Expert Systems With Applications 230 (2023) 120627

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Deep self-organizing cube: A novel multi-dimensional classifier for multiple output learning

Ahmed Abdelfattah Saleh [a,*], Li Weigang [b]

[a] PPMEC, Department of Mechanical Engineering, University of Brasilia, Brasilia, Brazil
[b] TransLab, Department of Computer Science, University of Brasilia, Brasilia, Brazil

ARTICLE INFO

Keywords:
Multi-dimensional classification
Multi-label classification
Deep learning classifiers
DSOC
Multiple-output learning

ABSTRACT

Multi-dimensional classification (MDC) task can be considered the most inclusive description of all classifications tasks, as it joins multiple class spaces and their multiple class members into a single compound classification problem. The challenges in MDC arise from the possible class dependencies across different class spaces, as well as the imbalance of labels in training datasets due to lack of all possible combinations. In this paper, we propose a straightforward, yet efficient, MDC deep learning classifier, named *Deep Self-Organizing Cube* (*DSOC*) that can model dependencies among classes in multiple class spaces, while consolidating its ability to classify rare combinations of labels. DSOC is formed of two *n*-dimensional components, namely the *Hypercube Classifier* and the multiple *DSOC Neural Networks* connected to the hypercube. The multiple neural networks component is responsible for feature selection and segregation of classes, while the Hypercube classifier is responsible for creating the semantics among multiple class spaces and accommodate the model for rare sample classification. DSOC is a multiple-output learning algorithm that successfully classify samples across all class spaces simultaneously. To challenge the proposed DSOC model, we conducted an assessment on seventeen benchmark datasets in the four types of classification tasks, binary, multi-class, multi-label and multi-dimensional. The obtained results were compared to four standard classifiers and eight competitive state-of-the-art approaches reported in literature. The DSOC has achieved superior performance over standard classifiers as well as the state-of-the-art approaches in all the four classification tasks. Moreover, in terms of *Exact Match* accuracy metrics, DSOC has outperformed all state-of-the-art approaches in 77.8% of the cases, which reflects the superior ability of DSOC to model dependencies and successfully classify rare samples across all dimensions simultaneously.

1. Introduction

Multi-dimensional classification (MDC) is a generalization of the multi-label classification (MLC) problem where the classification output is a vector in multiple heterogeneous class spaces. The output (Y) is a vector of multiple class variables, where each member (Y_i) specifies its class membership in one particular class space. Ma & Chen (Ma & Chen, 2018) have displayed the different types of classification problems in terms of the number of output class variables (m) and their possible values (k) as shown in Fig. 1. Both binary classification (BC) and multi-class classification (MCC) have a single output class variable, whose value is binary [0, 1] in case of BC and multiple [0, 1,..K] in MCC. On the other hand, both MLC and MDC have multiple output class variables ($m \geq 2$), whose values are binary in case of MLC and multiple in MDC.

From another perspective, BC and MCC are considered single-output learning problems, while MLC and MDC are categorized under multiple-output learning. Multiple-output classification aims at simultaneously predicting multiple outputs given a single set of input variables. Xu et al. (2020) have pointed out the lack of sufficient studies that generalize different forms of multiple-output learning despite of its growing importance in complex decision-making problems and the attention it has got in the recent years. As such, they conducted a thorough survey, depicting all aspects of multi-output learning including challenges, inputs and outputs structure, evaluation metrics as well as applications (e.g. Multi-label Document Categorization (Song, Vold, Madan, & Schilder, 2022), Multi-label Semantic Scene Classification (Senthilkumar & Akshayaa, 2020), Multi-label Video Annotation (Xu, Jiang, Xue, & Zhou, 2012), etc.).

Due to the multi-variate nature of the output space of multi-output classification, classes across various dimensions might exhibit

* Corresponding author.
E-mail addresses: ahmdsalh@yahoo.com (A.A. Saleh), weigang@unb.br (L. Weigang).

https://doi.org/10.1016/j.eswa.2023.120627
Received 6 January 2023; Received in revised form 25 April 2023; Accepted 27 May 2023
Available online 2 June 2023

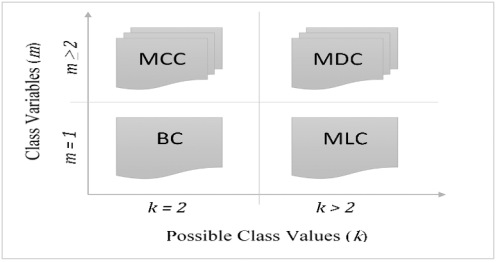

Fig. 1. Different classification tasks, where m is the number of class variables and k is the number of possible values per class variable. (Ma & Chen, 2018).

dependencies, correlations and/or complex interactions that should be considered when designing the classification model. Such dependencies and interactions means that the prediction of one label influences predictions of other labels. MDC models that do not take into consideration such dependencies will suffer from propagation of error and lack of classification accuracy across the entire class spaces. Independent Classifiers (IC) or Binary Relevance (BR) that involve building single dimension multi-class classifier for each class variable (Zhang M. L., Li, Liu, & Geng, 2018) suffer from lack of accuracy, while Classifiers Chain (CC) that involves chain of Bayesian classifiers (Read, Pfahringer, Holmes, & Frank, Classifier chains for multi-label classification., 2011) or even the Monte Carlo scheme for chain sequencing and inference (Read, Martino, & Luengo, 2014) suffer from error propagation due to the difficulties in finding optimal chain order. (Jia & Zhang, 2020b).

Dembczynski et al. (Dembszynski, Waegeman, Cheng, & Hüllermeier, 2010) have categorized label dependencies among classes in MLC into two categories, *conditional* and *unconditional* label dependencies. Using the standard statistical notations of multivariate regression models ($Y_i = h_i(\boldsymbol{X}) + \epsilon_i(\boldsymbol{X})$), *unconditional* dependence represents the deterministic part (i.e. the function $h_i(\boldsymbol{X})$), while the *conditional* dependence represents the stochastic part (i.e. the error $\epsilon_i(\boldsymbol{X})$). As such, *unconditional* dependence refers to the expected dependencies between classes in Y, and can be measured using Pearson correlation or any other correlation coefficients.

To overcome the aforementioned drawbacks and limitations as well as to model the underlying class dependencies in MLC, MCC and MDC, several models and techniques were introduced in literature. Some of which involve models that consider all possible class combinations as *label powerset* (LP) or modified versions of it (Junior, Faria, Silva, & Cerri, 2017), others create *Super-Classes* based on the *conditional dependencies* among all or part of the class spaces (Read, Bielza, & Larrañaga, Multi-dimensional classification with super-classes., 2013), as well as models based on pairwise interactions among those classes (Arias, Gamez, Nielsen, & Puerta, 2016). Others used ensembles to tackle the problem of label-combinations, as for example EPS that discard the less frequent label combinations, RAKEL that creates ensemble of random label subsets, etc. (Charte, Rivera, del Jesus, & Herrera, 2015). However, those methods still suffer from the *imbalance problem*, where the output class variables combinations might not be uniformly distributed over the data space in the training set. This imbalance causes the classifier to lean (skew) towards the majority classes and fail to model the characteristic of minority classes, with the risk of overfitting which will affect the accuracy and robustness of the overall model. (Tarekegn, Giacobini, & Michalak, 2021).

In this paper, inspired by Self-Organizing Map (Kohonen, 1990) and Once Learning approach (Weigang L., 1998) (Weigang & Silva, 1999), we propose a new Deep Learning classifier, the *Deep Self-organizing Cube*, (*DSOC*) that can be used for BC, MLC, MCC as well as MDC. DSOC takes into consideration class dependencies across class spaces while implementing multi-output learning, where classification across all dimensions is performed simultaneously. Due to the self-organizing nature of DSOC's *Pooling* layer as well as its probabilistic *Hypercube*, the model overcomes the problem of class imbalance among training dataset instances.

As such, the main contributions of this study are as follows: a) we propose an efficient, yet straightforward multidimensional deep learning classifier, the "DSOC". b) The model applies multi-output learning in the four classification tasks, with no need for a domain or task specific model adjustment. c) The model's straightforward approach in both training and implementation phases, regardless the nature of the classification task or dataset, makes it a *domain and task agnostic* classification model. d) The model has an embedded variable selection layers to achieve dimensionality reduction while increasing the discriminatory power of the model and achieve appropriate class segregation. e) This work experimentally demonstrates the effectiveness of DSOC in all types of classification tasks, regardless the severity of data imbalance or the strength of class dependencies, due to its design that models semantics among classes along different classification spaces even in case of imbalanced datasets.

The rest of the paper is structured as follows, a brief literature review for different classification approaches used in MDC tasks is presented in section 2, and then we introduce the proposed DSOC classifier in section 3, followed by the applied experiment in section 4 and the results and discussions in section 5. Finally, we conclude the paper in section 6.

2. Related work

Bogatinovski et al. (Bogatinovski, Todorovski, Džeroski, & Kocev, 2022) pointed out the increasing interest in MLC task from the machine learning community, where they displayed a graph that showed an exponential growth in the number of scientific papers published throughout the last decade. This section presents the recent work in literature that tackles the classification problem from a multi-way perspective and its recent applications.

As mentioned in the introduction section, MDC is a generalized form of multi-label classification problem. Therefore, we are going to use the term MDC throughout the paper when referring to all types of non-binary classification problems, unless otherwise specified.

Despite the inability of Binary Relevance (BR) approach to model class correlations and dependencies, nevertheless BR is one of the most widely used approaches for MDC due to its simplicity and intuitiveness. BR techniques are built in one of two structures, *Chaining Structure* and *Stacking Structure*, or a mix of them. In *Chaining Structure*, independent binary classifiers are arranged in a chain order based on the results of the previous classifiers (Read, Pfahringer, Holmes, & Frank, Classifier chains for multi-label classification., 2009). While in *Stacking Structure*, a set of *meta*-level BR models are stacked over another set of base-level BR models, where each *meta*-level binary classifier is built upon the predictions of all base level ones (Godbole & Sarawagi, 2004).

Many variations of BR were introduced lately in an attempt to overcome the BR core limitations, some of which are: a) BR Stacking based on Pareto Optimum, a modified version of the staking method that takes into consideration the inherit nature of class labels and their own related subsets (Weng, Chen, Wu, Li, & Wen, 2019). b) Dependent binary relevance (DBR), which is a mixed approach of chaining and stacking (Montañes, et al., 2014). c) Stacking Model with Label Selection (SMLS), a two layer stacking based approach that overcomes limitations of DBR, through intensifying the class correlations in subsets to augment the features space (Chen, et al., 2021).

While, most MDC approaches tends to model multiple class dependencies in the output class spaces, Jia and Zhang (Jia & Zhang, Multi-dimensional classification via kNN feature augmentation., 2020a) tried a different approach by manipulating the input features space by enriching the existing original feature space with a set of new features. Their augmented feature approach, called *KRAM*, uses simple counting statistics and weighted *k*NN techniques (with extra bias terms) depending on the class membership of k nearest neighbors in the training set. *KRAM* showed good results when compared to other models

that use original input features space. Both Jia and Zhang further extended their feature-augmentation strategy in (Jia & Zhang, Multi-dimensional classification via selective feature augmentation., 2022) by introducing SFAM, an abbreviation for Selective Features Augmentation for Multi-dimensional classification. SFAM synergizes multiple kind of augmented features (standard *k*NN, weighted *k*NN and maximum margin techniques) to achieve classification along different dimensions. Furthermore, Wang et al. (Wang, et al., 2020) proposed a deep neural network based model that integrates the Feature Augmentation and Label Embedding techniques to model the inter-class correlations and the intra-class exclusiveness in MDC problems.

In a related context, and for shading the light on the importance of features extraction, De Handschutter et al. (De Handschutter, Gillis, & Siebert, 2021) published a thorough survey on deep matrix factorization (deep MF) in comparison with constrained low-rank matrix approximations (CLRMA) to deal with the extraction of several layers of features as a principal step before conducting further machine learning tasks.

As mentioned earlier, class imbalance problem might affect the classification accuracy of a MDC model as well as its performance evaluation metrics (Luque, Carrasco, Martín, & de Las Heras, 2019). Feature space manipulation can tackle the imbalance problem of MDC. Mishra and Singh (Mishra & Singh, 2021) proposed a method called Feature Construction and Smote-based Imbalance handling (FCSMI), to tackle the problem of class combinations imbalances in MLD. FCSMI uses the distance between minority classes and others to alter the feature space and achieve the balance between minority and majority classes.

In addition, (Han & Zhang, 2022) proposed a Multi-Kernel Multi-Label (MKML) method to address, simultaneously, both class dependencies and class imbalance problems in MLC. While, (Duarte, Rawat, & Shah, 2021) presented Partial Label Masking (PLM) method to tackle imbalanced datasets by partially masking major and minor classes and then continually adapts the target ratio based on the output probabilities, aiming at improving precision on frequent classes and recall on less frequent ones. In image classification, (Kim, Lee, & Jeon, 2021) proposed a new loss function that reduces the risk of misclassification of less frequent classes, by neutralizing the probability distribution of incorrect classes leading to a more robust classification of class-imbalanced scenarios. It is worth mentioning that the degree of class imbalance and its impact on MDC can be assessed by using either the *imbalance-ratio* or the *imbalance-degree* coefficient presented by (Ortigosa-Hernández, Inza, & Lozano, 2017), which is more sensitive in reflecting the skewness in MDC distributions.

In recent years, many applications in different domains have benefited from MDC techniques. In the field of medicine, Yang (Yang, Li, Li, & Gong, 2022) constructed a MDC model based on Support Vector Machine (SVM) for the diagnosis of schizophrenia and bipolar disorder. While in text MDC, Xie et al. in (Xie, Lin, Lin, Wang, & Yu, 2021) proposed a multi-dimensional relation model to incorporate relations between dimensions for dimension score prediction in multi-dimensional sentiment analysis (valence-arousal-irony, VAI) of a Chinese corpus. Other applications in the field of text classification include Fake news classification using Long short-term memory (LSTM) and Bidirectional Encoder Representations from Transformers (BERT) (Rai, Kumar, Kaushik, Raj, & Ali, 2022), emotions classifications using LSTM and Transformer Networks (RoBERTa and DistilBERT) (Ameer, et al., 2023). In cybersecurity, a multi-dimensional feature fusion and stacking ensemble mechanism (MFFSEM) was used to detect network intrusion and abnormal behaviors (Zhang H., Li, Liu, & Dong, 2021).

3. Deep Self-Organizing cube (DSOC)

In this section, we propose a new Deep Learning classifier that can be used in all four types of supervised classification tasks (BC, MCC, MLC and MDC). The proposed model is called "*DSOC*" a short for "*Deep Self Organizing Cube*". The model is based on the well-established concept of Self-Organizing Maps (SOM) introduced by (Kohonen, 1990) and the Once Learning approach introduced by (Weigang L., 1998) (Weigang & Silva, 1999) that adapts the SOM training routine into all-at-once learning, imitating the human brain activity of learning just once, "Once Seen Never Forgotten" approach. As Xu [5] pointed out, the purpose of multi-instance multi-label learning is to predict multiple outputs at the same time given one input. Once learning mechanism reflects the requirement of this simultaneity.

In "*Deep Self-Organizing Cube*", the word "Deep" refers to the nature of the model that involves a multi-layer system of connected neurons, arranged in multiple abstraction levels (dimensions). While, the term "*Self-Organizing*" refers to the ability of the neurons within the model to organize themselves into various arrangements, enabling the "*DSOC*" to model inter and intra class dependencies. Finally, the word "*Cube*" refers to the multi-dimensional hypercube that lies at the center of the multi-layer neuron system.

In short, "*DSOC*" is a deep learning classifier, which consists of multi-layer neuron system connected to a central hypercube. The hypercube is an *n*-way cube formed of output neurons arranged in orthogonal planes, where each plane represents a classification dimension of a class apace.

3.1. DSOC components

Despite the multi-dimensional and multi-layer nature of the DSOC model, its structure can be divided into two major components, the *Hypercube Classifier* component, and the *Deep Neural Network* component.

3.1.1. The DSOC hypercube classifier

The *DSOC's* hypercube component is a multi-way cube responsible for the multi-output classification task across all class spaces. The hypercube is formed of *n* dimensions, each of which represents a classification space (C) with its own set of classes. Where, an *n-way* hypercube consists of *n* class spaces, i.e. ($Hypercube = C^1 \times C^2 \times \cdots \times C^n$), with each class space formed of *m* classes ($C^n = \{C_1^n, C_2^n, \cdots, C_m^n\}$). As such, a 3-dimensional hypercube is formed of N output neurons equal to the product of class members of all class spaces, i.e. ($N = C^1 \times C^2 \times C^3$). Each neuron $c_{1,2,\ldots,n}$ represents a combination of classes in the *n*-dimensional cube. The classification concept of the entire DSOC model is to find the winning neuron in the *Hypercube classifier*. Therefore, the final classification output of the model is a class vector (Y) of size *n* whose members represent the assigned classes of the winning neuron across the *n* dimensions, i.e. ($Y = [y^1, y^2, \cdots, y^n]$).

From a different viewpoint, in case of a 3-dimensional *DSOC*, the hypercube classifier can be viewed as a stack of slices/planes of size ($C^1 \times C^2$) each of which represent a two-dimensional classification plane for a single class space (C^3) along the third dimension. For example, in a problem where text scientific papers are to be classified according to their *Topics*, *Languages* and scientific *Fields*. *DSOC* considers this example a 3-way multidimensional classification problem along three linked but discrete modes, of size $C^1, C^2$and C^3 respectively. As seen in Fig. 2, the three-dimensional *hypercube classifier* could be seen as: a) a stack of C^2 two-dimensional planes of size ($C^1 \times C^3$) along the *Languages* dimension; b) a stack of C^3 two-dimensional planes of size ($C^1 \times C^2$) along the *Fields* dimension; or c) a stack of C^1 two-dimensional planes of size ($C^2 \times C^3$) along the *Topics* dimension.

Using the previous example, if the DSOC algorithm has selected a winning neuron ($c_{2,4,3}$) whose coordinates in the hypercube classifier are (2, 4, 3), then this neuron represents a document whose *Topic* belongs to class 2, written in *Language* 4 and under *Scientific Field* 3.

3.1.2. The deep neural network component

In order for the *DSOC's Hypercube classifier* to emit the classification vector (Y) across all dimensions, one of its output neuron has to be activated. The winning neuron activation process occur through a stack of deep neural networks, that we call "*DSOC Neural Network*", with one

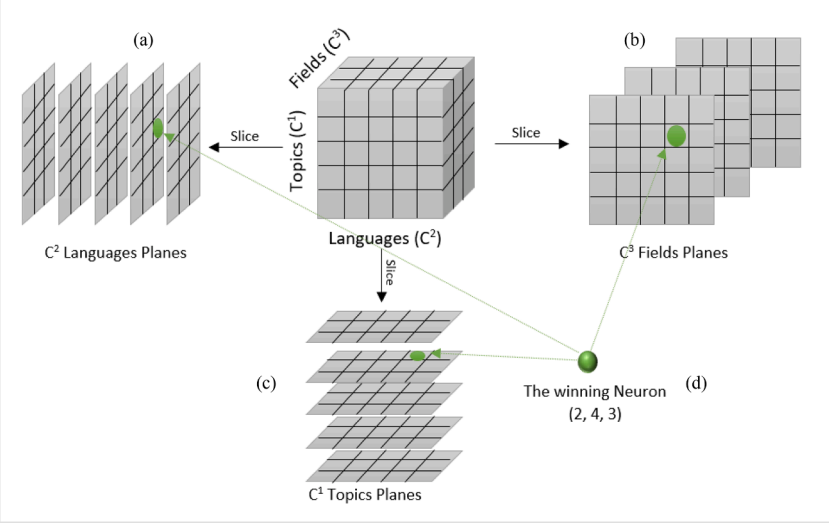

Fig. 2. A three-dimensional hypercube classifier of size ($C^1 \times C^2 \times C^3$). a) C^2 Slices along the Language dimension. b) C^3 slices along the *Fields* dimension. c) C^1 slices along the *Topics* dimension. d) The winning neuron at hypercube coordinates (2, 4, 3).

network for each dimension of the model. Therefore, in an *n*-dimensional DSOC model there are *n* "*DSOC Neural Networks*" that are connected to all *N* output neurons of the DSOC's *Hypercube*.

As seen in Fig. 3, a single "*DSOC Neural Network*" consists of three main layers of neurons:

a) *Input layer*, consists of *P* neurons corresponding to sample input features (*p*).
b) *VSC double layer*, which consists of two connected layers, *VSC Selector* layer and *VSC Scaling* layer, this double layer depends on the *Variables Selection Coefficient* (VSC) algorithm introduced by (Saleh & Weigang, A New Variables Selection And Dimensionality Reduction Technique Coupled With Simca Method For The Classification Of Text Documents, 2015).
c) *Pooling layer*, a Gaussian Probability function layer, whose neurons are directly connected to the DSOC hypercube classifier, and responsible for activating the hypercube winning neuron.

3.1.2.1. The VSC double layer. *Variable Strength Coefficient (VSC)* is the overall measure of strength of a variable (feature) in fitting the data, as well as discriminating the classes within a class space. VSC combines, linearly, both *Modeling Power* of a variable for all classes in the model with the *Discriminatory Power* of the same variable for the same class space. The *Modeling Power* is the assessment of the ability of a variable to model data of a specific class, while *Discriminatory Power* is the assessment of its ability to discriminate between two classes. For example, for a variable *i* in class *j* the *Modeling Power* M_i^j, and its *Discriminatory Power* $D_i^{j,j+1}$ for classes *j* and $j+1$, are calculated as follows (Brereton, 2003):

$$M_i^j = 1 - \frac{S_{iraw}^j}{S_{iresid}^j} \tag{1}$$

$$D_i^{j,j+1} = \sqrt{\frac{jmodel(j+1)_{S_{iresid}}{}^2 + (j+1)modelj_{S_{iresid}}{}^2}{jmodelj_{S_{iresid}}{}^2 + (j+1)model(j+1)_{S_{iresid}}{}^2}} \tag{2}$$

Where S_{iraw}^j is the standard deviation of variable *i* in the raw data and S_{iresid}^j is the standard deviation of the residuals for the same variable *i* in the model.

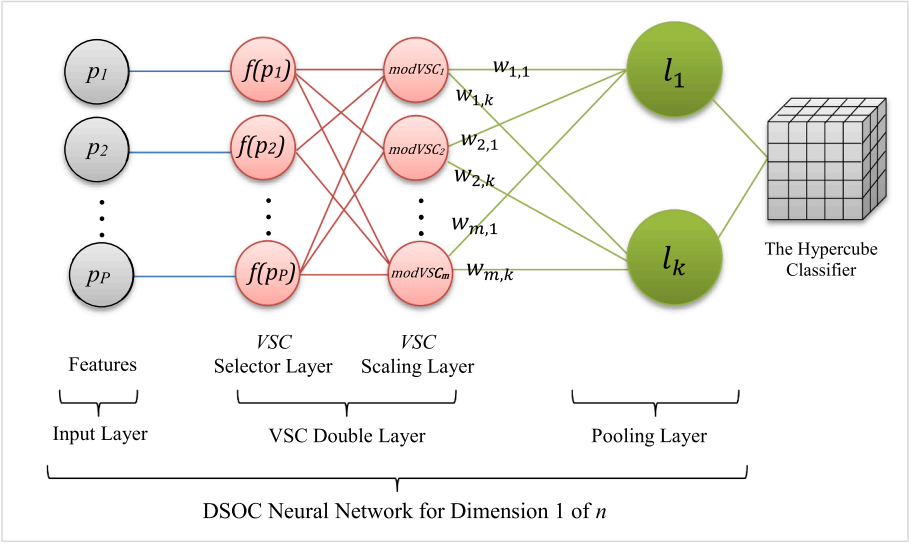

Fig. 3. The DSOC Neural Network for Dimension 1 of n.

The VSC for a variable i ($\boldsymbol{VSC}_i$) is then calculated by linearly combining the overall modeling power of that variable ($\boldsymbol{M}_i$) along all classes in the class space, with its overall discriminatory power ($\boldsymbol{D}_i$) as shown in the equations below (Saleh, Hegazy, Abbas, & Elkosasy, 2022).

$$\boldsymbol{M}_i = \sum_{j=1}^{l} \boldsymbol{M}_i^j . \frac{n^j}{N} \tag{3}$$

$$\boldsymbol{D}_i = \sqrt{\frac{\sum_{j}^{l} \sum_{c}^{l} jmodelc_{S_{iresid}}{}^2}{\sum_{j}^{l} jmodelj_{S_{iresid}}{}^2}} \tag{4}$$

$$\boldsymbol{D}_i = \frac{\boldsymbol{D}_i - \min\{\boldsymbol{D}\}}{\max\{\boldsymbol{D}\} - \min\{\boldsymbol{D}\}} \tag{5}$$

$$\boldsymbol{VSC}_i = \frac{\boldsymbol{w}.\boldsymbol{M}_i + (2 - \boldsymbol{w})\boldsymbol{D}_i}{2} \tag{6}$$

In the *DSOC Neural Network* component, the purpose of the *VSC double layer* is strengthening the discriminatory power of the model, as well as inducing proper segregation of adjacent classes, especially in models with dense class spaces, and overlapping class boundaries. This purpose is achieved through the following:

a) *VSC Selector Layer*, a VSC strength threshold is determined, where an input feature with a stronger VSC (i.e. the feature's VSC is higher than the predetermined threshold) will pass to the following layer, while weaker features are discarded and excluded from the model.
b) *VSC Scaling Layer*, this layer contains a modified version of the VSC equation, where the VSC weight variable $\boldsymbol{w}$ is not restricted to a value between [0, 2] as in the original coefficient, but could take any value greater than zero. As such, a strong variable i that has passed successfully through the previous *VSC Selector layer* is then scaled up or down by that dynamic modified coefficient ($modVSC_i$).

The *VSC double layer* results in: a) dimensionality reduction of input features, by discarding variables with weak modeling and discriminatory powers, and b) independently scale each of the remaining features. As such, adequate segregation between classes is achieved, which is then reflected in the overall classification performance of the DSOC model.

The *VSC Selector* threshold as well as the weight variable of the *VSC Scaling* layer are both determined through cross-validation procedure during the training process of the DSOC model.

The effect of *VSC double layer* on the performance of the model is discussed in section 5 Results and Discussions.

3.1.2.2. The Pooling layer. A single pooling layer consists a set of neurons (L) of size $C^n \times \alpha^n$, where C^n is the number of class members in this dimension and α^n is a scaler indicating the number of neurons that models each class member in a class space. The default value of α^n is 1, however it could be set to any value greater than or equal to 1. The optimum value of α^n is decided through cross validation routine. For example, in case of a binary class space, the number of neurons constructing this pooling layer is $2 \times \alpha^1$, i.e. the pooling layer is formed of at least two neurons ($L^1 = \{l_1, l_2\}$). Moreover, if $\alpha^1 = 3$, this means that each class is modeled by 3 neurons, and the total number of neurons forming the layer is 6 (i.e. $L^1 = \{l_1, \cdots, l_6\}$).

Each neuron (l) in the pooling layer has a weight vector (w) that connects it to the *VSC Scaling* layer. The size of w is equal to the number of neurons (M) in the *VSC Scaling* layer.

3.2. Designing the DSOC model

In order to design a *DSOC* model, certain parameters should be specified. Those parameters include; the number of classification dimensions (n), the size of each dimension (C) which is the number of classes modeled by that dimension, in addition to the value of the alpha parameter (α) per dimension.

3.2.1. Number of dimensions (n)

The number of dimensions in a DSOC model refers to the number of class spaces in the model. Where, in case of BC and MCC the number of dimensions (n) is equal to1 since those tasks have single output class space, while in case of MLC and MDC tasks, n is equal to the number of class variables (spaces) in the dataset.

Both DSOC components, the *Hypercube Classifier* and the *DSOC Neural Network*, have the same number of dimensions; therefore, by setting the value of n, we construct an *n-dimensional hypercube* as well as *n DSOC Neural Networks* connected to it.

3.2.2. Dimension size (C)

The size of dimension (C) is simply the number of class members in a specific class space (dimension), or the number of possible values a class variable can have. In binary class spaces, where the class members could be {0, 1} or {-1, 1}, then the dimension size is 2 ($C = 2$). The number of classes per dimension determines the size of the *Hypercube Classifier* as well as the minimum number of neurons in the *Pooling* layer in each of the *n DSOC Neural Networks*.

For example, in case of the 3-dimenstional *DSOC* model mentioned in section 3.1.1, the model has three classification dimensions representing (topics, languages and fields), therefore the size of every dimension depends on the number of possible classes C in that class space. As such, if the topics class space has 5 possible values, while fields and languages have 3 and 5 possible values respectively, (i.e. $C^{Topic} = 5$, $C^{Field} = 3$ and $C^{Lang} = 5$), then the *DSOC* model is formed of a 3-dimensional *Hypercube Classifier* of size ($5 \times 3 \times 5 = 75$ neuron), and 3 *DSOC Neural Networks* whose *Pooling* layers have 5 α^1, 3 α^2 and 5 α^3 neurons respectively.

3.2.3. The alpha parameter (α)

The alpha parameter α^n, is the parameter that decides the number of neurons that models each class in a particular dimension and span the variance among its members. For example, if $\alpha^2 = 10$, thus each class will be modeled by 10 neurons in the *Pooling* layer of the second *DSOC Neural Network* component. Moreover, if $C^2 = 5$ then the corresponding *Pooling* layer will have 50 neurons. As mentioned earlier, the default value of α^n is 1, but other values could be used after performing the cross validation routine during the training step.

Every neuron in the *n Pooling layers* of the model has a weights vector (w^n) of size M^n which is equal to the number of neurons in the previous *VSC Scaling* Layer. The weights of this vector are determined through the training process.

It is worth mentioning that the number of neurons in the *VSC Selector* layer is equal to the number of input neurons or features P^n in that dimension, while the number of neurons in the *VSC Scaling* Layer M^n is less than or equal to the number of neurons in the *VSC Selector* layer, i.e. $M^n \leq P^n$.

3.3. Deciding the training parameters of the DSOC model

DSOC model training parameters are those determining the rate by which the Pooling neurons weights vectors (w^n) are being updated, as well as the range of neurons to be updated in each step of training. Before training the *DSOC* model, three main training parameters should be decided, which are: the number of *iterations* or *epochs* (h), the *learning rate (R)* and the *neighborhood distance* (nD).

3.3.1. Number of iterations or epochs (h)

The number of iterations, which is also called (epochs), is the number of times by which the entire training set is presented to the *DSOC* model for training and weights update. In this research, the default value is

100.

3.3.2. Learning rate (R)

Learning rate is the rate by which the neurons weights vectors are updated at each iteration h as a function of time t. The learning rate should have a value greater than 0 and less than or equal to 1. In this study, the learning rate is initiated at ≈ 0.7–0.8 then decreases exponentially with time. It is worth mentioning that the word "time" with its notation t in this context refers to each time a training sample is presented to the cube. As such, the total "time" T of the training process equals to the product of the number of training samples and the number of iterations or epochs, as seen in equation (7).

$$T = h \times count(samples) \tag{7}$$

In this study, we propose that the value of the learning rate as a function of time t is calculated as shown in equation (8) below, which is then later used in the weights update process as per equation (10).

$$R(t) = e^{\left(-\frac{dt}{T}\right)} + 0.01 \tag{8}$$

Where, $R(t)$ is the learning rate as a function of time t and T is the total training time as calculated in equation (7) above while $e^{(\cdots)}$ is the natural exponential function (exp) and d is the decay parameter. Fig. 4 **(a)** demonstrates the effect of the decay parameter d on the learning rate. In this research we used $d = 6$ and the initial learning rate = 0.83.

3.3.3. The neighborhood distance (nD)

The neighborhood distance *nD* determines the number of neurons in the *Pooling Layers* whose weights are to be updated at each time t. This process aims at rearranging the *Pooling* neurons in a manner that reflects their class similarities in terms of features and properties.

A hand-coded decay function is applied to the *nD* as well so that the neighborhood of neurons decreases with time t. Based on experience, the model designer decides the neighborhood rate of decay as a function of time *nD(t)*. For example, the model designer can decide that the number of neighborhood neurons to be updated decreases linearly by 1 every one third of the epochs until it reaches 0.

Fig. 4 **(b)** shows the rate of decay of *nD(t)* as a function of time t. As noticed from the graph, when the initial value of *nD* is set to 1, then it reaches zero at half the total time of training (T). On the other hand, higher *nD* keeps its initial value during the first one third of the training time, and then decreases linearly until it reaches 0 at the beginning of the last third of training.

In this study, the default neighborhood distance was set to 1 (i.e. *nD = 1)* across all dimensions, using the rate of decay depicted in Fig. 4 **(b).**

3.4. Model training

DOSC model training is the process of updating the weights of the weight vectors (w^n) of all neurons in the *n-Pooling* layers of the model, and then build the probability model of the *n*-dimensional *Hypercube Classifier*. The training process aims to make every neuron of the *hypercube* capable of representing a set of dependent classes across all class spaces simultaneously. As such, the training process of the *DSOC* model is performed in two stages:

3.4.1. Updating the weights vectors (w^n)

Hereafter we present the weights update process in a specific dimension (j). The same weights update process is then repeated in all n dimensions of the model. The process of weights update reorganizes the *Pooling* neurons and bring them closer to the class they represent. As such, for any particular dimension in the *DSOC* model, the weights update process is performed as follows:

i. *Initialize the weights vectors:* For all neurons (L^j) in the *Pooling* layer, randomly initialize a single weights vector (w^j) of size M^j, which is equal to the number of neurons in the previous *VSC Scaling* layer in dimension j.
ii. *Start the first Epoch*: for each sample (i) in the training set, compute its *Euclidian distance* (ED_i^j) from the pooling weights vector. For each samples in the model, ED_i^j is computed as the distance between the *VSC Scaling neurons* (v^j) and (w^j) in the *DSOC Neural Network* of dimension j.

$$\boldsymbol{ED_i^j = SQRT}\left(\sum_{q=1}^{|M^j|}\left(v_q^i - w_q^j\right)^2\right) \tag{9}$$

iii. *Get the winning neurons*: the neuron with the least *ED* is the winning neuron (u).
iv. *Get the sub-set of neighboring neurons*: For each winning neuron get a subset of neighboring neurons according to the neighborhood distance *nD*.
v. *Update the weights of the winning neuron and its neighbors*: The weight update is done using equation (10) below for both the wining neuron and its neighbors. The magnitude of weights update depends on the learning rate of the model.

$$\boldsymbol{w^j(t+1) = w^j(t) - R(t)(v^j - w^j(t))} \tag{10}$$

vi. *Repeat until end of Epochs*: A single epoch means going through the entire training set and applying the steps of finding the winning neuron and weights update for all the samples in the training set. After finishing the first epoch, repeat the steps from ii to *v* on the

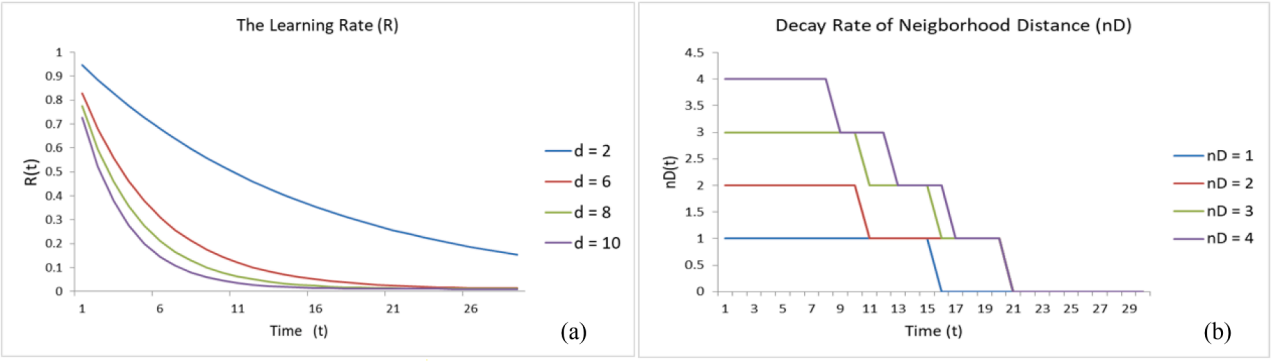

Fig. 4. DSOC training parameters. a) The learning rate *R(t)* as a function of time t, using different values for the decay parameter *d*. b) The rate of decay of the neighborhood distance *nD(t)* as a function of time.

entire training set until the maximum number of epochs h is reached.

3.4.2. Build the probabilistic model and construct the hypercube classifier

The last few steps of training the model are performed based on the Gaussian Naïve Bayes concept. The remaining steps aim at constructing the *Hypercube* classifier and populate it with the classification prior probabilities. These steps are carried out as follows:

3.4.2.1. Construct the hypercube.

i. Create n dimensional hypercube of size $\prod_{j=1}^{n} C^j$ neurons where each neuron of the cube represent a combination of classes across all dimensions of the model.

ii. For each neuron (r) in the hypercube, set its initial value to 1 (to account for class imbalance and rare combinations with no training samples).

iii. Iterate through the training samples, where the value of the hypercube neuron (r) is incremented by 1 if, and only if, the class vector (Y_i) of that sample (i) exactly matches the combination of classes of that neuron.

iv. Calculate the probability of each neuron (r) in the cube. This is considered the prior probability of that neuron given the combinations of class members of all n-class spaces.

v. Each of the hypercube neurons is connected to the *Pooling* layer of each dimension through a weight vector (b^j) of size equals to the number of neurons L^j in that *Pooling* layer. For example, in a 3-dimensional DSOC model, each neuron in the *hypercube classifier* has three weights vector (b^1, b^2 and b^3) of size L^1, L^2 and L^3 respectively (see Fig. 5). The members of (b^j) vector are the prior probabilities of neurons L^j given a class member $\{c \in C^j\}$.

3.4.2.2. Build the probabilistic model. For every neuron (l) of the L^j neurons of the *Pooling Layer* of dimension j, do the following:

i. Compute the standard deviation and mean *ED* for all samples belonging to each class of the dimension class space (C^j).

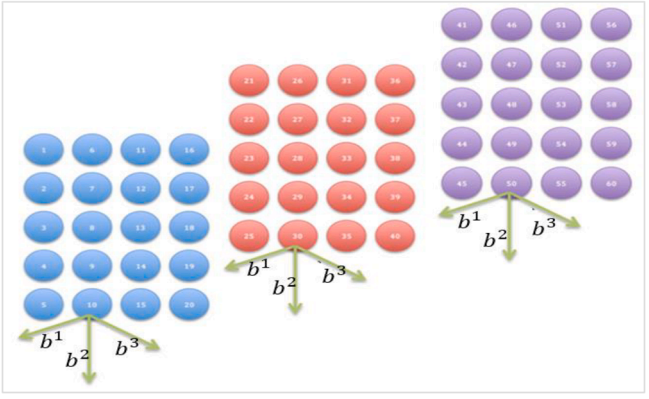

Fig. 5. A DSOC *Hypercube classifier* of size 5 × 4 × 3 sliced along the C^3 dimension into 3 planes of size 5 × 4. Every neuron has three weights vectors (b^1, b^2 and b^3) connected to the *Pooling* layers of the three DSOC Neural Networks. Each neuron of the *Hypercube* represent a combination of classes from the three class spaces of the model.

ii. Compute the probability when this neuron was activated, given each class in (C^j) as per equation (11). This is considered the prior probability of that neuron given a specific class.

$$p\left(l_k|C_q^j\right) = \frac{\sum active(l_k|C_q^j)}{\sum samples|C_q^j} \tag{11}$$

iii. These three values are used during the classification process of new samples.

3.5. Classify future samples

When a new sample (i) is introduced to the DSOC, the classification process is performed in the following steps in each dimension j of the n dimensional model:

i. Sample features are presented to the *VSC double layer* of the DSOC Neural Network j, where sample features that have registered weak VSC during the training process are discarded by the *VSC Selector* layer and only strong features pass through to the *VSC Scaling* layer for feature independent scaling.
ii. The surviving neurons are then presented to the *Pooling layer* of dimension j, where the ED_i^j is computed between the (v^j) and (w^j) as per equation (9).
iii. For each neuron (l^j) of the L^j neurons, use the computed ED_i^j as well the mean and standard deviations computed in 3.4.2.2 to compute the likelihood of that neuron, $Likelihhod\left(l_k^j \middle| c_q^j\right)$, given each class c_q where $\{c_q \in C^j\}$.
iv. The previous steps (*i* to *iii*) are repeated for all n dimensions in the model.
v. As per equation (12), for every neuron (r) in the *Hypercube Classifier*, Compute the sum of the natural log of its vector (b^j), which is the likelihoods computed for the *Pooling layer* in the previous step as well as the prior probabilities computed in equation (11), given the class represented by that neuron in each dimension (c_q^j). Add the result to the natural log of the neuron's prior probability computed in 3.4.2.1 to calculate the probability $p(r)$ that this neuron is activated.

$$p(r) = \sum_{i=1}^{j}\sum_{k=1}^{L} \ln(Likelihhod\left(l_k^j|c_q^j\right)) + \sum_{i=1}^{j}\sum_{k=1}^{L} \ln(p\left(l_k|C_q^j\right)) + \ln(prior(r)) \tag{12}$$

vi. Compare the computed $p(r)$ for all neurons in the Hypercube Classifier, where the neuron with the highest $p(r)$ is the winning neuron.
vii. Emit the classification vector Y_r related to the winning neuron r, whose members are the classes represented by the winning neuron, $Y_r = [y^1, y^2, \cdots, y^n]$

As such, a full classification across the n dimensions of the MDC task was performed simultaneously. In this setup, the *Hypercube classifier* is capable of classifying class combinations that were not presented to the model during the training process, due to the bond it creates among various class spaces in the model through discrete yet connected probabilities.

4. Experiments

The aim of the study is to propose a deep learning classifier that can be used for multi-output learning problems as well as for the single-output learning ones. The experiment was designed to evaluate the performance of the proposed DSOC model in the four types of

classification problems discussed earlier: BC, MCC, MLC and MDC.

As such, we conducted an empirical assessment on a variety of benchmark datasets, in order to compare the proposed model to standard classifiers. The proposed DSOC was then challenged with competitive state-of-the-art techniques reported in literature.

4.1. Benchmark datasets

To evaluate the performance of the proposed model, DSOC was tested on seventeen benchmark datasets frequently used in literature to evaluate the performance of different classification models. The datasets were chosen to cover different aspects of classification tasks that are tackled by the proposed DSOC model. Those aspects include multiple dimensions, multi-class variables and labels per dimension, imbalance of labels, etc. The datasets are divided into four categories according to the type of the classification problem, where four datasets were chosen for BC problems, three for MCC, four for MLC and six for MDC.

Table 1 summarizes the main characteristics of the chosen datasets, as the number of observations and features, number of class spaces (i.e. number of dimensions), number of class labels per dimension, as well as their classification task category.

4.2. Parameter selection

For the datasets that were not initially divided into *Training* and *Test* sets, a random selection technique was applied to divide the observations of the original dataset set into *Training* and *Test* sets in 75:25 ratio. Then a tenfold cross validation (CV) technique was employed on the training set to decide the DSOC model parameters. For the sake of simplicity, each of the general DSOC parameters (neighborhood, epochs and the decay parameter) was set to a constant value throughout the experiment ($nD = 1$, $h = 100$ and $d = 6$).

On the other hand, for the dimension specific parameters (*VSC Scaling* parameter, *VSC Selector* and α) the aforementioned tenfold CV was applied to decide those three parameters for each of the model multi-dimensional space.

4.3. Evaluation metrics

In this paper, three widely used classification evaluation metrics are used to evaluate the performance of the proposed model. In BC problem, the *F1* measure is used as shown in equation (13). Where TP, FP and FN are True Positive, False Positive and False Negative respectively and are used to calculate the recall and precision as in (Tsoumakas & Katakis, 2007).

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Or,

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{13}$$

While in MLC and MDC tasks, *Hamming Accuracy* (HAccuracy) and *Exact Match* (EM) are used for evaluation. The *Hamming Accuracy* is the average of classification accuracy of all class variables in all dimensions, and it is used as well in MCC task. On the other hand, the *Exact Match* is the accuracy of predicting the entire classification vector across all class spaces simultaneously (Wang, et al., 2017), (Wang, et al., 2020).

$$\boldsymbol{HAccuracy} = \frac{1}{\boldsymbol{N}} \sum_{i=1}^{N} \frac{\left|\boldsymbol{Y}_i \cap \boldsymbol{Y}_i^*\right|}{\boldsymbol{L}} \tag{14}$$

$$\boldsymbol{EM} = \frac{1}{\boldsymbol{N}} \sum_{i=1}^{N} \boldsymbol{f}(\boldsymbol{Y}_i, \boldsymbol{Y}_i^*) \tag{15}$$

Where, N is the number of observations, and n is the number of dimensions in the model. Y_i is the actual classification vector of size L, while Y_i^* is the predicted classification vector of the same size and $|Y_i \cap Y_i^*|$ indicates the number of intersections between both vectors. On the other hand, $f(Y_i, Y_i^*)$ is a function that returns 1 when both classification vectors exactly match, and 0 otherwise.

4.4. Performance assessment

In this paper, the classification performance of the proposed DSOC

Table 1
The characteristics of the benchmark datasets.

Classification Category	Dataset	Number of Observations	Number of Features	Number of Dimensions	Number of Classes per Dimension	Domain and Reference
BC	Diabetes1	768	8	1	2	Medical (Smith, Everhart, Dickson, Knowler, & Johannes, 1988)
	Sonar2	208	60	1	2	Signal (Gorman, Sejnowski, & J., 1988)
	Banknote3	1372	4	1	2	Forensic (Dua & Graff, UCI Machine Learning Repository, 2019a)
	Ionosphere[2]	351	34	1	2	Signal (Sigillito, Wing, Hutton, & Baker, 1989)
MCC	Iris[3]	150	4	1	3	Plant (Fisher, 1936)
	Seeds[2]	210	7	1	3	Plant (Charytanowicz, et al., 2010)
	Abalone[3]	4177	8	1	28	Marine (Dua & Graff, UCI Machine Learning Repository, 2019b)
MLC	Scene[2]	2407	294	6	2	Image (Boutell, Luo, Shen, & Brown, 2004)
	Emotions[2]	593	72	6	2	Music (Trohidis, Tsoumakas, Kalliris, & Vlahavas, 2008)
	Yeast[2]	2417	103	14	2	Genes (Elisseeff & Weston, 2001)
	Image[2]	2000	135	5	2	Image (Zhang & Zhou, ML-KNN: A lazy learning approach to multi-label learning., 2007)
MDC	Solar Flare[3]	315	10	3	3,4,2	Astronomy (Dheeru & Taniskidou, 2017)
	Edm[2]	154	16	2	3	Machinery (Karalič & Bratko, 1997)
	WaterQuality[2]	1060	16	14	4	Life form (Džeroski, Demšar, & Grbović, 2000)
	WQplants[2]*	1060	16	7	4	Plants (Džeroski, Demšar, & Grbović, 2000)
	WQanimals[2]*	1060	16	7	4	Animals (Džeroski, Demšar, & Grbović, 2000)
	Enb[2]	768	6	2	2,4	Energy (Tsanas & Xifara, 2012)

* WQplants and WQanimals are subsets of the parent Water Quality dataset used to predict the relative representation of plant and animal species in Slovenian rivers.
[1] https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.
[2] https://www.openml.org/.
[3] https://archive.ics.uci.edu/ml/index.php.

model was assessed by comparing the obtained results to that obtained by standard classification algorithms, as well as state-of-the-art MDC approaches. All techniques were applied on the same benchmarks datasets.

For the single-output learning tasks, BC and MCC, the DSOC classification performance was assessed against four standard classification algorithms, namely k-Nearest Neighbor (*k*NN), Support Vector Machine (SVM), Naïve Bayes (NB) and Decision Tree (DT).

However, for the multi-output learning tasks, MLD and MDC, the DSOC performance was compared to eleven different approaches, divided into five categories that span the full classification challenges:

a) Standard Classification Algorithms: the same standard methods mentioned above.
b) Approaches based on manipulating the class spaces: *Binary Relevance* (BR), which divides the task into independent binary classifiers. *Ensembles of Class Powersets* (ECP) creates a combination of labels and treat the entire set of combinations as a single-dimension single-label class space. *Ensembles of Classifier Chains* (ECC), which divides the task into a chain of multi-class independent classifiers. *Ensembles of Super Class Classifiers* (ESC), which divides the classification space into sections of super-classes, formed of combination of subsets of class spaces. (Read, Bielza, & Larrañaga, Multi-dimensional classification with super-classes., 2013)
c) Approaches based on feature augmentation: those approaches tend to manipulate the input feature spaces rather than the output ones. This category includes, KRAM and SFAM (mentioned earlier in the literature review section).
d) Dependency modeling approach, the *Stacked Dependency Exploitation* (SEEM), which uses deterministic strategy to model class dependencies instead of the widely used probabilistic models. (Jia & Zhang, Multi-dimensional classification via stacked dependency exploitation., 2020b)
e) Regression based approach, gMML that creates independent regression models for each class variable and then, uses the Mahalanobis distance in the final classification step. (Ma & Chen, 2018)

The results obtained by applying those approaches on the selected benchmark datasets were extracted from literature, and used to assess the performance of the proposed DSOC model.

5. Results and discussions

As mentioned earlier, the aim of the study is to construct a deep learning classifier that fits both single and multi-output learning tasks, with no structural alteration in the core design of the model other than simple parameters tweaking. In addition, the proposed design should be capable of successfully classifying observations in MLC and MDC tasks while modeling the class dependencies across all dimensions. Moreover, the model was challenged with datasets suffering from class imbalance in its multi-class spaces to assess its ability to classify, successfully, future rare observations.

A separate DSOC model was built for each of the 17 datasets. The DSOC dimensional parameters were determined using 10 fold CV applied on the training set. Table 2 presents the selected parameters per dataset as well as the number of features that passed through the *VSC Selector* layer. The detailed parameters per dimension for one of the datasets, "*Image*" dataset, are presented in Table 3.

As seen in the last column of Table 2, the *VSC Selector* layer conducted a feature reduction routine that resulted in a substantial reduction in number of features in most of the models. The *Ionosphere* dataset was subjected to a sever dimensionality reduction, where only 35% of the original features have passed through the *VSC Selector* layer, while in case of *Image* and *Seeds* datasets only 52% and 57% have passed through that layer respectively. The other datasets were subjected to the same routine resulting in various levels of dimensionality reduction. On the other hand, six datasets have retained their full number of input features along all dimensions.

The *VSC Selector* layer serves two purposes, a) maximizing the DSOC's modeling and discriminatory power among classes in each of the class spaces, as well as b) reducing the size of input spaces by discarding the features that will inversely affect the classification power of the model. Reducing the input space is reflected in less computational power and time needed to train and implement the model as well as in maximizing the DSOC's discriminatory and modeling powers.

The effect of the *VSC Selector* layer is further augmented by the next layer, the *VSC Scaling layer*, which aims at deepening the discriminatory differences among classes along the remaining features.

Table 3
Values of the DSOC parameters for the "*Image*" dataset along the model 5 dimensions ($L = 5$).

Parameter	L_1	L_2	L_3	L_4	L_5
VSC Selector	0.6526	0.6383	0.9571	0.8554	0
modVSC	2.5	2	5	2	7.5
α	1	4	4	4	2
W	86	88	10	32	135

Table 2
DSOC Model parameters for each of the benchmark dataset obtained by applying 10 fold CV.

Classification Category	Dataset	VSC Selector	Features Neurons	VSC Scaling Parameter	α	Selected Features after VSC
BC	Diabetes	0	8	12.5	2	100%
	Sonar	0.6123	46	5.0	3	77%
	Banknote	0.735	3	5.5	3	75%
	Ionosphere	0.891	12	5.0	1	35%
MCC	Iris	0	4	10.5	1	100%
	Seeds	0.5383	4	2.5	1	57%
	Abalone	0	8	1.0	1	100%
MLC*	Scene	[0 – 0.7717]	[271 – 294]	[1 – 12]	[1 – 10]	98%
	Emotions	0	72	[1 – 5.5]	[1 – 3]	100%
	Yeast	[0–0.928]	[7–103]	[1 – 10.5]	[1 – 10]	76%
	Image	[0–0.9571]	[10–135]	[2–7.5]	[1 – 4]	52%
MDC	Solar Flare 1	0	10	1,1.5,1	1	100%
	Edm	0.3398,0	9,16	1,1.5	3,4	78%
	WaterQuality	0	16	[1–3.5]	[1–3]	100%
	WQplants*	[0–0.7375]	[4–16]	[1–2]	[1–2]	70%
	WQanimals*	[0–0.8059]	[10–16]	[1–4.5]	[1–2]	82%
	Enb*	0	6	1,3.5	1,3	100%

* Values between brackets are the range of values of the DSOC parameter along the model's dimensions. For example, in the *Scene* dataset, the VSC Scaling parameter has a range of values of [1–12], which represents the range of values along the model's 6 dimensions whose values were $modVSC_1 = 1$, $modVSC_2 = 1$, $modVSC_3 = 12$, $modVSC_4 = 10.5$, $modVSC_5 = 8$, $modVSC_6 = 1$.

In datasets with dense class spaces, classes are not clearly separated, but might suffer from sever overlap of their class spheres. Such density and overlap could be noticed in the tight Euclidian distances and the high similarity/correlation between classes that might, in some cases, reaches full correlation. DSOC model tackles this problem through its VSC double layer, *Selector* and *Scaling* layers, where both layers tend to intensify the discriminatory power of variables that, in turn, results in widening the distances between classes and relatively break the correlation among them.

Fig. 6 demonstrates the effect of applying the VSC double layer on the "*Image*" dataset. The substantial reduction in dimensionality and the noticeable strengthening of the discriminatory power of the remaining input variables have resulted in widening the distance between classes from 0.0060 to 0.0191, and breaking the tight correlation from 0.9643 down to 0.7242, i.e. 313% and 25% improvement in class separation respectively.

In a closer look at Fig. 6 **(a)**, the input features within the range [18–30] have the worst adverse effect on the model's discriminatory power. Such an effect was detected by the *VSC Selector* layer, which applied a threshold of **0.6383**, where all features with VSC weaker than that threshold were discarded. As such, and as seen in subfigure **(b)**, only 88 features have passed the *VSC Selector* layer, causing a significant dimensionality reduction, where only 65% of the original input features have survived the *Selector*. In order to deepen the difference between classes, the next layer, the *VSC Scaling* layer, has applied a dynamic scaling factor (based on its $modVSC_2 = 2$) ranging from 0.2654 to 0.9578 on the remaining 88 features, which results in expanding the ED between both classes by 313%.

Moreover, applying the VSC double layer on the "*Image*" dataset has increased the classification *Hamming* accuracy from 0.7100 to 0.8325, while the *Exact Match* metric has improved from 0.3198 to 0.3750. These results reflect the importance of the VSC double layer in improving the overall model accuracy as well as its ability to predict the exact classification vectors. Table 4 presents the effect of applying the VSC double layer on the seventeen DSOC models constructed in this study, in terms of *HAccuracy* and *EM*. In addition, Fig. 7 visualizes the results presented in the table, demonstrating that the VSC double layer has enhanced the classification performance of the DSOC model in all of the four classification tasks.

For evaluating of the proposed DSOC, comparisons have been conducted to the DSOC classification performance against that of standard classifiers as well as state-of-the-art MDC approaches on the benchmark datasets. Those comparisons are presented in the following three subsections.

Table 4
Effect of the VSC double layers on the performance of the DSOC Model applied on all benchmark datasets in the four classification tasks.

Classification Category	Datasets	Hamming Accuracy		Exact Match	
		VSC double layer Applied	*Without VSC double layer*	*VSC double layer Applied*	*Without VSC double layer*
BC	Diabetes	0.7708	0.6471	–	–
	Sonar	0.7692	0.5952	–	–
	Banknote	0.9854	0.9489	–	–
	Ionosphere	0.8740	0.7571	–	–
MCC	Iris	0.9867	0.9000	–	–
	Seeds	0.9143	0.8810	–	–
	Abalone	0.3599	0.2491	–	–
MLC	Scene	0.9398	0.8312	0.5334	0.4749
	Emotions	0.7463	0.5998	0.2178	0.0941
	Yeast	0.7719	0.7501	0.1538	0.1298
	Image	0.8325	0.7100	0.3750	0.3198
MDC	Solar Flare 1	0.9418	0.9101	0.8846	0.7460
	Edm	0.8000	0.7667	0.6333	0.6333
	WaterQuality	0.6405	0.5893	0.0047	0.0000
	WQplants	0.6604	0.6220	0.1038	0.0660
	WQanimals	0.6462	0.4993	0.0849	0.0142
	Enb	0.9019	0.8497	0.8823	0.7516

5.1. Comparison to standard classifiers

The experimental results are reported in details in four tables, one for each of the classification tasks. Table 5 for BC, Table 6 for MCC, Table 7 for MLC and Table 8 for MDC. Based on the results obtained in these tables, we can point out the following observations:

- For the standard classification techniques (kNN, NB, SVM and DT), 112 experiments were carried out. Those experiments are composed of 5 standard algorithms, in addition to the proposed DSOC, 2 evaluation metrics and 17 datasets.
- Those experiments are divided across the four classifications tasks as follows: 20 experiments in the BC task, 16 in MCC, 32 in MLC and 48 in MDC.
- As seen in the aforementioned tables (with the winning results embossed in bold), the proposed DSOC model has outperformed the standard methods in 50% of the cases in the BC task, 100% in MCC, 75% in MLC and 83% in case of MDC. Moreover, Fig. 8 **(a)** shows the accuracy results of the proposed DSOC in comparison with the other standard classifiers.
- With the exception of the "*EnB*" and "*Yeast*" datasets, the DSOC has outperformed all other standard algorithms in MLC and MDC, in terms of the *Exact Match* metrics. Since the EM metric reflects the

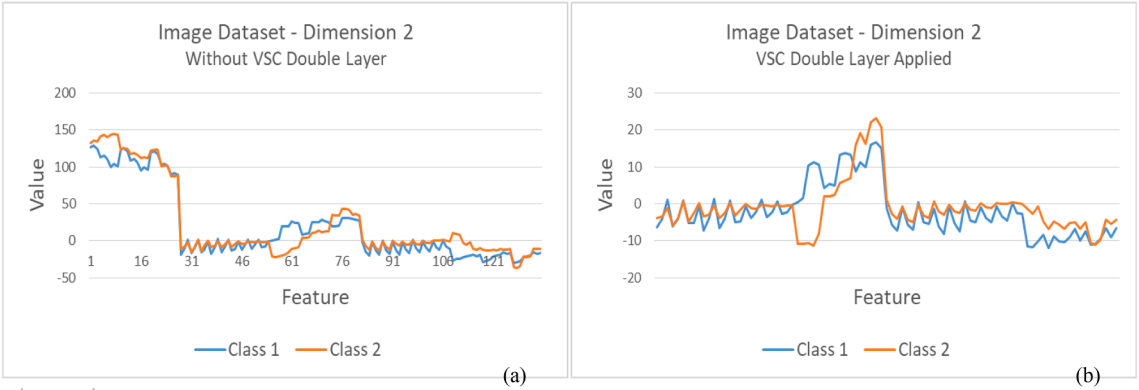

Fig. 6. The effect of applying the VSC double layers on both classes of the second dimension of the Image dataset, a) before applying the VSC double layer (135 features and 0.9643 correlation coefficient), b) after applying both the VSC Selector and VSC Scaling layers (88 remaining features and 0.7242 correlation coefficient).

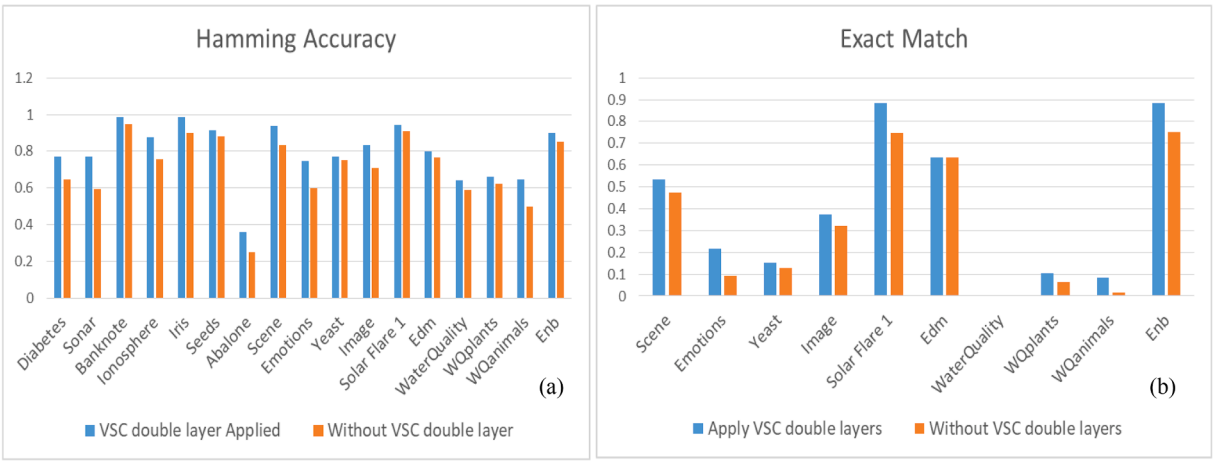

Fig. 7. The effect of the VSC double layer on the performance of the DSOC Model applied on all benchmark datasets, a) Hamming Accuracy on all 17 datasets, b) the Exact Match metric applied on the datasets of MLC and MDC tasks only.

Table 5
Comparing the classification results of DSOC against standard classification algorithms applied on BC benchmark datasets.

Dataset	F1 Measure				
	kNN	SVM	NB	DT	***DSOC***
Diabetes	0.7396	**0.7760**	0.7552	0.6979	0.7708
Sonar	**0.8077**	0.8077	0.6923	0.6538	0.7692
Banknote	0.9708	0.9854	0.8321	0.9635	**0.9854**
Ionosphere	0.8000	0.8429	0.8286	0.8143	**0.8740**

Table 6
Comparing the classification results of DSOC against standard classification algorithms applied on MCC benchmark datasets in terms of Hamming Accuracy.

Dataset	Hamming Accuracy			
	kNN	NB	DT	***DSOC***
Iris	0.9067	0.9333	0.9467	**0.9867**
Seeds	0. 8952	0.8857	0. 8952	**0.9143**
Abalone	0.1657	0.1897	0.1933	**0.3599**

ability of the model to classify successfully the instance along all dimensions, therefore the proposed DSOC is more capable of classifying the observations across all class spaces simultaneously. A comparison of the EM accuracy of the proposed DSOC and the other standard classifiers are shown in Fig. 8 **(b).**

- The last mentioned observation points out the intrinsic ability of the DSOC to span and to model dependencies among class variables across different dimensions.

5.2. Comparisons to the state-of-the-art methods

In order to assess the ability of the proposed DSOC algorithm to model complex dependencies as well as imbalanced class spaces, the obtained results were challenged with state-of-the-art approaches from literature. The selected approaches span different approaches for solving classification tasks, these include approaches that depends on manipulating the class spaces during training (BR, ECC, ECP and ESC), approaches that uses feature augmentation to manipulate the input space (KRAM and SFAM), in addition to SEEM approach that model dependencies and gMML that uses regression approach to solve the classification task.

The results of the selected approaches were collected from recent

Table 7
Comparing the classification results of DSOC against standard classification algorithms applied on MLC benchmark datasets in terms of Hamming Accuracy and Exact Match metrics.

Dataset	Hamming Accuracy				Exact Match			
	kNN	NB	DT	***DSOC***	kNN	NB	DT	***DSOC***
Scene	0.8913	0.7565	0.8478	**0.9398**	0.5008	0.1756	0.3687	**0.5334**
Emotions	0.7046	0.7302	0.7112	**0.7463**	0.1931	0.1683	0.1436	**0.2178**
Yeast	**0.7803**	0.6991	0.7184	0.7719	**0.2061**	0.1036	0.0523	0.1450
Image	0.7925	0.7235	0.7435	**0.8325**	0.3329	0.1725	0.2350	**0.3750**

Table 8
Comparing the classification results of DSOC against standard classification algorithms applied on MDC benchmark datasets, in terms of Hamming Accuracy and Exact Match metrics.

Dataset	Hamming Accuracy				Exact Match			
	kNN	NB	DT	***DSOC***	kNN	NB	DT	***DSOC***
Solar Flare 1	0.9103	0.8974	0.8846	**0.9418**	0.8333	0.8462	0.859	**0.8846**
Edm	0.6833	0.6667	0.650	**0.80**	0.433	0.4667	0.4333	**0.6333**
Water Quality	0.6179	0.4481	0.5236	**0.6405**	0	0	0	**0.0047**
WQplants	0.6301	0.4023	0.5553	**0.6604**	0.0755	0	0.0472	**0.1038**
WQanimals	0.6186	0.4434	0.5802	**0.6462**	0.0660	0	0.0556	**0.0849**
Enb	**0.9935**	0.9412	0.9869	0.9019	**0.9869**	0.9150	0.9739	0.8823

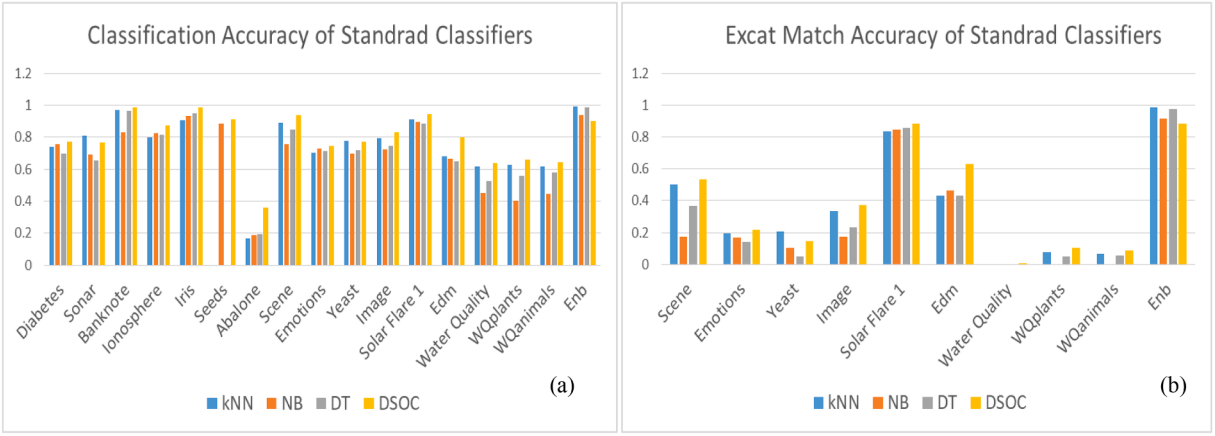

Fig. 8. The classification performance of DSOC model compared to standard classifiers. a) Classification accuracy for BC, MCC, MLC and MDC datasets. b) Exact Match accuracy for MLC and MDC datasets.

literature ((Jia & Zhang, Multi-dimensional classification via stacked dependency exploitation., 2020b), (Jia & Zhang, Multi-dimensional classification via kNN feature augmentation., 2020a) and (Jia & Zhang, Multi-dimensional classification via selective feature augmentation., 2022)) to reflect the most recent work applied on the MLC and MDC benchmark datasets. It is worth mentioning that KRAM and the ensembles approaches are all based on Naïve Bayes classifiers.

Table 9 and Fig. 9 depict the comparison between the proposed DSOC model and eight state-of-the-art approaches in terms of the *HAccuracy*. On the other hand, Table 10 and Fig. 10 present the same comparison in terms of *EM* accuracy to assess the model ability to model dependencies. From those tables and figures, one can observe the following:

- In terms of *Hamming Accuracy*, DSOC has outperformed all state-of-the-art approaches in five datasets (55.6%), and achieved equivalent results in three datasets (33.3%).
- In terms of *Exact Match*, DSOC has outperformed all state-of-the-art approaches over seven datasets (77.8%), which reflects the superior ability of DSOC to model dependencies and successfully classify samples across all dimensions simultaneously.

5.3. Collective performance remarks

By analyzing the overall results obtained under sections 5.1 and 5.2, we can conclude the following:

a. Despite the imbalance of class labels in MLC and MDC datasets (especially those with high dimensionality and variety of class labels), DSOC was capable of achieving better results than most standard classifiers as well as state-of-the-art approaches included in the study.
b. The better classification performance achieved by the DSOC implies its ability to segregate different classes within all dimensions simultaneously.
c. The superior performance of DSOC over other approaches in terms of EM indicates that the model is not affected by dependencies that exist among different classification dimensions as well as class imbalance.
d. Moreover, the better performance of DOSC as compared to *Stacked Dependency Exploitation* (SEEM), which uses deterministic strategy to model class dependencies, reflects the intrinsic ability of DSOC's

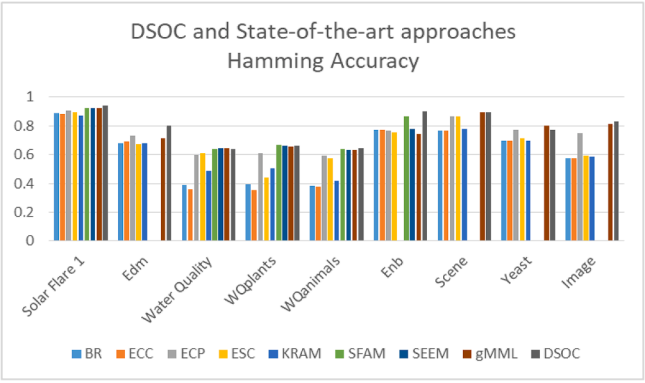

Fig. 9. Comparison of the Hamming Accuracy of DSOC vs state-of-the-art approaches applied on MLC and MDC benchmark datasets.

Table 9
Comparing the classification results of DSOC against state-of-the-art techniques applied on the same MLC and MDC benchmark datasets in terms of the Hamming Accuracy.

Dataset	Hamming Accuracy								
	Classes Based Approaches				*Feature Augmentation*		*Dependency Modelling*	*Regression Based*	
	BR	*ECC*	*ECP*	*ESC*	*KRAM*	*SFAM*	*SEEM*	*gMML*	***DSOC***
Solar Flare 1	0.886	0.883	0.908	0.896	0.872	0.925	0.925	0.923	**0.9418**
Edm	0.677	0.690	0.731	0.674	0.680	–	–	0.714	**0.8000**
Water Quality	0.389	0.360	0.599	0.609	0.488	0.641	**0.646**	0.643	*0.6405*
WQplants	0.397	0.353	0.607	0.442	0.506	**0.666**	0.661	0.655	0.6604
WQanimals	0.381	0.377	0.590	0.577	0.419	0.641	0.635	0.630	**0.6462**
Enb	0.774	0.773	0.764	0.754	–	0.865	0.777	0.742	**0.9019**
Scene	0.763	0.767	0.867	0.866	0.777	–	–	**0.893**	0.8913
Yeast	0.699	0.696	0.773	0.716	0.695	–	–	**0.800**	0.7719
Image	0.573	0.576	0.746	0.593	0.586	–	–	0.811	**0.8325**

Table 10
Comparing the classification results of DSOC against state-of-the-art approaches applied on the same MLC and MDC benchmark datasets in terms of the Exact Match.

Dataset	Exact Match								
	Classes Based Approaches				*Feature Augmentation*		*Dependency Modelling*	*Regression Based*	
	BR	*ECC*	*ECP*	*ESC*	*KRAM*	*SFAM*	*SEEM*	*gMML*	***DSOC***
Solar Flare 1	0.774	0.774	0.790	0.780	0.756	0.824	0.818	0.821	**0.8846**
Edm	0.432	0.451	0.554	0.432	0.445	–	–	0.487	**0.6333**
Water Quality	0.000	0.000	0.008	0.002	0.000	0.009	0.009	**0.006**	0.0047
WQplants	0.001	0.001	0.034	0.001	0.036	0.100	0.096	0.092	**0.1038**
WQanimals	0.004	0.007	0.020	0.024	0.008	0.058	0.049	0.062	**0.0849**
Enb	0.548	0.546	0.529	0.508	–	0.729	0.554	0.554	**0.8823**
Scene	0.177	0.181	0.550	**0.541**	0.198	–	–	0.457	0.5334
Yeast	0.095	0.102	0.203	0.110	0.115	–	–	0.134	**0.1450**
Image	0.069	0.069	0.285	0.069	0.074	–	–	0.289	**0.3750**

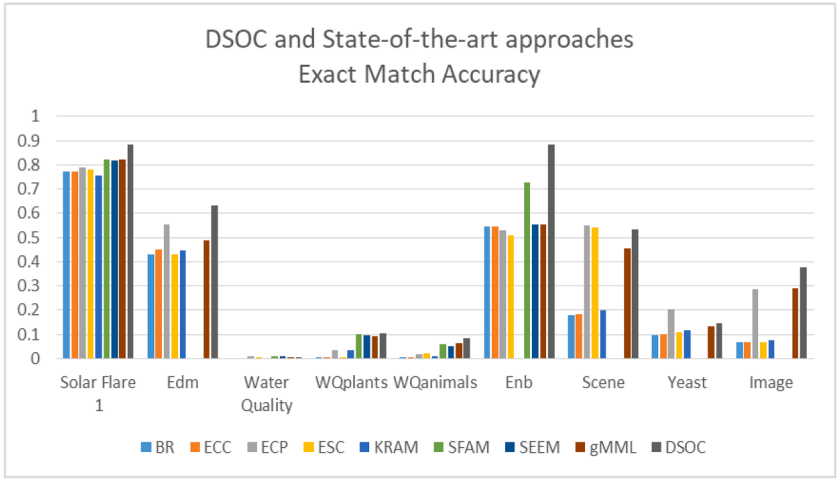

Fig. 10. Comparison of the Exact Match Accuracy of DSOC vs state-of-the-art techniques applied on MLC and MDC benchmark datasets.

Hypercube and layers to model semantics and dependencies among classes.

e. In contrast to other approaches, which require specific steps to model decencies or handle class imbalance, DSOC uses a single straightforward approach that tackles those problems simultaneously in a single learning routine without prior treatment or algorithms.

From those four conclusion remarks, the superior performance of DSOC could be attributed to the following design characteristics of the proposed model:

a. The VSC double layer (*VSC Selector* and *VSC Scaling* layers) is responsible for segregation of classes and deepening the differences between them, which is then reflected in a better discriminatory power and superior classification performance.
b. The DSOC's final classifier hypercube and its self-organizing structure, is responsible for modeling class dependencies and classify unseen class combinations in future samples of imbalanced datasets.
c. Moreover, the straightforward learning approach of DSOC provides it with the ability to solve tasks in different domains regardless the intensity of class dependencies or the severity of data imbalance.

6. Conclusion

Multi-dimensional classification (MDC) task can be considered the most inclusive description of all classifications tasks, as it joins multiple class spaces and their multiple class membership into a single compound classification problem. The challenges in MDC arise from the possible dependencies that classes in different class spaces could have, as well as the imbalance of labels in training datasets due to the enormous number of combinations needed for all labels across multiple class spaces. In this paper, we proposed an MDC deep learning classifier, named "*Deep Self-Organizing Cube*" or "*DSOC*", which could model dependencies among classes in multiple class spaces, while consolidating its ability to classify combinations of labels that were not presented to the model during the training process. *DSOC* achieved its intended purpose through its two connected components, the *n*-dimensional *Hypercube classifier*, and *n DSOC Neural Networks* connected to that hypercube. Each of the *n* DSOC's neural networks consists of three main hidden layers. The *VSC Selector, VSC Scaling* layers are responsible for feature selection and segregation of classes while the *Pooling* layer is responsible for developing the probability model per dimension. While, the central *Hypercube classifier* is responsible for creating the semantics among multiple class spaces and accommodate the model for rare samples classification.

DSOC model is a *multiple-output learning* algorithm, where it classifies a sample along multiple class spaces simultaneously, through emitting a classification vector formed of a class label from each class space in the model. As such, DSOC can be used in multiple output classification tasks, as MDC and MLC, as well as in single-output learning, as in BC and MCC problems.

For challenging the proposed DSOC model, an experiment was designed to evaluate the performance of the proposed DSOC model in the four types of classification problems BC, MCC, MLC and MDC. We conducted an empirical assessment of the model on seventeen benchmark datasets, and the obtained classification results were compared to those obtained by four standard classifiers as well as by eight competitive state-of-the-art approaches reported in literature. The selected state-of-the-art approaches were chosen to challenge the model in solving the aforementioned MDC problems. These include approaches that manipulate the input feature space through feature augmentation, and

approaches that manipulate the output class spaces to tackle the problem of dependencies, in addition to others based on regression or pure dependency modelling.

The DSOC has achieved superior performance over both standard classifiers as well as the state-of-the-art approaches in all the four classification tasks. In case of standard classifiers, DSOC has outperformed the standard methods in 50% of the cases in the BC task, 100% in MCC, 75% in MLC and 83% in case of MDC. Moreover, with the exception of only two datasets, DSOC has outperformed the standard algorithms in terms of the *Exact Match* accuracy metric, which measures the ability of the model to, successfully; classify a sample across all dimensions simultaneously.

In the same context, in terms of *Exact Match*, DSOC has outperformed all state-of-the-art approaches in 77.8% of the cases, which reflects the superior ability of DSOC to model dependencies and successfully classify samples across all dimensions simultaneously.

In contrast to other state-of-the-art approaches, the DSOC's structure, components and design do not change from one classification task to the other. DSOC maintains its straightforward training and classification technique regardless of the type of classification task. In other words, the model's straightforward approach in both training and implementation phases, regardless the nature of the classification task or dataset, makes it a *domain and task agnostic* classification model.

As such, the main contribution of this study is proposing a straightforward yet efficient deep learning classifier of superior performance, which does not require structural modifications regardless the task in hand. A model that manipulates the input features space, via its VSC layers, to increase the discriminatory power of the model and segregate classes through augmenting the differences between different input variables. A model that is not affected by the existing dependencies among classes along different classification spaces, due to the intrinsic capabilities of the *Hypercube* in modeling semantics, even in case of data imbalance.

7. Future work

Future research using the DSOC model could tackle the following points: a) Increase the number of *Pooling layers* of the DSOC neural networks in order to achieve features extraction at various abstract level, and study its effect on the overall classification performance of the model. b) Study the applicability and impact of adding additional *VSC double layer* after the *Pooling layers*, in order to achieve more class segregation as well as better modeling of dependencies across different class spaces.

CRediT authorship contribution statement

Ahmed Abdelfattah Saleh: Conceptualization, Methodology, Data curation, Validation, Software, Visualization, Writing – original draft. **Li Weigang:** Methodology, Writing – review & editing, Project administration, Supervision.

Declaration of Competing Interest

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

Data availability

Data will be made available on request.

Acknowledgment

This work has been partially supported by Brazilian National Council for Scientific and Technological Development (CNPq), under the grant number 309545/2021-8.

References

Ameer, I., Bölücü, N., Siddiqui, M. H., Can, B., Sidorov, G., & Gelbukh, A. (2023). Multi-label emotion classification in texts using transfer learning. *Expert Systems with Applications, 213*.

Arias, J., Gamez, J. A., Nielsen, T. D., & Puerta, J. M. (2016). A scalable pairwise class interaction framework for multidimensional classification. *International Journal of Approximate Reasoning, 68*, 194–210.

Bogatinovski, J., Todorovski, L., Džeroski, S., & Kocev, D. (2022). Comprehensive comparative study of multi-label classification methods. *Expert Systems with Applications, 203*, 1–18.

Boutell, M. R., Luo, J., Shen, X., & Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition, 37*(9), 1757–1771.

Brereton, R. (2003). *Chemometrics: Data Analysis for the Laboratory and Chemical Plant*. Chichester: Wiley.

Charte, F., Rivera, A. J., del Jesus, M. J., & Herrera, F. (2015). Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing, 163*, 3–16.

Charytanowicz, M., Niewczas, J., Kulczycki, P., Kowalski, P. A., Łukasik, S., & Żak, S. (2010). Complete gradient clustering algorithm for features analysis of x-ray images. *Information Technologies in Biomedicine* (pp. 15-24). Berlin: Springer.

Chen, Y. N., Weng, W., Wu, S. X., Chen, B. H., Fan, Y. L., & Liu, J. H. (2021). An efficient stacking model with label selection for multi-label classification. *Applied Intelligence, 51*(1), 308–325.

De Handschutter, P., Gillis, N., & Siebert, X. (2021). A survey on deep matrix factorizations. *Computer Science Review, 42*, 1–18.

Dembszynski, K., Waegeman, W., Cheng, W., & Hüllermeier, E. (2010). *On label dependence in multilabel classification*. Haifa: Multi-Label Data.

Dheeru, D., & Taniskidou, E. K. (2017). UCI machine learning repository. *http://archive.ics.uci.edu/ml/datasets/solar+flare*.

Dua, D., & Graff, C. (2019a). UCI Machine Learning Repository. *https://archive.ics.uci.edu/ml/datasets/banknote+authentication*.

Duarte, K., Rawat, Y., & Shah, M. (2021). Plm: Partial label masking for imbalanced multi-label classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2739–2748).

Džeroski, S., Demšar, D., & Grbović, J. (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence, 13*(1), 7–17.

Elisseeff, A., & Weston, J. (2001). A kernel method for multi-labelled classification. *Advances in Neural Information Processing Systems, 14*.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics, 7*(2), 179–188.

Godbole, S., & Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. *Pacific-Asia Conference on Knowledge Discovery and Data Mining.*, (pp. 22-30). Berlin.

Gorman, R. P., & Sejnowski, & J., T.. (1988). Analysis of hidden units in a layered network trained to classify sonar Targets. *Neural Networks, 1*(1), 75–89.

Han, M., & Zhang, H. (2022). Multiple kernel learning for label relation and class imbalance in multi-label learning. *Information Sciences, 613*, 344–356.

Jia, B. B., & Zhang, M. L. (2020a). Multi-dimensional classification via kNN feature augmentation. *Pattern Recognition, 106*, Article 107423.

Jia, B. B., & Zhang, M. L. (2020b). Multi-dimensional classification via stacked dependency exploitation. *Science China Information Sciences, 63*(12), 1–14.

Jia, B. B., & Zhang, M. L. (2022). Multi-dimensional classification via selective feature augmentation. *Machine Intelligence Research, 19*(1), 38–51.

Junior, J. C., Faria, E. R., Silva, J. A., & Cerri, R. (2017). Label powerset for multi-label data streams classification with concept drift. *Proceedings of the 5th Symposium on Knowledge Discovery, Mining Learn*, (pp. 97-104). Uberlandia.

Karalič, A., & Bratko, I. (1997). *First order regression. Machine learning, 26*(2), 147–176.

Kim, Y., Lee, Y., & Jeon, M. (2021). Imbalanced image classification with complement cross entropy. *Pattern Recognition Letters, 151*, 33–40.

Kohonen, T. (1990). The Self-Organizing Map. *Proceedings of the IEEE, 78*, 1464–1480. IEEE 78.

Luque, A., Carrasco, A., Martín, A., & de Las Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition, 91*, 216–231.

Ma, Z., & Chen, S. (2018). Multi-dimensional classification via a metric approach. *Neurocomputing, 275*, 1121–1131.

Mishra, N. K., & Singh, P. K. (2021). Feature construction and smote-based imbalance handling for multi-label learning. *Information Sciences, 563*, 342–357.

Montañes, E., Senge, R., Barranquero, J., Quevedo, J. R., del Coz, J. J., & Hüllermeier, E. (2014). Dependent binary relevance models for multi-label classification. *Pattern Recognition, 47*(3), 1494–1508.

Ortigosa-Hernández, J., Inza, I., & Lozano, J. A. (2017). Measuring the class-imbalance extent of multi-class problems. *Pattern Recognition Letters, 98*, 32–38.

Rai, N., Kumar, D., Kaushik, N., Raj, C., & Ali, A. (2022). Fake News Classification using transformer based enhanced LSTM and BERT. *International Journal of Cognitive Computing in Engineering, 3*, 98–105.

Read, J., Bielza, C., & Larrañaga, P. (2013). Multi-dimensional classification with super-classes. *IEEE Transactions on Knowledge and Data Engineering, 26*(7), 1720–1733.

Read, J., Martino, L., & Luengo, D. (2014). Efficient monte carlo methods for multi-dimensional learning with classifier chains. *Pattern Recognition, 47*(3), 1535–1546.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2009). Classifier chains for multi-label classification. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.*, (pp. 254-269). Berlin.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning, 85*(3), 333–359.

Saleh, A. A., & Weigang, L. (2015). A New Variables Selection And Dimensionality Reduction Technique Coupled With Simca Method For The Classification Of Text Documents. *Proceeding of MakeLearn & TIM Joint International Conference 2015*, (pp. 583-591). Bari, Italy.

Saleh, A. A., Hegazy, M., Abbas, S., & Elkosasy, A. (2022). Development of distribution maps of spectrally similar degradation products by Raman chemical imaging microscope coupled with a new variable selection technique and SIMCA classifier. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy, 268*, 1–11.

Senthilkumar, D., & Akshayaa, C. (2020). Efficient Deep Learning Approach for Multi-label Semantic Scene Classification. *Proceedings of the International Conference on Image Processing and Capsule Networks.* Bangkok: Springer.

Sigillito, V. G., Wing, S. P., Hutton, L. V., & Baker, K. B. (1989). Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest, 10*, 262–266.

Smith, J., Everhart, J., Dickson, W., Knowler, W., & Johannes, R. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *Proceedings of the 10th Symposium on Computer Applications and Medical Care* (pp. 261–265). Piscataway: IEEE Computer Society Press. Retrieved from https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.

Song, D., Vold, A., Madan, K., & Schilder, F. (2022). Multi-label legal document classification: A deep learning-based approach with label-attention and domain-specific pre-training. *Information Systems, 106*.

Tarekegn, A. N., Giacobini, M., & Michalak, K. (2021). A review of methods for imbalanced multi-label classification. *Pattern Recognition, 118*, 1–12.

Trohidis, K., Tsoumakas, G., Kalliris, G., & Vlahavas, I. P. (2008). Multi-label classification of music into emotions. *ISMIR, 8*, 325–330.

Tsanas, A., & Xifara, A. (2012). Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings, 49*, 560–567.

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining, 3*(3), 1–13.

Wang, H., Chen, C., Liu, W., Chen, K., Hu, T., & Chen, G. (2020). Incorporating label embedding and feature augmentation for multi-dimensional classification. *Proceedings of the 34th AAAI Conference on Artificial Intelligence, 34*. New York.

Wang, J., Tian, F., Yu, H., Liu, C. H., Zhan, K., & Wang, X. (2017). Diverse non-negative matrix factorization for multiview data representation. *IEEE Transactions on Cybernetics, 48*(9), 2620–2632.

Weigang, L. (1998). A study of parallel self-organizing map. *arXiv preprint quant-ph/9808025*.

Weigang, L., & Silva, N. (1999). A Study of Parallel Self-Organizing Map. *Proceedings of the International Joint Conference on Neural Networks. 2*, pp. 1113-1116. IJCNN.

Weng, W., Chen, C. L., Wu, S. X., Li, Y. W., & Wen, J. (2019). An efficient stacking model of multi-label classification based on pareto optimum. *IEEE Access, 7*, 127427–127437.

Xie, H., Lin, W., Lin, S., Wang, J., & Yu, L. C. (2021). A multi-dimensional relation model for dimensional sentiment analysis. *Information Sciences, 579*, 832–844.

Xu, D., Shi, Y., Tsang, I. W., Ong, Y. S., Gong, C., & Shen, X. (2020). Survey on multi-output learning. *IEEE Transactions on Neural Networks and Learning Systems, 31*(7), 2409–2429.

Xu, X. S., Jiang, Y., Xue, X., & Zhou, Z. H. (2012). Semi-supervised multi-instance multi-label learning for video annotation task. *Proceedings of the 20th ACM International conference on Multimedia*, (pp. 737-740). Nara.

Yang, Q., Li, Y., Li, B., & Gong, Y. (2022). A novel multi-class classification model for schizophrenia, bipolar disorder and healthy controls using comprehensive transcriptomic data. *Computers in Biology and Medicine, 148*, 1–11.

Zhang, H., Li, J. L., Liu, X. M., & Dong, C. (2021). Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection. *Future Generation Computer Systems, 122*, 130–143.

Zhang, M. L., & Zhou, Z. H. (2007). ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition, 40*(7), 2038–2048.

Zhang, M. L., Li, Y. K., Liu, X. Y., & Geng, X. (2018). Binary relevance for multi-label learning: An overview. *Frontiers of Computer Science, 12*(2), 191–202.