mathbf{b}) \end{bmatrix},$$

where g° is the element-wise activation function that applies the activation function, g, to each component of the vector. Based on this definition, it is easy to recover the usual alternative expression for the jth component of the output vector:

$$(\mathbf{x}_{\text{out}})_j = g(\mathbf{w}_j \cdot \mathbf{x}_{\text{in}} + \mathbf{b}) = \sum_{s=1}^{n_{\text{in}}} g(w_{js}(\mathbf{x}_{\text{out}})_s + b_s).$$

In particular, the ELM neural network consists of a hidden layer that converts the initial samples $\mathbf{y}_i \in \mathbb{R}^d$ into the hidden feature vector $\mathbf{h}_i \in \mathbb{R}^{\tilde{d}}$ and an output layer that converts the hidden vector $\mathbf{h}_i \in \mathbb{R}^{\tilde{d}}$ into the output vector $\mathbf{o}_i \in \mathbb{R}^L$ (see Figure A4). The hidden layer consists of a weight matrix $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{\tilde{d}})^T \in \mathbb{R}^{\tilde{d} \times d}$, a bias vector $\mathbf{b} \in \mathbb{R}^{\tilde{d}}$, and a user-defined activation function g, whereas the output layer consists of a weight matrix $\mathbf{B} = (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_L)^T \in \mathbb{R}^{L \times \tilde{d}}$, a zero-bias vector, and an identity activation function. Therefore, after composing the two layers, we obtain an output vector of:

$$\mathbf{o}_i = \mathbf{B}\mathbf{h}_i = \mathbf{B}g^{\circ}(\mathbf{W}\mathbf{y}_i + \mathbf{b}).$$

Once the neural network parameters $(g, \mathbf{W}, \mathbf{b}, \mathbf{B})$ are set, it is straightforward to provide the class prediction for any given sample.

As mentioned before, the training step of the ELM neural network is comparable to that of a traditional neural network, because the weights and biases of the hidden layer are randomly assigned at the beginning of the learning process and remain unchanged during the entire training process. In particular, once the hidden number of features $\tilde{d}$

is set, the hidden weights and biases are randomly computed using the standardized normal distribution:

$$\mathbf{W} = \text{rand}_{\mathcal{N}(0,1)}(\tilde{d}, d) \in \mathbb{R}^{\tilde{d} \times d},$$
$$\mathbf{b} = \text{rand}_{\mathcal{N}(0,1)}(\tilde{d}, 1) \in \mathbb{R}^{\tilde{d}}.$$

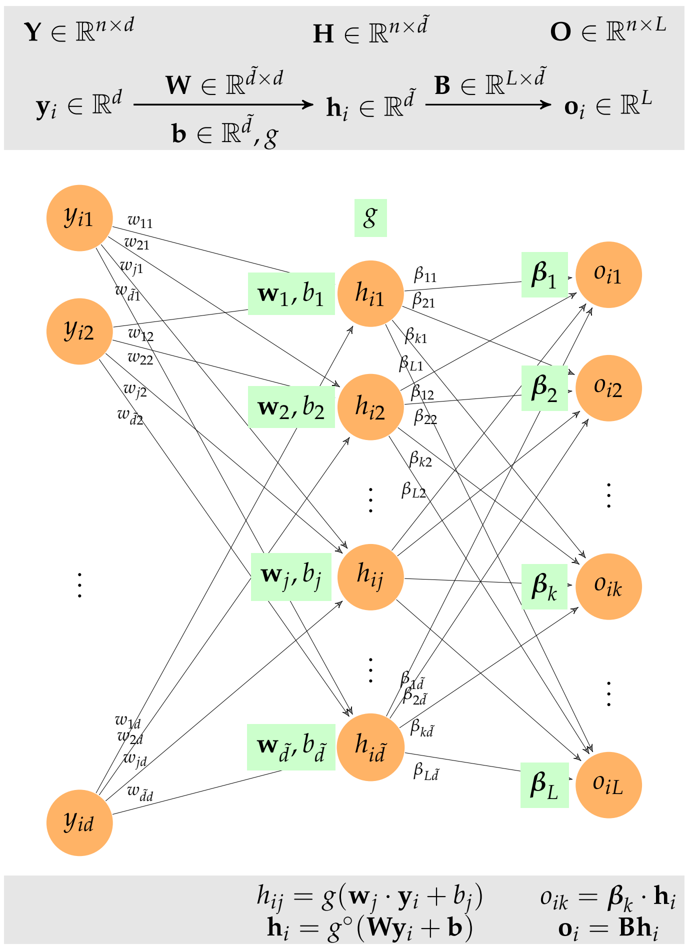
$\mathbf{Y} \in \mathbb{R}^{n\times d}$
$\mathbf{H} \in \mathbb{R}^{n\times \tilde{d}}$
$\mathbf{O} \in \mathbb{R}^{n\times L}$
$\mathbf{y}_i \in \mathbb{R}^d \xrightarrow[\mathbf{b} \in \mathbb{R}^{\tilde{d}}, g]{\mathbf{W} \in \mathbb{R}^{\tilde{d}\times d}} \mathbf{h}_i \in \mathbb{R}^{\tilde{d}} \xrightarrow{\mathbf{B} \in \mathbb{R}^{L\times \tilde{d}}} \mathbf{o}_i \in \mathbb{R}^L$
y_{i1}, y_{i2}, y_{id}
g
$\mathbf{w}_1, b_1$ $\mathbf{w}_2, b_2$ $\mathbf{w}_j, b_j$ $\mathbf{w}_{\tilde{d}}, b_{\tilde{d}}$
h_{i1} h_{i2} h_{ij} $h_{i\tilde{d}}$
β_1 β_2 β_k β_L
o_{i1} o_{i2} o_{ik} o_{iL}
w_{11} w_{21} w_{j1} $w_{\tilde{d}1}$ w_{12} w_{22} w_{j2} $w_{\tilde{d}2}$ w_{1d} w_{2d} w_{jd} $w_{\tilde{d}d}$
β_{11} β_{21} β_{k1} β_{L1} β_{12} β_{22} β_{k2} β_{L2} $\beta_{1\tilde{d}}$ $\beta_{2\tilde{d}}$ $\beta_{k\tilde{d}}$ $\beta_{L\tilde{d}}$
$h_{ij} = g(\mathbf{w}_j \cdot \mathbf{y}_i + b_j)$ $o_{ik} = \boldsymbol{\beta}_k \cdot \mathbf{h}_i$
$\mathbf{h}_i = g^{\circ}(\mathbf{W}\mathbf{y}_i + \mathbf{b})$ $\mathbf{o}_i = \mathbf{B}\mathbf{h}_i$

Figure A4. The ELM neural network consists of a hidden layer that converts the initial samples $\mathbf{y}_i \in \mathbb{R}^d$ into the hidden feature vector $\mathbf{h}_i \in \mathbb{R}^{\tilde{d}}$ and an output layer that converts the hidden vector $\mathbf{h}_i \in \mathbb{R}^{\tilde{d}}$ into the output vector $\mathbf{o}_i \in \mathbb{R}^L$.

Therefore, only the weights of the output layer $\mathbf{B}$ are computed using the training dataset. To illustrate the computation of $\mathbf{B}$, for ease of presentation and without loss of generality, let us assume that all data $\mathbf{Y} \in \mathbb{R}^{n\times d}$ are used to train the neural network (in practice, only some of the total data are used for training the ELM classifier, and $\mathbf{Y} \in \mathbb{R}^{n\times d}$ should be replaced by $\mathbf{Y} \in \mathbb{R}^{n_{\text{train}}\times d}$).

Because the class of the training samples is known, the raw score of the class membership for each sample is stored in the true output matrix:

$$\mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n)^T = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_n^T \end{bmatrix} \in \mathbb{R}^{n\times L}, \tag{A5}$$

where $\mathbf{t}_i = \text{row}_i(\mathbf{T})^T \in \mathbb{R}^L$ corresponds to the raw scores of the ith sample (containing -1 in the non-actual class and 1 in the actual class). Specifically, if the ith sample belongs to class $\mathcal{C}_{c_i}$, then $(\mathbf{t}_i)_j = 1$ if $j = c_i$ and -1 otherwise. Moreover, because the hidden layer is already known and is described by $(g, \mathbf{W}, \mathbf{b})$, we can compute all the hidden features of the input samples $\mathbf{h}_i = g^\circ(\mathbf{W}\mathbf{y}_i + \mathbf{b}) \in \mathbb{R}^{\tilde{d}}$ and collect them in the matrix:

$$\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)^T = \begin{bmatrix} \mathbf{h}_1^T \\ \vdots \\ \mathbf{h}_n^T \end{bmatrix} \in \mathbb{R}^{n \times \tilde{d}}, \tag{A6}$$

where the vectors $\mathbf{h}_i = \mathrm{row}_i(\mathbf{H})^T$ are now placed in the rows of matrix $\mathbf{H}$. Then, the output of the ELM neural network is:

$$\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_n)^T = \begin{bmatrix} \mathbf{o}_1^T \\ \vdots \\ \mathbf{o}_n^T \end{bmatrix} = \mathbf{H}\mathbf{B}^T \in \mathbb{R}^{n \times L}, \tag{A7}$$

where recall that $\mathbf{o}_i = \mathbf{B}\mathbf{h}_i$.

The output weights are then computed by minimizing the distances between the true outputs $\mathbf{T}$ and the predictions $\mathbf{O}$, namely, by finding a least-squares solution $\hat{\mathbf{B}}$ of the linear system $\mathbf{H}\mathbf{B}^T = \mathbf{T}$:

$$\hat{\mathbf{B}} = \underset{\mathbf{B} \in \mathbb{R}^{L \times \tilde{d}}}{\arg\min} \|\mathbf{H}\mathbf{B}^T - \mathbf{T}\|_F, \tag{A8}$$

where the Frobenius norm of a matrix is $\|\mathbf{A}\|_F = \sqrt{\mathrm{trace}(\mathbf{A}^T\mathbf{A})}$. Note that the system of equations, $\mathbf{H}\mathbf{B}^T = \mathbf{T}$, represents the $n \times L$ restrictions to be met, matrix $\mathbf{B}^T$ has $L \times \tilde{d}$ degrees of freedom, and, in general, the ELM can only approximate the training samples with zero error if the number of hidden nodes coincides with the total number of samples, $\tilde{d} = n$, in which case, $\mathbf{B}^T = \mathbf{H}^{-1}\mathbf{T}$. In the general case in which $\tilde{d} \neq n$ (note that the number of hidden nodes is usually much smaller than the number of samples), the output weights are computed by solving the least-squares problem in Equation (A8), yielding:

$$\hat{\mathbf{B}}^T = \mathbf{H}^\dagger\mathbf{T}, \tag{A9}$$

where $\mathbf{H}^\dagger$ is the Moore-–Penrose generalized inverse of $\mathbf{H}$ (see [68]). To further describe this key computation of the training step, despite the Moore–Penrose pseudo-inverse usually being computed using a singular decomposition of the matrix, in the usual case in which $\tilde{d} < n$, we have the following:

$$\hat{\mathbf{B}}^T = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T}.$$

The training steps of the ELM are summarized in Algorithm A1.

Algorithm A1. shows that the hidden layer parameters are randomly generated, independently of the dataset, and that the weights associated with the output layer are the only parameters that need to be trained. Equation (A9) reveals that the weights are computed explicitly without the use of an iterative procedure, and therefore, the ELM has a higher training speed than the BP learning algorithm.

Algorithm A1. ELM neural network training for classification.

```
Input: A training dataset, $\mathbf{Y} \in \mathbb{R}^{n\times d}$, of $n$ samples containing $d$ features; the class labels of the samples, $\{c_i\}_{i=1,\dots,n} \in \{1,\dots,L\}$; the number of nodes in hidden layer, $\tilde{d}$; the activation function, $g$, of the hidden layer.
Output: ELM classifier
1: Create the true raw score class membership matrix, $\mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n)^T \in \mathbb{R}^{n\times L}$, using binarized $\{-1, 1\}$ class labels, where $t_{ij} = 1$ if $j = c_i$ or $-1$, otherwise.
2: Randomly generate weighting matrix $\mathbf{W} \in \mathbb{R}^{\tilde{d}\times d}$ and bias vector $\mathbf{b} \in \mathbb{R}^{\tilde{d}}$ of the hidden layer, $\mathbf{W} = \mathrm{rand}_{\mathcal{N}(0,1)}(\tilde{d}, d)$, where $\mathbf{b} = \mathrm{rand}_{\mathcal{N}(0,1)}(\tilde{d}, 1)$.
3: Compute the output matrix of the hidden layer, $\mathbf{H} \in \mathbb{R}^{n\times\tilde{d}}$, associated with $\mathbf{Y}$, namely, $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n)^T$, where $\mathbf{h}_i = g^{\circ}(\mathbf{W}\mathbf{y}_i + \mathbf{b}) \in \mathbb{R}^{\tilde{d}}$.
4: Compute the weights of the output layer, $\mathbf{B} = (\mathbf{H}^{\dagger}\mathbf{T})^T \in \mathbb{R}^{L\times\tilde{d}}$.
```

References

1. Gardner, J.W.; Bartlett, P.N. A brief history of electronic noses. *Sens. Actuators B Chem.* **1994**, *18*, 210–211. [CrossRef]
2. Karakaya, D.; Ulucan, O.; Turkan, M. Electronic nose and its applications: A survey. *Int. J. Autom. Comput.* **2020**, *17*, 179–209. [CrossRef]
3. Marco, S.; Gutierrez-Galvez, A. Signal and data processing for machine olfaction and chemical sensing: A review. *IEEE Sens. J.* **2012**, *12*, 3189–3214. [CrossRef]
4. Yan, J.; Guo, X.; Duan, S.; Jia, P.; Wang, L.; Peng, C.; Zhang, S. Electronic nose feature extraction methods: A review. *Sensors* **2015**, *15*, 27804–27831. [CrossRef] [PubMed]
5. Lundström, I. Approaches and mechanisms to solid state based sensing. *Sens. Actuators B Chem.* **1996**, *35*, 11–19. [CrossRef]
6. Gutierrez-Osuna, R.; Nagle, H.T.; Schiffman, S.S. Transient response analysis of an electronic nose using multi-exponential models. *Sens. Actuators B Chem.* **1999**, *61*, 170–182. [CrossRef]
7. Gutierrez-Osuna, R.; Gutierrez-Galvez, A.; Powar, N. Transient response analysis for temperature-modulated chemoresistors. *Sens. Actuators B Chem.* **2003**, *93*, 57–66. [CrossRef]
8. Varpula, A.; Novikov, S.; Haarahiltunen, A.; Kuivalainen, P. Transient characterization techniques for resistive metal-oxide gas sensors. *Sens. Actuators B Chem.* **2011**, *159*, 12–26. [CrossRef]
9. Ducéré, J.M.; Hemeryck, A.; Estève, A.; Rouhani, M.D.; Landa, G.; Ménini, P.; Tropis, C.; Maisonnat, A.; Fau, P.; Chaudret, B. A computational chemist approach to gas sensors: Modeling the response of SnO2 to CO, O2, and H2O Gases. *J. Comput. Chem.* **2012**, *33*, 247–258. [CrossRef] [PubMed]
10. Kamarudin, K.; Mamduh, S.M.; Md Shakaff, A.Y.; Mad Saad, S.; Zakaria, A.; Abdullah, A.H.; Kamarudin, L.M. Flexible and autonomous integrated system for characterizing metal oxide gas sensor response in dynamic environment. *Instrum. Sci. Technol.* **2015**, *43*, 74–88. [CrossRef]
11. Siqueira, A.; Melo, M.; Giordani, D.; Galhardi, D.; Santos, B.; Batista, P.; Ferreira, A. Stochastic modeling of the transient regime of an electronic nose for waste cooking oil classification. *J. Food Eng.* **2018**, *221*, 114–123. [CrossRef]
12. Siqueira, A.F.; Vidigal, I.G.; Melo, M.P.; Giordani, D.S.; Batista, P.S.; Ferreira, A.L. Assessing waste cooking oils for the production of quality biodiesel using an electronic nose and a stochastic model. *Energy Fuels* **2019**, *33*, 3221–3226. [CrossRef]
13. Scott, S.M.; James, D.; Ali, Z. *Data Analysis for Electronic Nose Systems*; Springer: Berlin/Heidelberg, Germany, 2006. [CrossRef]
14. Zhang, L.; Tian, F.C. A new kernel discriminant analysis framework for electronic nose recognition. *Anal. Chim. Acta* **2014**, *816*, 8–17. [CrossRef] [PubMed]
15. Leon-Medina, J.X.; Anaya, M.; Parés, N.; Tibaduiza, D.A.; Pozo, F. Structural damage classification in a Jacket-type wind-turbine foundation using principal component analysis and extreme gradient boosting. *Sensors* **2021**, *21*, 2748. [CrossRef] [PubMed]
16. Leon-Medina, J.X.; Anaya, M.; Tibaduiza, D.A.; Pozo, F. Manifold Learning Algorithms Applied to Structural Damage Classification. *J. Appl. Comput. Mech.* **2021**, *7*, 1158–1166. [CrossRef]
17. Leon-Medina, J.X.; Anaya, M.; Pozo, F.; Tibaduiza, D. Nonlinear Feature Extraction Through Manifold Learning in an Electronic Tongue Classification Task. *Sensors* **2020**, *20*, 4834. [CrossRef] [PubMed]
18. Zhu, P.; Du, J.; Xu, B.; Lu, M. Modified unsupervised discriminant projection with an electronic nose for the rapid determination of Chinese mitten crab freshness. *Anal. Methods* **2017**, *9*, 1806–1815. [CrossRef]
19. Ding, L.; Guo, Z.; Pan, S.; Zhu, P. Manifold learning for dimension reduction of electronic nose data. In Proceedings of the 2017 International Conference on Control, Automation and Information Sciences, ICCAIS 2017, Xi'an, China, 31 October–1 November 2017; pp. 169–174. [CrossRef]
20. Jia, P.; Huang, T.; Wang, L.; Duan, S.; Yan, J.; Wang, L. A novel pre-processing technique for original feature matrix of electronic nose based on supervised locality preserving projections. *Sensors* **2016**, *16*, 1019. [CrossRef] [PubMed]

21. Leon-Medina, J.X.; Anaya, M.; Pozo, F.; Tibaduiza, D.A. Application of manifold learning algorithms to improve the classification performance of an electronic nose. In Proceedings of the 2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Dubrovnik, Croatia, 25–28 May 2020; pp. 1–6. [CrossRef]
22. Pardo, M.; Sberveglieri, G. Classification of electronic nose data with support vector machines. *Sens. Actuators B Chem.* **2005**, *107*, 730–737. [CrossRef]
23. Tan, J.; Kerr, W.L. Determining degree of roasting in cocoa beans by artificial neural network (ANN)-based electronic nose system and gas chromatography/mass spectrometry (GC/MS). *J. Sci. Food Agric.* **2018**, *98*, 3851–3859. [CrossRef]
24. Leon-Medina, J.X.; Cardenas-Flechas, L.J.; Tibaduiza, D.A. A data-driven methodology for the classification of different liquids in artificial taste recognition applications with a pulse voltammetric electronic tongue. *Int. J. Distrib. Sens. Netw.* **2019**, *15*, 1550147719881601. [CrossRef]
25. Vergara, A.; Vembu, S.; Ayhan, T.; Ryan, M.A.; Homer, M.L.; Huerta, R. Chemical gas sensor drift compensation using classifier ensembles. *Sens. Actuators B Chem.* **2012**, *166*, 320–329. [CrossRef]
26. Fonollosa, J.; Rodríguez-Luján, I.; Huerta, R. Chemical gas sensor array dataset. *Data Brief* **2015**, *3*, 85–89. [CrossRef]
27. Vallejo, M.; de la Espriella, C.; Gómez-Santamaría, J.; Ramírez-Barrera, A.F.; Delgado-Trejos, E. Soft metrology based on machine learning: A review. *Meas. Sci. Technol.* **2019**, *31*, 032001. [CrossRef]
28. Ting, W.; Guo-Zheng, Y.; Bang-Hua, Y.; Hong, S. EEG feature extraction based on wavelet packet decomposition for brain computer interface. *Measurement* **2008**, *41*, 618–625. [CrossRef]
29. Zhang, L.; Liu, Y.; Deng, P. Odor recognition in multiple E-nose systems with cross-domain discriminative subspace learning. *IEEE Trans. Instrum. Meas.* **2017**, *66*, 1679–1692. [CrossRef]
30. Zhang, L.; Zhang, D. Domain adaptation extreme learning machines for drift compensation in E-nose systems. *IEEE Trans. Instrum. Meas.* **2014**, *64*, 1790–1801. [CrossRef]
31. Kumar, S.; Ghosh, A. A Feature Extraction Method Using Linear Model Identification of Voltammetric Electronic Tongue. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 9243–9250. [CrossRef]
32. Jing, Y.Q.; Meng, Q.H.; Qi, P.F.; Cao, M.L.; Zeng, M.; Ma, S.G. A bioinspired neural network for data processing in an electronic nose. *IEEE Trans. Instrum. Meas.* **2016**, *65*, 2369–2380. [CrossRef]
33. Ozmen, A.; Dogan, E. Design of a portable E-nose instrument for gas classifications. *IEEE Trans. Instrum. Meas.* **2009**, *58*, 3609–3618. [CrossRef]
34. Krutzler, C.; Unger, A.; Marhold, H.; Fricke, T.; Conrad, T.; Schütze, A. Influence of MOS gas-sensor production tolerances on pattern recognition techniques in electronic noses. *IEEE Trans. Instrum. Meas.* **2011**, *61*, 276–283. [CrossRef]
35. Brudzewski, K.; Osowski, S.; Dwulit, A. Recognition of coffee using differential electronic nose. *IEEE Trans. Instrum. Meas.* **2012**, *61*, 1803–1810. [CrossRef]
36. Tudu, B.; Metla, A.; Das, B.; Bhattacharyya, N.; Jana, A.; Ghosh, D.; Bandyopadhyay, R. Towards versatile electronic nose pattern classifier for black tea quality evaluation: An incremental fuzzy approach. *IEEE Trans. Instrum. Meas.* **2009**, *58*, 3069–3078. [CrossRef]
37. Zhang, L.; Tian, F. Performance study of multilayer perceptrons in a low-cost electronic nose. *IEEE Trans. Instrum. Meas.* **2014**, *63*, 1670–1679. [CrossRef]
38. Vanarse, A.; Osseiran, A.; Rassau, A. Neuromorphic engineering—A paradigm shift for future im technologies. *IEEE Instrum. Meas. Mag.* **2019**, *22*, 4–9. [CrossRef]
39. Al Yamani, J.H.J.; Boussaid, F.; Bermak, A.; Martinez, D. Bio-inspired gas recognition based on the organization of the olfactory pathway. In Proceedings of the 2012 IEEE International Symposium on Circuits and Systems (ISCAS), Seoul, Korea, 20–23 May 2012; pp. 1391–1394.
40. Pan, X.; Zhang, H.; Ye, W.; Bermak, A.; Zhao, X. A fast and robust gas recognition algorithm based on hybrid convolutional and recurrent neural network. *IEEE Access* **2019**, *7*, 100954–100963. [CrossRef]
41. Liu, H.; Yu, D.; Gu, Y. Classification and evaluation of quality grades of organic green teas using an electronic nose based on machine learning algorithms. *IEEE Access* **2019**, *7*, 172965–172973. [CrossRef]
42. Liu, H.; Li, Q.; Gu, Y. A multi-task learning framework for gas detection and concentration estimation. *Neurocomputing* **2020**, *416*, 28–37. [CrossRef]
43. Cheng, Y.; Wong, K.Y.; Hung, K.; Li, W.; Li, Z.; Zhang, J. Deep Nearest Class Mean Model for Incremental Odor Classification. *IEEE Trans. Instrum. Meas.* **2018**, *68*, 952–962. [CrossRef]
44. Grover, A.; Lall, B. A Novel Method For Removing Baseline Drifts in Multivariate Chemical Sensor. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 7306–7316. [CrossRef]
45. Zhang, L.; Wang, X.; Huang, G.B.; Liu, T.; Tan, X. Taste recognition in e-tongue using local discriminant preservation projection. *IEEE Trans. Cybern.* **2018**, *49*, 947–960. [CrossRef] [PubMed]
46. Leon-Medina, J.X.; Pineda-Muñoz, W.A.; Burgos, D.A.T. Joint Distribution Adaptation for Drift Correction in Electronic Nose Type Sensor Arrays. *IEEE Access* **2020**, *8*, 134413–134421. [CrossRef]
47. Gamboa, J.C.R.; da Silva, A.J.; de Andrade Lima, L.L.; Ferreira, T.A. Wine quality rapid detection using a compact electronic nose system: Application focused on spoilage thresholds by acetic acid. *LWT* **2019**, *108*, 377–384. [CrossRef]
48. Gamboa, J.C.R.; da Silva, A.J.; Ferreira, T.A. Electronic nose dataset for detection of wine spoilage thresholds. *Data Brief* **2019**, *25*, 104202. [CrossRef] [PubMed]

49. Yin, X.; Zhang, L.; Tian, F.; Zhang, D. Temperature modulated gas sensing E-nose system for low-cost and fast detection. *IEEE Sens. J.* **2015**, *16*, 464–474. [CrossRef]
50. Plastria, F.; De Bruyne, S.; Carrizosa, E. Dimensionality Reduction for Classification. In *Proceedings of the 4th International Conference on Advanced Data Mining and Applications*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 411–418. [CrossRef]
51. Saul, L.; Weinberger, K.; Ham, J.; Sha, F. Spectral methods for dimensionality reduction. *Semisupervised Learn.* **2006**, 293–306. [CrossRef]
52. Schölkopf, B.; Smola, A.J.; Bach, F. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*; MIT Press: Cambridge, MA, USA, 2002.
53. Scholkopf, B.; Mika, S.; Burges, C.J.; Knirsch, P.; Muller, K.R.; Ratsch, G.; Smola, A.J. Input space versus feature space in kernel-based methods. *IEEE Trans. Neural Netw.* **1999**, *10*, 1000–1017. [CrossRef] [PubMed]
54. McInnes, L.; Healy, J.; Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv* **2018**, arXiv:1802.03426.
55. McInnes, L. UMAP for Supervised Dimension Reduction and Metric Learning. 2018. Available online: https://umap-learn.readthedocs.io/en/latest/supervised.html (accessed on 19 July 2021).
56. Sainburg, T.; McInnes, L.; Gentner, T.Q. Parametric UMAP: Learning embeddings with deep neural networks for representation and semi-supervised learning. *arXiv* **2020**, arXiv:2009.12981.
57. McInnes, L. Transforming New Data with UMAP. 2018. Available online: https://umap-learn.readthedocs.io/en/latest/transform.html (accessed on 10 October 2021).
58. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. [CrossRef]
59. Xu, S.; Liu, S.; Feng, L. Fuzzy granularity neighborhood extreme clustering. *Neurocomputing* **2020**, *379*, 236–249. [CrossRef]
60. Huang, G.B.; Zhou, H.; Ding, X.; Zhang, R. Extreme learning machine for regression and multiclass classification. *IEEE Trans. Syst. Man Cybern.* **2011**, *42*, 513–529. [CrossRef] [PubMed]
61. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manag.* **2009**, *45*, 427–437. [CrossRef]
62. McGinnis, W. Extreme Learning Machines, Sklearn-Extensions. 2015. Available online: http://wdm0006.github.io/sklearn-extensions/extreme_learning_machines.html (accessed on 19 July 2021).
63. Jiménez, Á.B.; Lázaro, J.L.; Dorronsoro, J.R. Finding optimal model parameters by deterministic and annealed focused grid search. *Neurocomputing* **2009**, *72*, 2824–2832. [CrossRef]
64. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
65. León, J.X.; Pineda Muñoz, W.A.; Anaya, M.; Vitola, J.; Tibaduiza, D.A. Structural Damage classification using machine learning algorithms and performance measures. In Proceedings of the 12th International Workshop On Structural Health Monitoring-IWSHM 2019, Stanford, CA, USA, 10–12 September 2019.
66. Melville, J. Fine-Tuning UMAP Visualizations. 2020. Available online: https://jlmelville.github.io/uwot/abparams.html (accessed on 10 October 2021).
67. Shen, T. The Mathematics Behind Spectral Clustering And The Equivalence To PCA. *arXiv* **2021**, arXiv:2103.00733.
68. Serre, D. *Matrices Theory and Applications*, 2nd ed.; Springer: New York, NY, USA, 2010.