Information Fusion 79 (2022) 200–228

Contents lists available at ScienceDirect

Information Fusion

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

Supervised contrastive learning over prototype-label embeddings for network intrusion detection

Manuel Lopez-Martin [a,*], Antonio Sanchez-Esguevillas [a], Juan Ignacio Arribas [a,b], Belen Carro [a]

[a] Dpto. TSyCeIT, ETSIT, University of Valladolid, Paseo de Belén 15, Valladolid 47011, Spain
[b] Castilla-Leon Neuroscience Institute, University of Salamanca, Salamanca 37007, Spain

ARTICLE INFO

Keywords:
Label embedding
contrastive learning
Max margin loss
Deep learning
Embeddings fusion
Network intrusion detection

ABSTRACT

Contrastive learning makes it possible to establish similarities between samples by comparing their distances in an intermediate representation space (embedding space) and using loss functions designed to attract/repel similar/dissimilar samples. The distance comparison is based exclusively on the sample features. We propose a novel contrastive learning scheme by including the labels in the same embedding space as the features and performing the distance comparison between features and labels in this shared embedding space. Following this idea, the sample features should be close to its ground-truth (positive) label and away from the other labels (negative labels). This scheme allows to implement a supervised classification based on contrastive learning. Each embedded label will assume the role of a class prototype in embedding space, with sample features that share the label gathering around it. The aim is to separate the label prototypes while minimizing the distance between each prototype and its same-class samples. A novel set of loss functions is proposed with this objective. Loss minimization will drive the allocation of sample features and labels in embedding space. Loss functions and their associated training and prediction architectures are analyzed in detail, along with different strategies for label separation. The proposed scheme drastically reduces the number of pair-wise comparisons, thus improving model performance. In order to further reduce the number of pair-wise comparisons, this initial scheme is extended by replacing the set of negative labels by its best single representative: either the negative label nearest to the sample features or the centroid of the cluster of negative labels. This idea creates a new subset of models which are analyzed in detail.

The outputs of the proposed models are the distances (in embedding space) between each sample and the label prototypes. These distances can be used to perform classification (minimum distance label), features dimensionality reduction (using the distances and the embeddings instead of the original features) and data visualization (with 2 or 3D embeddings).

Although the proposed models are generic, their application and performance evaluation is done here for network intrusion detection, characterized by noisy and unbalanced labels and a challenging classification of the various types of attacks. Empirical results of the model applied to intrusion detection are presented in detail for two well-known intrusion detection datasets, and a thorough set of classification and clustering performance evaluation metrics are included.

1. Introduction

Contrastive learning is attracting great research attention for its interesting properties to implement classifiers [1]. In a contrastive learning framework, each sample is translated into a representational space (embedding) where it is compared with other similar and dissimilar samples with the aim of pulling similar samples together while pushing apart the dissimilar ones.

There is a plethora of strategies to fulfill this aim by considering different alternatives: (a) **Define similar and dissimilar**: A similar element can be a manually slightly modified replica of the original element (self-supervised) [2] or an element in a closeness position (in time, space, order…) to the original element (unsupervised) [3]. In this latter case, similar elements can be obtained by sampling the data source (*e.g.*, neighboring words in a sentence) in the "proximity" of the original element. Furthermore, similarity between elements can also be defined by

* Corresponding author.
E-mail addresses: manuel.lopezm@uva.es (M. Lopez-Martin), antoniojavier.sanchez@uva.es (A. Sanchez-Esguevillas), jarribas@tel.uva.es (J.I. Arribas), belcar@tel.uva.es (B. Carro).

https://doi.org/10.1016/j.inffus.2021.09.014
Received 24 March 2021; Received in revised form 27 July 2021; Accepted 15 September 2021
Available online 20 September 2021

belonging to the same class/label (supervised) [4], which requires elements to be explicitly labeled. (b) **Type of elements represented in embedding space**: The elements represented in embedding space are normally the sample features. Nevertheless, in this paper we propose to use additional elements that can also be mapped to the same embedding space and jointly compared with the sample features. Specifically, we propose to use the labels as additional elements to be represented in embedding space. (c) **Define the separation process**: Similar and dissimilar elements with respect to a certain reference (anchor) element are usually referred to as positive and negative, respectively. The separation process can be

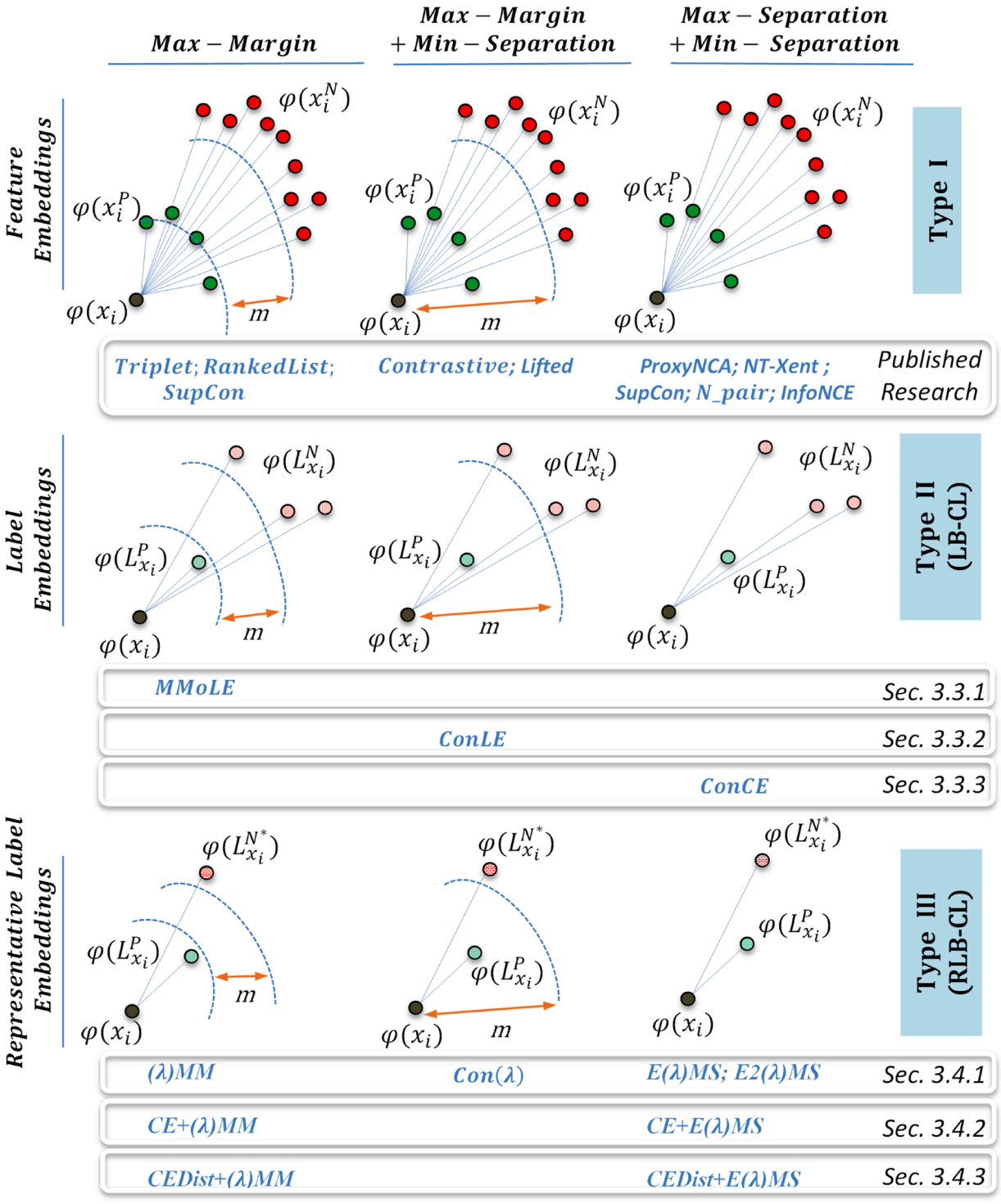

Fig. 1. Contrastive learning reference view considering the embedding elements (**vertical-axis**) and separation process strategy (**horizontal-axis**). Distance computation can be done: only between samples using their feature embeddings (**Upper row**), between each sample (using its feature embedding) and all its positive and negative labels (**Middle row**), and between each sample and its positive label and a single representative of its negative labels (**Lower row**). The separation process can be implemented: with a maximum margin between positive and negative points (features or labels) (**Left column**), with a maximum margin for negative points and a minimum separation to the anchor for positive ones (**Middle column**), and with a maximum/minimum separation between negative/positive points to the anchor (**Right column**). Type II and III solutions correspond to our proposed models. Below each type of solution, its most representative models are listed with a reference to the Section that shows its architecture.

attained in different ways: The positive and negative elements can be pulled/pushed to/from the anchor trying to reduce/increase the distance as much as possible [5,6]. On the other hand, the objective can be to increase the distance between positive and negative just beyond a margin [7]. Additionally, a combined objective can be to minimize the distance of the positive to the anchor and maximize the distance to the negative beyond a margin [8]. (d) **Number and nature of positive and negative samples used in each comparison**: We can choose a single positive and negative element per comparison [7,8], a group of positives and negatives [5,9] or a representative(s) for the positive and negative groups serving as proxy for the group [10]. (e) **Distance function**: Euclidean, Cosine… (f) **Loss function**: contrastive [8], triplet [7], …

Considering the contrastive learning alternatives mentioned above, there is a wide range of solutions found in the literature. These solutions are mainly based on self-supervised and unsupervised settings, and only recently on supervised schemes. They have achieved excellent results [1–4], but they also have some significant drawbacks: (a) long training periods, (b) the need to create complex sampling strategies to extract negative samples, and (c) difficulties in creating representative prototypes for each class/label in a supervised scheme.

In this work, we propose a solution to these drawbacks by creating a supervised contrastive learning scheme where the sample features and labels are mapped to the same embedding space, and the contrastive mechanism (pulling/pushing of similar/dissimilar elements) is applied between features and labels, instead of being applied exclusively between features.

The sample embeddings used in contrastive learning are usually created from the features of the sample. The approach proposed here, which combines features and label embeddings into the same representational vector space, where they can be jointly manipulated, has been recently applied in image [11] and text processing [12] as an intermediate feature processing step, but has not been used, to our knowledge, in any systematic approach within a contrastive learning framework. In this work, we provide a systematic reference framework for current contrastive learning solutions using the **classic features-only embeddings approach (Fig. 1, Type I)**, and our proposed solutions using **jointly features and label embeddings (Fig. 1, Type II and III)**.

Fig. 1 graphically shows where our proposed solutions fit in a generic contrastive learning reference view. The charts in Fig. 1 are based on the following definitions and notation. We assume to have a dataset with N samples: $\{x_i\}_{i=1}^{N}$. We also assume that each sample (x_i) has a set of similar (positive) samples (x_i^P) and a set of dissimilar (negative) samples (x_i^N). Similar and dissimilar samples can be established by a "proximity" function without requiring an explicit labeling (unsupervised and self-supervised) or by belonging to the same explicit label/class (supervised). For the supervised case, we have a set of C labels (classes): $\{L_j\}_{j=1}^{C}$, where each sample x_i is associated to a ground-truth label/class which we call its positive label, implying that the rest $C-1$ labels will be its negative labels. The positive label for x_i is identified as $L_{x_i}^P$ and any of the negative labels for x_i as $L_{x_i}^N$. We can also select a representative label from the group of negative labels, this representative can be one of the existing negative labels (the one that best represents the set of negative labels for our task) or it can be a new one built from them. We will denote as $L_{x_i}^{N^*}$ the unique label acting as a representative of the group of negative labels.

It is important to note that in Fig. 1 the elements represented are in embedding space. The embedding transformation is represented as: $\varphi(a)$, where a is a generic input vector and its embedding is a low dimensional representation of vector a. The function $\varphi(a)$ is a mapping of vector a into a space of different dimensionality (usually smaller) where the objective is to maintain the representational capacity of the original vector into the new mapped vector.

Fig. 1 provides a reference view for contrastive learning solutions considering two axis: the elements represented in embedding space (vertical-axis), and the separation process adopted (horizontal-axis). In this reference view, our research scope corresponds to Type II and III solutions:

- **The embedding elements used** (Fig. 1, vertical-axis) can be: (a) only the sample features (x_i) (**Type I**), (b) the sample features (x_i) plus the positive ($L_{x_i}^P$) and all negative labels ($L_{x_i}^N$) for that sample (**Type II**), and (c) the sample features (x_i) plus a reduced number of its representative labels that provide all the required information for our task (**Type III**). In this latter case, our proposal is to use for each sample, its positive label ($L_{x_i}^P$) and a single representative element of its negative labels ($L_{x_i}^{N^*}$), as the two unique representatives. There are several ways to choose a single representative for the negative labels; we propose two useful alternatives: choose the element which is nearest to the sample features (in embedding space), or choose the centroid of the cluster formed by the negative labels (Fig. 3).

  For Type II and III solutions, the positive and negative labels depend on each sample (x_i) and the aim is to pull closer the sample features (x_i) to its positive label ($L_{x_i}^P$) while pushing apart all the negative labels ($L_{x_i}^N$), or their representatives ($L_{x_i}^{N^*}$). Type I solutions have a similar aim, but in this case the objective is to attract similar features and repel dissimilar ones. If similarity is based on class membership, all types of solutions (Type I, II or III) will create a cluster of samples for each label, however, only Type II and III solutions can use the labels as prototypes acting as cluster centers attracting all their same-class samples and repelling others. Having these prototypes is very important as they can be used as unique representative elements of each class.
- **The separation process** (Fig. 1, horizontal-axis) may involve: (a) to push apart the positive and negative points (features or labels) beyond a certain margin m **(Max-margin)**, (b) to push apart the negative points beyond a certain margin m and to pull closer the positive points as much as possible to the reference sample (**Max-Margin+Min-Separation**), and (c) to push apart the negative points and to pull closer the positive points (both) as much as possible to the reference sample (**Max-Separation+Min-Separation**). The separation process is implemented by specific loss functions which are based in combinations and variants of quadratic, exponential, logloss and maximum-margin (hinge) losses.

Fig. 1 presents the names of the most representative models for each type of solution (below each chart). These models are implemented through a series of proposed architecture (Sections 3.3 and 3.4). In Fig. 1, each of the proposed architectures is assigned a horizontal rectangle where the different models implemented by that architecture are located, along with a reference to the corresponding Section that explains it. There are architectures that provide a single model (for example, the architecture in Section 3.3.2 implements only the Contrastive over Label Embeddings -ConLE model), while other architectures can implement different types of models (with different properties) by changing the loss function used within that architecture; for example, the architecture proposed in Section 3.4.1 implements all types of models, with different properties, and corresponding to different separation approaches (different position on the horizontal-axis of Fig. 1). The two main categories of models in relation to this research are:

- The Type I solution models correspond to currently published research works: Triplet [7], RankedList [13], SupCon [4], Contrastive [8,14], Lifted [9], N-pair [5], NT-Xent [15], ProxyNCA [10,16], InfoNCE [6]. SupCon is the only supervised contrastive learning model identified in the literature, and it is located in two different columns in Fig. 1, because it can adopt different separation strategies based on the number of negative labels considered.
- The proposed models for Type II and III solutions are novel solutions proposed in this paper (Sections 3.3 and 3.4): Max Margin over Label Embeddings (MMoLE), Contrastive over Label Embeddings (ConLE), Contrastive with Cross Entropy (ConCE), Max-Margin ((λ)MM), Cross

Entropy with Max-Margin (CE+(λ)MM), Cross Entropy plus Distances with Max-Margin (CEDist+(λ)MM), Contrastive (Con(λ)), Exponential loss for Max-Separation (E(λ)MS), Squared Exponential loss for Max-Separation (E2(λ)MS), Cross Entropy with Exponential loss for Max-Separation (CE+E(λ)MS) and Cross Entropy plus Distances with Exponential loss for Max-Separation (CEDist+E(λ)MS). Where (λ) stands for a dummy letter representing a particular loss function, as defined in Sections 3.3 and 3.4.

We call the models included in Type II and III solutions as **Label Based Contrastive Learning (LB-CL)** and **Representative Label Based Contrastive Learning (RLB-CL)**, respectively. **They correspond to our proposed models.** These models can be used as: (a) Classifiers, predicting the correct label for a new sample. (b) The embeddings of the sample features and the distances between features and labels embeddings can be used to replace the original features as a dimensionality reduction technique. (c) Finally, the samples form clusters around the label prototypes, which can help to assess the location of new samples in a low dimensional graphical representation of sample and label embeddings (*e.g.*, security personnel identifying attacks in new samples). The graph can be obtained directly using low dimensional embeddings or by transforming the embeddings for visualization (*e.g.*, PCA, t-SNE).

As mentioned earlier, Type I solutions are usually based in self-supervised [2] or unsupervised learning [3]. The Type I solutions based on supervised learning [4] establish two samples as similar when they belong to the same class and dissimilar otherwise. In this case, the classes are used only to select the group of positive and negative samples corresponding to each reference sample (x_i). Type II and III solutions use the labels/classes in a different way. They use the labels as class prototypes in embedding space. The samples are clustered around their same-class label prototype. The aim is to create maximum separated label prototypes, as well as minimizing the distance between each prototype and their same-class samples. In this work, we propose new losses specifically designed for the comparison of distances between the samples and their labels or their label representatives (Sections 3.3 and 3.4).

Type I solutions demand a large number of comparisons (contrastive learning) between a vast combinatorial space of pair-wise positive (similar) and negative (dissimilar) samples. Type II solutions tremendously reduce the necessary pair-wise comparisons to the number of labels (classes) only, which is necessarily much less than the number of samples for a sensible learning process. Type III solutions reduce even more the number of comparisons to just two: one between the sample features and the positive label and other between the sample features and the single representative of the cluster formed by all the negative labels. In addition, in Type III solutions we offer a novel way of inputting labels to the classifier at both training and prediction (test) phases, by presenting all labels in parallel (Figs. 7–9). In this way we further reduce the required training and prediction times (Section 4). Both, Type II and III solutions are novel solutions proposed in this work.

Network intrusion detection (NID) is a complex field that faces the problem of detecting cybersecurity issues (malicious activity or policy attacks) on data networks by analyzing the information contained in the exchanged data packets. The information in the data packets is transformed into a vector of continuous and categorical values (*e.g.*, size, addresses, flags...) that represent the network connection. This vector can be compared, searching for similar patterns, with pre-registered vectors associated to normal traffic or attacks (signature-based intrusion detection), or the vector can be used as the input to statistical methods or machine learning classification methods to detect attacks (anomaly-based intrusion detection). In this work we apply the anomaly-based approach.

NID is an active and challenging area of machine learning research [17–19]. A wide range of machine learning methods have been applied to NID, the most common being: linear models, decision trees, gradient boosting methods, support vector classifiers, multilayer perceptron, convolutional and recurrent neural networks, kernel methods and generative models [20–22]

The proposed models are generic and applicable to any supervised classification problem, and translating these models to NID is straightforward. Network intrusion samples (NIS) are represented by vectors of network features. NIS vectors length is usually large since each feature brings information about a particular property of the network flow carrying the attack. Intrusion labels are represented by one-hot-vectors associated to each label. Label vectors length equal the number of different labels. Labels and NIS vectors are of very different sizes. The objective is to map the NIS and labels into a common representational vector space (embedding space) where the fundamental original information is preserved and where semantically meaningful comparisons can be established.

NID datasets, which contain the activity of network flows and their associated status: normal or intrusion type, are characterized by being good representatives of noisy and unbalanced datasets. We have chosen two well-known NID datasets (NSL-KDD and UNSW-NB15) to carry out the experiments for the novel proposed models (Type II and III solutions). These datasets provide diverse and challenging scenarios for the proposed models. A complete analysis of results is provided on the application of the proposed solutions (Type I and II) to the two data sets (Section 4). The results include: (a) classification performance metrics to detect the attacks, (b) clustering performance metrics on the quality of the clusters of samples around each label prototype and, (c) improvement in classification metrics from several well-known machine learning (ML) algorithms, when using the sample embeddings and their distances to label embeddings as new features, replacing the original sample features.

The **contributions** offered by the proposed models are: (a) Introduce a fusion of features and labels representation within a common vector space. (b) Present a wide range of novel architectures and loss functions that leverage this fused representation space. (c) Provide new alternatives to contrastive learning by avoiding negative sampling or any other sophisticated selection strategy for negative/positive labels. (d) Significatively reduce the number of pair-wise comparisons required. (e) Create a novel approach by replacing negative labels with their best representatives, further reducing the number of comparisons and their associated training and prediction times. (f) Show that the proposed models create excellent classifiers for noisy and unbalanced datasets. (g) Show that features embeddings, created as a by-product of the classification process, can be used instead of the original features. We show that a wide range of classifiers improve their performance by using these low-dimensional alternative features.

The rest of the paper follows this schema: Section 2 summarizes related works. Section 3 describes the proposed model and its relationship with related models. Section 4 describes the datasets, and the experimental results and Section 5 presents the conclusions.

2. Related works

Referring to the three main topics of this research: contrastive learning, label embeddings and NID. As far as we know, there is no work presenting label embeddings within a contrastive framework; there are works using label embeddings as a dimensionality reduction technique, and there are works presenting Type I contrastive learning solutions (Fig. 1) for NID. We have not identified any work proposing label embeddings for NID. The main related works from the literature are as follows:

- **Contrastive learning with feature embeddings for classification:** Contrastive learning is an active research topic with many recent works following different approaches with an emphasis in: (a) Type of separation process (*e.g.*, Max-Margin...)(Fig. 1). (b) Acquisition of similar/dissimilar elements (supervised, self-supervised or unsupervised).(c) Number of positive/negatives samples per comparison (single, group or representative). All these works correspond to Type I solutions (Fig. 1):

  Max-Margin

  Triplet loss was presented in [7] with a max-margin separation for the

distances difference between an anchor point and a selection of its negative and positive samples. One positive and negative sample is used per comparison, and samples are acquired in a self-supervised or unsupervised manner. A group of positive and negative samples per comparison is used in [13] with a max-margin loss called ranked list loss based on weighting positive and negative examples with respect to their distance to the anchor point using class information (supervised).

Max-Margin+Min-Separation

Authors in [23] present a k-nearest neighbor framework to minimize Mahalanobis distance to positive samples and maximize distance with a margin to negative samples. They use a subset of positive samples to compute the loss which is a contrastive loss [14]. Contrastive loss is presented in [8,14] with a quadratic loss for the distance to positive samples and a quadratic max-margin loss to the distance of negative samples. It is applied originally to image recognition using label information [8] and face recognition with a Siamese network [14] in a self-supervised manner. The comparison in all cases is done between a single positive and negative sample. Alternatively, in [9] all negative samples are selected with a modified loss that comprises an exponential loss with a margin for the distance to all negative samples plus the direct distance to the positive sample. The experiments are done with a self-supervised setting.

Max-Separation + Min-Separation

With a supervised learning framework, [16] proposes Neighborhood Component Analysis (NCA) where a k-nearest neighbor with stochastic (soft) neighbor assignments optimizes the probability of a sample to be classified into the correct class. It has connections with Linear Discriminant Analysis (LDA) [24] but it does not require all class distributions to be Gaussian with equal covariance [16]. A single positive sample and a group of negative samples is used per comparison. In [5] the comparison is performed between a positive sample and a random selected element for each negative class (multi-class N-pair). As with the previous work the objective is to push apart or pull closer the negative/positive points as much as possible to the reference sample. Authors in [15] present NT-Xent/SimCLR with a exponential loss over a distance function between similar points normalized by the sum of distances for the rest of points in the same training batch (assumed as negative samples). The work in [25] extracts label prototypes by sampling and averaging their associated embedded features. There is no fusion of labels and features in representation space. The label prototypes are used in a few-shot learning framework. A group of positive and negative samples are implicitly used per comparison.

With a (self/un) supervised framework, the loss presented in NCA is used also in ProxyNCA [10] where instead of using a group of negative samples this method creates a reduced number of proxy elements that can replace the original points in the comparison. The method creates the proxy points among the original points as part of the learning process. It extends NCA to a subset of points represented by proxies instead of the original points. A different loss is presented in [6] with InfoNCE which is based on noise contrastive estimation and applied to sequential data. The loss is based in a log-bilinear score between a latent variable created by an autoregressive model applied to past samples embeddings and future positive samples normalized by the sum of the scores for all samples. A single positive sample and a group of negative samples are used per comparison. The structure of the loss is similar to [5,15,16]. Based on SimCLR, [26] presents a sophisticated sampling mechanism for negative points to minimize the number of false negative points (positive points taken as negative). ProtoNCE [27] is a variant of InfoNCE that creates prototypes of the positive and negative samples with an unsupervised framework by constructing latent variables with an expectation-maximization algorithm. The proposed loss includes the original InfoNCE loss plus an additional loss (also based on InfoNCE) formed by sampling a reduced number of negative and positive prototypes.

- **Alternative frameworks for contrastive learning:** Traditional pair-wise distance ranking applied in contrastive learning for classification can be extended in different directions. Authors in [28] propose an adversarial contrastive learning based in SimCLR. The embedded features can be particularly useful for learning with an small number of samples (few-shot learning) as proposed in [29]. Alternative models based on ranking the recovery error performed by generative models (*e.g.* variational autoencoders) conditioned on the labels can also be used for classification [30].
- **Application of label embeddings:** Label embedding has been applied in image classification [11], text classification with attention mechanism [12], text classification based on a bilinear ranking function [31] and text classification in a multi-task setup [32]. It has also been applied for multi-label embedding with a neural factorization machine [33]. None of these works uses a contrastive learning approach. They employ the label embeddings as an intermediate step for downstream tasks.
- **NID and contrastive learning**: In latter survey studies on the application of machine learning and deep learning to network intrusion detection there is no mention to the application of contrastive learning [18,20,22]. However, there is a growing interest in these techniques in current works; for example, [34] proposes anomaly detection by self-supervised contrastive learning, following the framework proposed in SimCLR [15]. An unsupervised anomaly detection with contrastive learning is proposed in [35], where a neural network is trained by contrasting distances, only between normal instances, and a threshold is used to detect outliers when its distance to the center of normal instances is above the threshold. Authors in [36] propose a supervised intrusion detector using a Siamese network with similar and dissimilar samples being differentiated by belonging to the same class. A similar solution is adopted in [37] to detect intrusions for the NSL-KDD and UNSW-NB15 datasets. A dimensionality reduction technique, by applying a Siamese network, is presented in [38]. Sample features are transformed into a 1D feature space. The resulting variable is used to identify attacks with a simple visualization tool.

3. Methods description

We position the proposed models (LB-CL, RLB-CL), which correspond to Type II and III solutions (Fig. 1), in relation with alternative contrastive learning models and describe in detail the different architectures used to implement them.

3.1. Contrastive learning overview

Contrast learning was originally thought [8,14] to perform a feature transformation such that similar samples would be placed close to each other (in the transformed feature space) according to a certain predefined similarity goal. The similarity of the samples is based on a distance function (usually Euclidean or cosine) and this distance is used to establish the proximity of an anchor sample to a set of positive (similar) and negative (different) samples (Fig. 2). Similarity is imposed on the transformed space by training alternately with groups of similar and dissimilar samples. A special loss function is used to penalize dissimilar elements approaching or similar elements moving apart (Fig. 2). This method is especially useful for creating a feature transformation into an embedding space that incorporates domain-specific semantics using the distance between embeddings. These embeddings can be used as features in downstream classification tasks *e.g.*, word2vec [39]. During training, samples can be presented in pairs or in different combinations/group arrangements of similar/dissimilar samples (*e.g.*, triplets [7]).

Most of the loss functions used in the contrastive learning literature are pairwise ranking losses, based on a combination of quadratic, logarithmic, exponential and maximum-margin losses. As a summary, we provide an overview of the most popular contrastive learning architectures with their associated loss function equations (Fig. 1, Upper charts): **Contrastive** [8,14] (Eq. (1)), **Triplet** [7] (Eq. (2)), **Lifted** [9] (Eq. (3)),

N-pair [5] (Eq. (4)), **InfoNCE** [6] (Eq. (5)), **SimCLR**, NT-Xent [15] (Eq. (6)) and **NCA**, proxy-NCA [10,16] (Eq. (7)):

$$\frac{1}{N}\sum_{N}\left\{y^{+}.(D^{+})^{2}+y^{-}.[\max(m-\ D^{-},\ 0)]^{2}\right\} \tag{1}$$

$$\frac{1}{N}\sum_{N} max\ \left[(D^{+})^{2}-\ (D^{-})^{2}+m,\ 0\right] \tag{2}$$

$$\frac{1}{2P}\sum_{P}\left[max\left(\log\ \left[\sum_{P_n} exp(m-D^{-})\right]+D^{+},\ 0\right)\right]^{2} \tag{3}$$

$$\frac{1}{N}\sum_{N} log\ \left[1+\sum_{P_n} exp(S^{-}-\ S^{+})\right] \tag{4}$$

$$\frac{1}{N}\sum_{N}-log\left[\frac{exp(S^{+}/\tau)}{\exp(S^{+}/\tau)+\sum_{P_n} exp(S^{-}/\tau)}\right] \tag{5}$$

$$\frac{1}{N}\sum_{N}-log\left[\frac{exp(S^{+}/\tau)}{\sum_{P_n} exp(S^{-}/\tau)}\right] \tag{6}$$

$$\frac{1}{N}\sum_{N}-log\left[\frac{exp(-D^{+})}{\sum_{P_n} exp(-D^{-})}\right] \tag{7}$$

Note: In previous equations we have used the following simplified notation: D^{+} and $D^{-}$are distances to positive and negative points (*e.g.*, Euclidean distance). S^{+} and $S^{-}$are similarities to positive and negative points (*e.g.*, Cosine similarity). m is a margin. τ is a temperature parameter. N is the number of samples. P is the number of pairs of samples. P_n is the number of pairs of negative samples (for a particular reference sample). y^{+} is an indicator function whose value is 1 if two similar samples are compared during training and 0 otherwise; y^{-} is also an indicator function to mark dissimilar samples ($y^{-}\ =\ 1\ -y^{+}$)

When considering the literature for contrastive learning, much of the complexity and sophistication goes into finding a good representative set of positive (similar) and negative (dissimilar) points to compare to a specific anchor point. This creates implementation problems because the distance computation between samples (sample-wise) demands a complex training process. The training is performed by selecting a representative number of positive and negative samples for each reference sample. To reduce variance and ensure generalization of results the set of positive/negative representative samples is usually large and difficult to choose.

Contrastive learning is generally applied in an unsupervised/self-supervised scheme. In a self-supervised scheme, similar samples are constructed (synthesized) by some controlled modification of the anchor sample (*e.g.*, rotate/translate an image). In an unsupervised scheme, similar samples are selected by some proximity attribute imposed on the data (*e.g.*, time, location, order, position...). In this latter case, the selection is performed by random sampling conditioned on the proximity attribute (*e.g.* similar words are selected by proximity to the anchor word in a document corpus [39]). Contrastive learning has difficulties to be applied in a supervised scheme, since we have to select a representative number of samples per class to compare with a certain anchor sample, and in the inference (prediction) phase we also need to compare the new sample with a significant number of samples corresponding to all classes. The label assigned to the new sample is selected by some form of majority voting using the labels of the closest samples. The main problem with this approach is the initial lack of a natural representative for the classes; having this natural representative of the classes would reduce the number of distance comparisons required at the time of inference to a single comparison with each class representative.

The above mentioned problems: a) large number of pair-wise comparisons, b) complexity in selecting representative sets of positive and negative samples, and c) difficulty to integrate supervised learning, are addressed by our proposed solutions using the labels themselves as the best class representatives. This approach reduces drastically the number of required pair-wise comparisons since the points to compare with now are the labels, which we assume have a significant smaller number than the number of samples. Additionally, label embeddings now get a natural representative position in the cluster of points sharing their label; either by choosing the geometric center of this cluster or simply the point with the smallest distance to the majority of points within the same class.

3.2. Scenarios for contrastive learning

In this section we provide an alternative schematic view (Fig. 3) showing the main elements that define contrastive learning frameworks and how our proposed models relate to them. It is a complementary view to Fig. 1. A visual summary of this alternative view is given in Fig. 3. The upper-chart in Fig. 3 shows the process followed to create common embeddings for features and labels. The middle-chart shows the alternative contrastive learning scenarios corresponding to Type I, II and III solutions (Fig. 1), where Type I solutions have been divided by the data

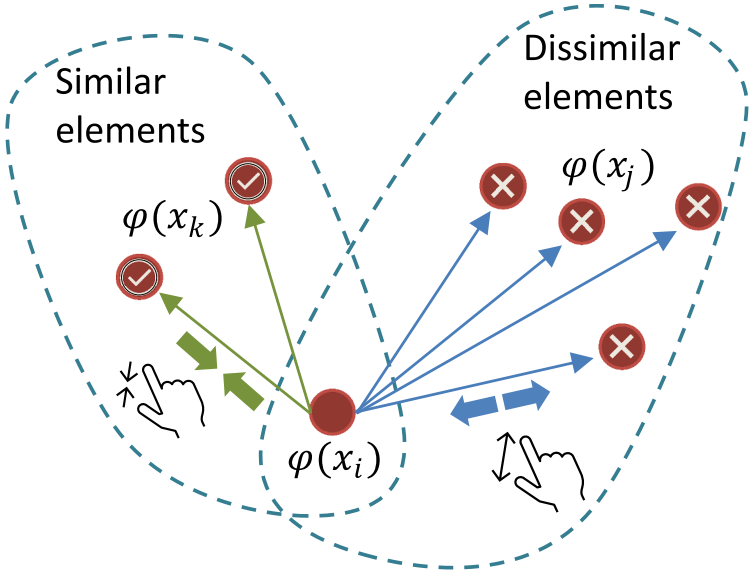

Fig. 2. Overview representation of contrastive learning. An anchor sample (x_i) in embedding space ($\varphi(x_i)$) is moved close to similar samples (by some similarity criteria, *e.g.*, same age faces) ($\varphi(x_k)$) and is separated from dissimilar samples ($\varphi(x_j)$).

acquisition process into self-supervised/unsupervised (I.A) or supervised (I.B). Finally, the lower-charts present two variants of Type III solutions (III.A and III.B), showing two alternative ways to choose the representative element (prototype) assigned to the negative labels ($L_{x_i}^{N^*}$).

We now extend the definitions and notation given in Section 1; this extended notation will be applied in the rest of this work. We assume to

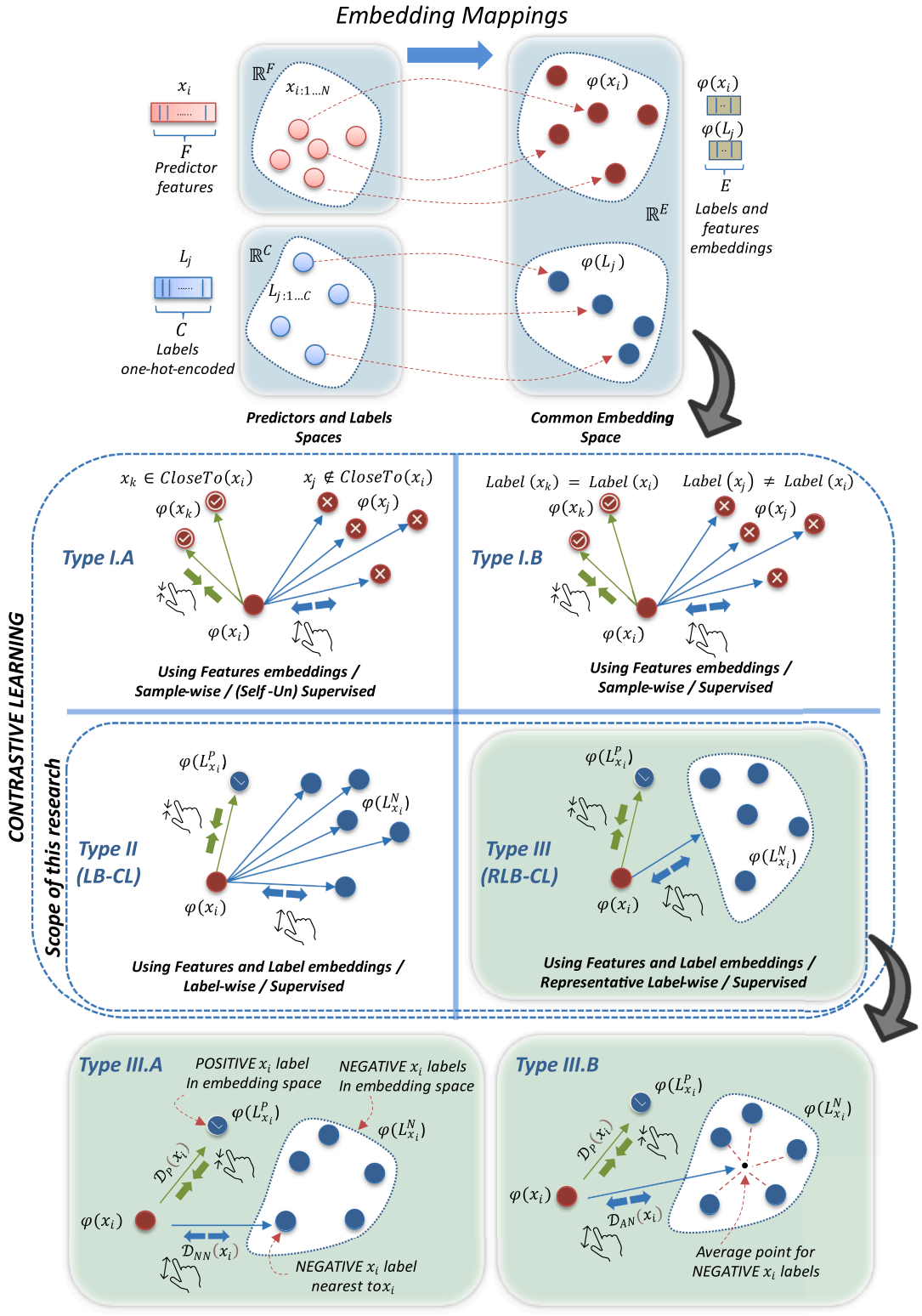

Fig. 3. (**Upper chart**) Schema of the mapping of labels and predictor features into the same embedding representational space. The labels and sample features are vectors of different dimensions (C and F respectively). The embedding mapping translates them into vectors of the same dimensionality (E) where their comparison is feasible. (**Middle charts**) Four scenarios for contrastive learning with different defined distances between features (Type I) and between features and labels (Types II and III). (**Lower Charts**) Detail of Type III solutions, describing the distances $\mathscr{D}_{NN}(x_i)$ (**Lower-left**) and $\mathscr{D}_{AN}(x_i)$ (**Lower-right**) between the sample to either its nearest negative label ($\mathscr{D}_{NN}(x_i)$) or the average point formed by all its negative labels ($\mathscr{D}_{AN}(x_i)$), respectively. Which labels are positive or negative is set for each sample (x_i). The positive label for a sample x_i is its ground-truth label, all others are negative labels for this particular sample. All distances are computed using the vector embeddings.

have a dataset with N samples: $\{x_i\}_{i=1}^{N}$, each sample having F features: $x_i \in \mathbb{R}^F$ (predictor features). And a set of C labels (classes): $\{L_j\}_{j=1}^{C}$(Fig. 3, upper-chart). Each sample x_i can be associated to a ground-truth label/class (y_i) which we call its positive label, implying that the rest $C-1$ labels will be its negative labels. We identify the index of the positive label for x_i as $k_i \in [1,C]$, such that $L_{k_i}$is the ground-truth label for x_i, *i.e.*, $L_{k_i} = y_i$. We will also identify the positive label for x_i as $L_{x_i}^P$ (*i.e.*, $L_{x_i}^P = L_{k_i} = y_i$) and any of the negative labels for x_i as $L_{x_i}^N$ (*i.e.*, $L_{x_i}^N \neq y_i$). Labels are one-hot-encoded represented with a zero array of length C with a single 1 in a specific position assigned to each label; in particular, for the ground-truth label y_i, we indicate the position j in its one-hot-encoded array as y_i^j, that means that $y_i^j = 1$ if $j = k_i$ and $y_i^j = 0$ if $j \neq k_i$. The same notation will be used for the predicted label ($\widehat{y}_i$). In line with Section 1, we will denote as $L_{x_i}^{N^*}$ as the unique label acting as a representative of the group of negative labels (Type III solutions).

The labels and sample features are vectors of different dimensions (C and F respectively). We define a specific embedding mapping for each of these vectors (Fig. 3, upper-chart) with two aims: (a) translate them into vectors of the same dimensionality (E) where their comparison is feasible and, (b) create relationships between the embedded vectors with the objective to classify each sample x_i with its associated positive label $L_{x_i}^P$. The embedding transformation is represented as: $\varphi(a): \mathbb{R}^m \rightarrow \mathbb{R}^n$, where a is a generic input vector $a \in \mathbb{R}^m$ and its embedding is a low dimensional representation of vector a: $\varphi(a) \in \mathbb{R}^n$,where usually $m > n$. The function $\varphi(a)$ is a mapping of vector a into a space of different dimensionality (usually smaller) where the objective is to maintain the representational capacity of the original vector into the new mapped vector.

Fig. 3 (middle-charts) shows the different options available to perform the approach between similar samples and the separation between samples that are not similar or that do not belong to the same class. The approach/separation process is done in embedding space under four different scenarios:

- In **Type I.A solutions** (Fig. 3), we assume that each sample x_i has a set of associated samples x_k which are similar/close to x_i and are identified as $x_k \in CloseTo(x_i)$. In this scenario, the aim is to bring x_i closer to its similar samples (x_k), while separating x_i from samples that do not belong to $CloseTo(x_i)$. The elements belonging to $CloseTo(x_i)$ can be obtained by time, spatial or context proximity to x_i without requiring a direct labeling of the "close to" set. This set can be obtained by simply sampling the data source (*e.g.*, neighboring words, next sound) in the "proximity" of x_i. This approach is usually called **unsupervised** [3,6]. Another approach is the generation of augmented replicas of each sample by introducing controlled modifications to the original sample (**self-supervised**) [2]. In this case, the replicas would be considered similar elements.

  No labels are used in this scenario. Only distances between **feature-embeddings** are employed. The contrastive mechanism is applied to each sample x_i and requires that the sample be compared with a significant number of samples similar to x_i ($\{x_k \in CloseTo(x_i)\}$) and also with a representative number of samples not similar to x_i ($\{x_j \notin CloseTo(x_i)\}$) (**sample-wise**)
- A labeling of the samples is assumed in **Type I.B solutions** (Fig. 3). The labeling is done in a **supervised** manner, and only distances between **feature-embeddings** are employed. We call positive samples those samples that share the same class as the reference sample, and negative samples those samples with a different class from the reference sample [4]. In much the same way as Type I.A solutions, the aim is to bring x_i closer to its positive samples, while separating x_i from its negative samples. The contrastive mechanism is also applied (**sample-wise**) between each sample with other samples sharing the same label, as well as samples with other labels. Using the sample-wise distances, the classification can be done by majority vote, if a distance threshold is previously established for the similarity of the samples pair, or some other aggregated distance computation (*e.g.*, choose the class with a minimum sum of distances between the reference sample and the representative samples of that class). The process has similarities to a K-Nearest Neighbors (KNN) model where the class of a new element is assigned according to the majority class of its K nearest elements. The difference in our case is that the set of elements to perform the distance comparison is not the complete population (as in KNN) but a selection of samples from the different classes, and the selection is made by random sampling of these classes. We can see, that this solution has an implicit problem due to the lack of a single representative that serves as a **prototype** for the different classes [40]. This is the problem solved with Type II and III solutions (Fig. 3) by building a best representative (prototype) in embedding space for each of the classes. When using these class prototypes, the objective will now be to reduce the distance between a reference sample to its same-class (positive) prototype and to increase the distance to its different-class (negative) prototypes. By using prototypes, the training and prediction pair-wise comparison process is significantly reduced and creates clear references to perform comparisons. Both Type II and III solutions correspond to supervised models with a label associated with each training sample.
- In **Type II solutions (LB-CL)**(Fig. 3), the sample **features and the labels are embedded** into the same vector space where their distance can be easily computed. The aim in this scenario is for each sample to be as close as possible to its label embedding while separating itself from all other label embeddings. We apply this scenario to perform classification by comparing an anchor sample with its positive and each of its negative labels (**Label-wise**) and choosing the label with the shortest distance to the sample (in embedding space). This is a **supervised** scenario. The models presented for this scenario are part of this research work, and are novel in applying contrastive learning and systematically extend other previous works based exclusively on features embeddings to a new scenario that simultaneously considers feature and label embeddings.
- **Type III solutions (RLB-CL)**(Fig. 3) are also part of our proposed models. In this case, we also use the **features and label embeddings**, but we replace all the negative labels corresponding to a sample with a single element that represents them. This element is assumed to be the most representative element of the cluster of negative labels (this cluster is sample dependent). In this way, we will not perform a comparison of the sample's features with each of the labels (as in Type II) but just with two labels: the same-class or positive label, and the representative of the out-of-class or negative labels (**Representative Label-wise**). We will consider two main variants for choosing the most representative element of the cluster of negative labels: (a) the nearest negative label to the sample (III.A, Fig. 3, Lower-left) and (b) the centroid of the negative labels cluster (III.B, Fig. 3, Lower-right). This is also a **supervised** scenario. There is no work (in any field, as far as we know) proposing a similar approach of using a single element (representative) of the negative labels (in embedding space).

In Fig. 3, for Type II and III solutions, we can define different distances between labels (L_j) and predictor features (x_i) in embedding space (Fig. 3, lower-charts): (a) $\mathscr{D}_P(x_i)$ (Eq. (8)) is the distance of the x_i embedding to its positive label embedding (L_{k_i}). (b) $\mathscr{D}_N(x_i,\ L_j)$ (Eq. (9)) is the distance of the x_i embedding to one of its negative labels embedding (L_j *with* $j \neq k_i$). (c) $\mathscr{D}_{NN}(x_i)$ (Eq. (10)) is the distance between the x_i embedding and its negative label embedding which is the nearest to x_i. (d) $\mathscr{D}_{AN}(x_i)$ (Eq. (11)) is the average distance of the x_i embedding to all its negative label embeddings. $\mathscr{D}_{NN}$ and $\mathscr{D}_{AN}$ correspond to distances between a sample embedding (x_i) and a representative of its negative labels ($L_{x_i}^{N^*}$).

$$\mathscr{D}_P(x_i) = \mathscr{D}[\varphi(x_i),\ \varphi(L_{k_i})] \tag{8}$$

$$\mathscr{D}_N(x_i,\ L_j) = \mathscr{D}\left[\varphi(x_i),\ \varphi(L_j)\right] \text{where } j \neq k_i \tag{9}$$

$$\mathscr{D}_{NN}(x_i) = \min_{j \neq k_i} \mathscr{D}\left[\varphi(x_i),\ \varphi(L_j)\right] \tag{10}$$

$$\mathscr{D}_{AN}(x_i) = \frac{1}{C-1} \sum_{j \neq k_i} \mathscr{D}\left[\varphi(x_i),\ \varphi(L_j)\right] \tag{11}$$

A schematic representation of $\mathscr{D}_P$, $\mathscr{D}_{NN}$ and $\mathscr{D}_{AN}$ is given in Fig. 3 (lower-charts). The average distance to the negative labels ($\mathscr{D}_{AN}$) is an upper bound (Appendix. A) for the distance to the cluster's centroid of the negative labels, therefore, by maximizing this distance we can assume that the other is also maximized.

3.3. Label based contrastive learning (LB-CL) (Type II)

There are three proposed architectures for Label Based Contrastive Learning (LB-CL) solutions (Type II solutions, Fig. 1): (a) Max Margin over Label Embeddings (**MMoLE**), (b) Contrastive over label embeddings (**ConLE**), and (c) Contrastive with Cross Entropy (**ConCE**). These architectures implement a contrastive learning framework between the sample features and all its labels (positive and negatives).

3.3.1. Max margin over label embeddings

The **Max Margin over Label Embeddings architecture (MMoLE)** (Fig. 4) is based on a max margin loss acting over the distance between the positive and one negative label for a sample x_i. The MMoLE loss (Eq. (12)) is defined as follows, where: N is the number of samples; C is the number of labels; $\{L_j\}_{j=1}^{C}$ is the set of C labels; $k_i \in \{1..C\}$ corresponds to the index of the positive label of x_i and the loss is extended to all negative labels for all samples:

$$MMoLELoss = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C-1} \max\left(\mathscr{D}\left[\varphi(x_i), \varphi(L_{k_i})\right] - \mathscr{D}\left[\varphi(x_i), \varphi\left(L_{j \neq k_i}\right)\right] + 1, 0\right) \tag{12}$$

This model implements a Max-Margin separation strategy (Fig. 1) by increasing the difference, beyond a margin, between two distances: the distance between the anchor-sample (x_i) and the positive label, and the distance between the anchor-sample and each negative label. MMoLEloss is a novel max margin loss between the distance of a sample (x_i) to its true label (L_{k_i}) and each of the distances to the negative labels ($L_{j \neq k_i}$). MMoLEarchitecture analysis can be split between its training and prediction phases (Fig. 4). Three inputs per sample are required during

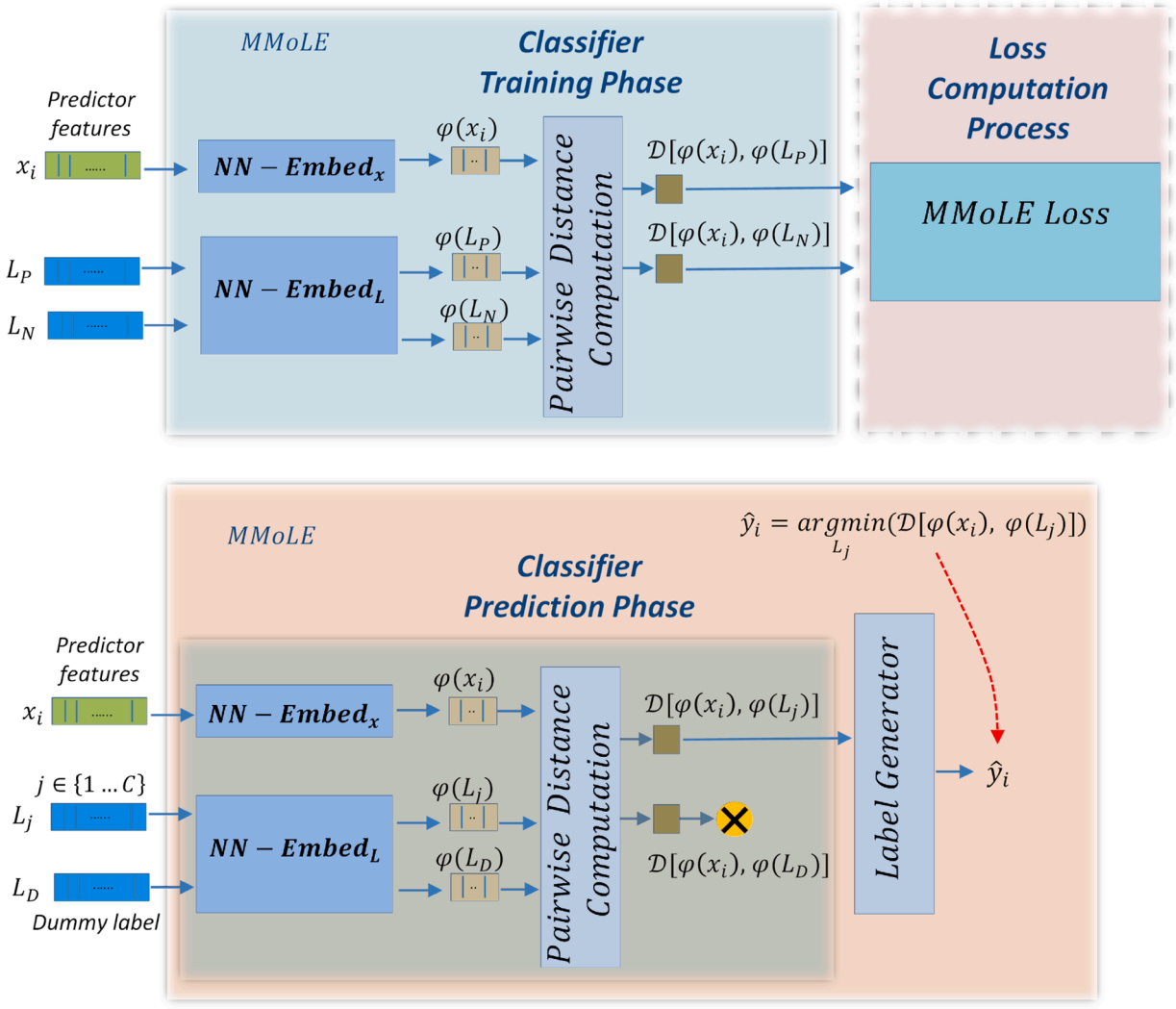

Fig. 4. ***Max Margin over Label Embeddings*** **(*MMoLE*)** ***architecture*** is based on a max margin loss acting over the distance between the positive and one of the negative labels, for each sample x_i. The loss is extended to all negative labels. The training phase (**Upper chart**) requires three inputs per sample: sample features (x_i), positive label (L_P) and one of the negative labels (L_N). Each sample has a training round with every negative label. In the prediction phase (**Lower chart**) we need to try all labels to obtain the one with the smallest distance to the sample, and this operation requires as many forward-passes as labels. Since the training phase requires two labels per forward-pass we treat only one of the labels and set to a dummy value the other.

training: sample features (x_i), positive label (L_P), and one of the negative labels (L_N). L_P and L_N correspond to L_{k_i} and $L_{j\neq k_i}$, respectively in Eq. (12). Each sample has a training round with each one of the negative labels, thus requiring C-1 training samples per dataset sample. This requirement increments the dataset size required for training. The features and the two labels (one hot encoded) are the inputs to two different neural networks (NN): $NN-Embed_x$ and $NN-Embed_L$. The neural network for the labels ($NN-Embed_L$) is shared for both the positive and negative labels (L_P and L_N).These NNs have different input sizes and the same small dimension output size. The outputs of the NNs are the respective features and label embeddings: $\varphi(x_i)$, $\varphi(L_P)$ and $\varphi(L_N)$. Then, a distance function (Euclidean) is applied to the label embeddings to compute two distances: the distance between the features and the positive label ($\mathscr{D}[\varphi(x_i), \varphi(L_P)]$) and the distance between the features and each of the negative labels given as input ($\mathscr{D}[\varphi(x_i), \varphi(L_N)]$). These two distances are used as the input to the MMoLEloss (Eq. (12)), and are also the output of MMoLE.

In the prediction phase, the inputs to the model are the features of the test sample and the label we want to test with. Since the model requires two labels as input, we provide a dummy label as the third input. The distance to this dummy label is discarded and only the distance to the first label is considered. As in previous LB-CL models, we need to try all labels to obtain the one with the smallest distance to each sample, and this operation requires as many forward-passes as labels. MMoLE is the architecture that presents the worst results.

3.3.2. Contrastive over label embeddings

The **Contrastive over label embeddings architecture (ConLE)** (Fig. 5) is based on the contrastive over label embedding loss (*ConLE Loss*) (Eq. 13), defined as follows, where: $\mathscr{D}[a, b]$ is the Euclidean or Cosine distance between vectors a and b; N is the number of samples; y_i is the ground-truth label for the x_i sample; $\{L_j\}_{j=1}^{C}$ is the set of C labels; and, y_{i,L_j} is an indicator function, such that:

$$y_{i,L_j} = \begin{cases} 1 & if\ L_j = y_i \\ 0 & if\ L_j \neq y_i \end{cases}$$

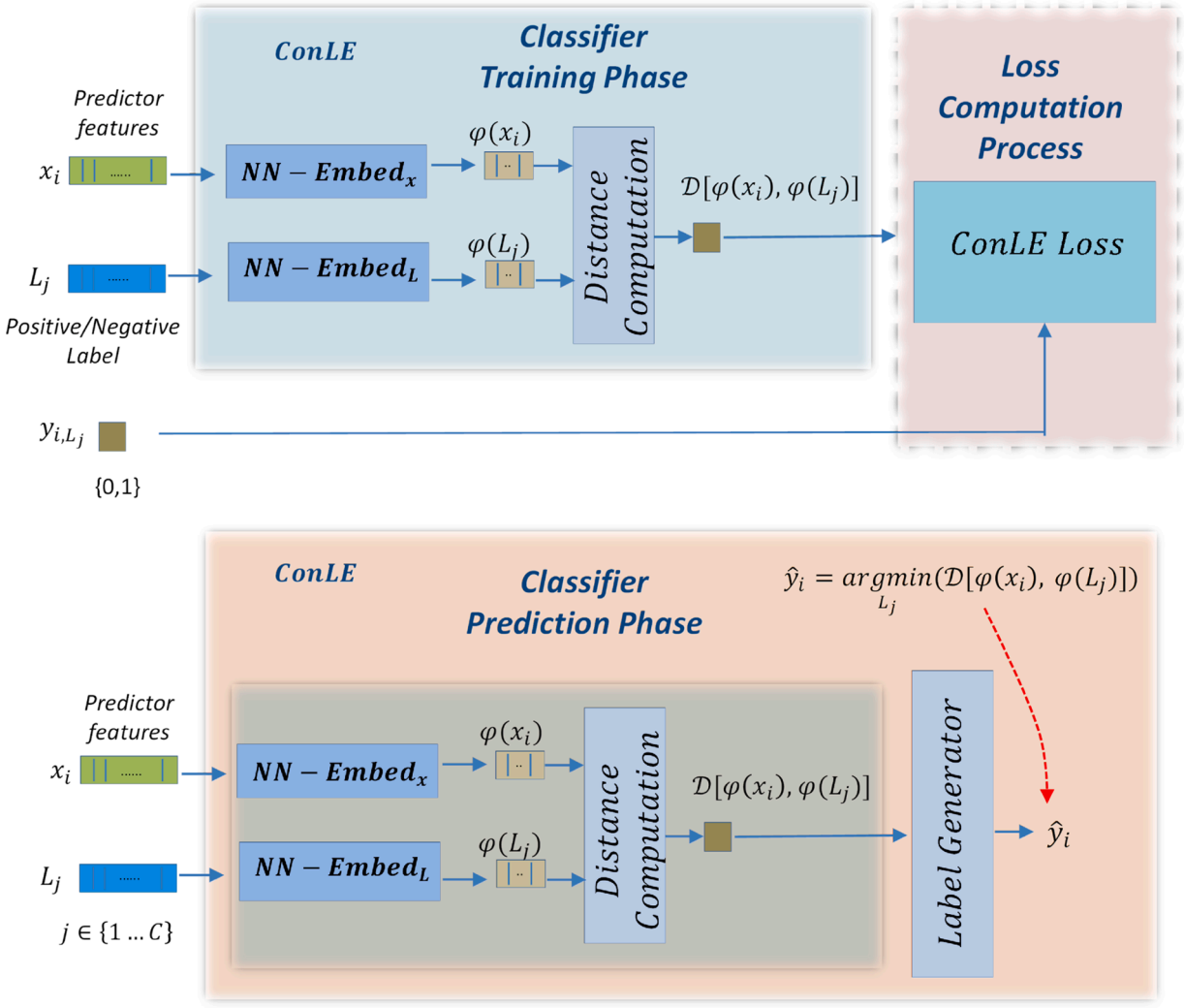

Fig. 5. Contrastive over Label Embeddings (*ConLE*) *architecture:* Model using a contrastive loss function. During training (**Upper chart**) three inputs per sample are required: sample features (x_i), label (L_j) and a binary value (y_{i,L_j}) indicating if the label is positive or negative. Each sample has a training round with every label. In the prediction phase (**Lower chart**) we need to try all labels to obtain the one with the smallest distance to the sample, and this operation requires as many forward-passes as the number of different labels (classes).

$$ConLE\ Loss = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\left[y_{i,L_j}\cdot\mathscr{D}\left[\varphi(x_i),\ \varphi(L_j)\right] + \left(1-y_{i,L_j}\right)\cdot\max\left(1-\mathscr{D}\left[\varphi(x_i),\ \varphi(L_j)\right],0\right)\right] \quad (13)$$

This model implements a Max-Margin+Min-Separation separation strategy (Fig. 1) by increasing the distance, beyond a margin, between the anchor-sample (x_i) and each negative label while reducing the distance, as much as possible, between the anchor-sample (x_i) and the positive label. ConLE loss is similar to contrastive loss (Eq. (13)) with no squared distances. ConLE architecture can be differentiated between its training and prediction phases (Fig. 5). Three inputs per sample are required during training : sample features (x_i), label (L_j), and a binary indicator (y_{i,L_j}). The binary indicator marks if the label is positive (ground truth label) o negative (wrong label). Each sample has a training round with each of the labels, thus requiring C training samples per dataset sample. This requirement increments the dataset size required for training. The features and the label (one hot encoded) are the inputs to two different neural networks (NN): $NN-Embed_X$ and $NN-Embed_L$. These NNs have different input sizes and the same small dimension output size. The outputs of the NNs are the respective features and label embeddings: $\varphi(x_i)$ and $\varphi(L_j)$. Then, a distance function (Euclidean or Cosine) is applied to the label embeddings to compute their distance ($\mathscr{D}[\varphi(x_i), \varphi(L_j)]$) which is used as an input, together with the indicator variable, to the ConLE loss (Eq. (13)). The output of ConLE is the distance between the features and the label (a single label) that are used as inputs. In the prediction phase we need to try all labels to obtain the one with the smallest distance to each sample, and this operation requires as many forward-passes as the number of different labels (classes).

3.3.3. Contrastive with cross entropy

The **Contrastive with Cross Entropy architecture (ConCE)** (Fig. 6) is based on contrastive learning implemented with a binary Cross Entropy loss function (*ConCE Loss*) (Eq. (14)), with the following definition, where: N is the number of samples; C is the number of labels; y_i is the ground-truth label for x_i; $\hat{y}_{i,L_j}$ is the output of the model for sample x_i and label L_j; and, finally y_{i,L_j} is an indicator function, such that:

$$y_{i,L_j} = \begin{cases} 1 & if\ L_j = y_i \\ 0 & if\ L_j \neq y_i \end{cases} \text{ with } \hat{y}_{i,L_j} \in [0,1]$$

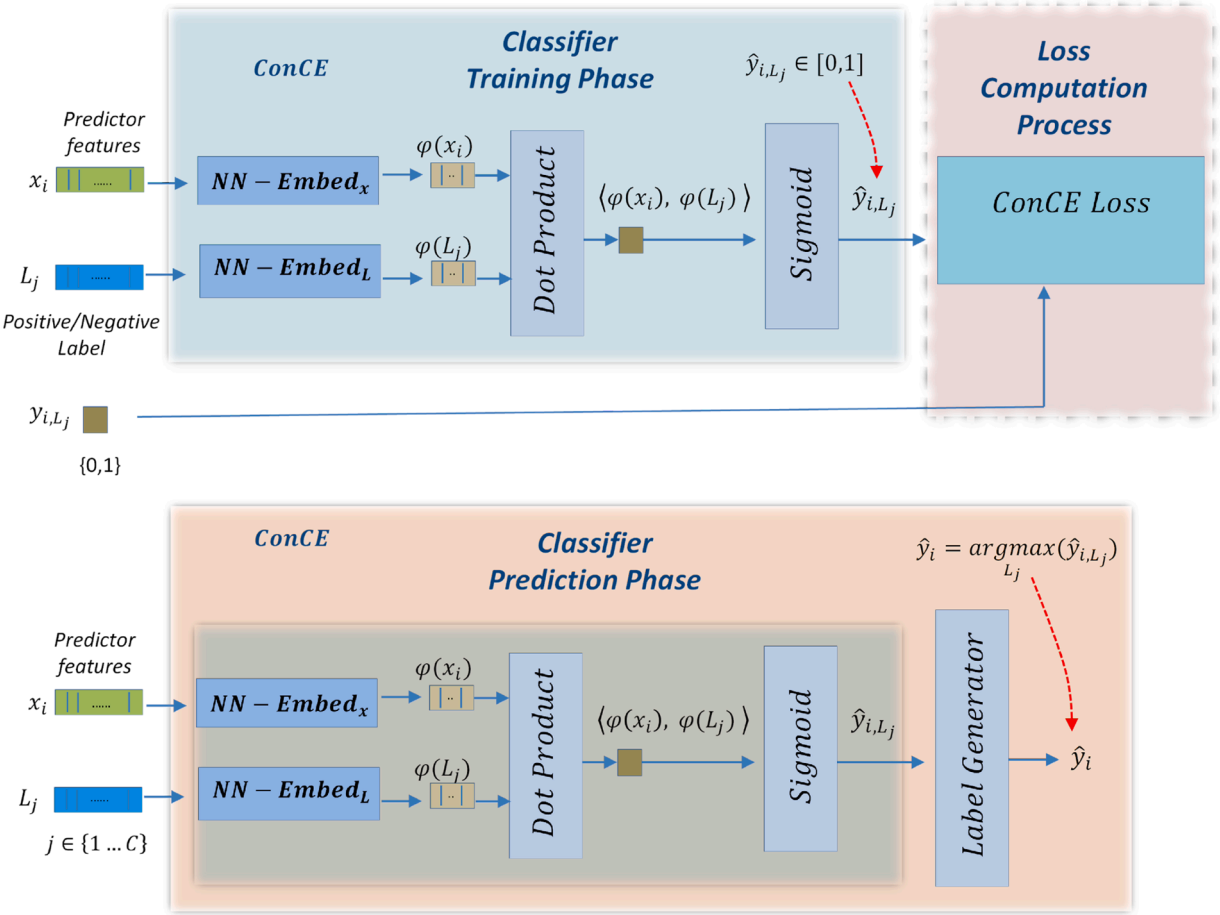

Fig. 6. Contrastive with Cross Entropy (*ConCE*) *architecture:* Model using a contrastive learning framework implemented with a cross entropy loss. The training phase (**Upper chart**) requires three inputs per sample: sample features (x_i), label (L_j) and a binary value (y_{i,L_j}) indicating if the label is positive or negative. Each sample has a training round with every label. In the prediction phase (**Lower chart**) we need to try all labels to obtain the one with the smallest distance to the sample, and this operation requires as many forward-passes as labels.

$$ConCE\ Loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\left[y_{i,L_j}\cdot\log\left(\widehat{y}_{i,L_j}\right) + \left(1-y_{i,L_j}\right)\cdot\log\left(1-\widehat{y}_{i,L_j}\right)\right] \quad (14)$$

This model implements a Max-Separation+Min-Separation separation strategy (Fig. 1) by reducing, as much as possible, the distance between the anchor-sample (x_i) and the positive label, while increasing, as much as possible, the distance between the anchor-sample (x_i) and each negative label. In this model the distances are transformed to a range of values between 0 and 1 by means of a sigmoid function, which allows us to interpret them as a probability and apply the cross-entropy loss. The proposed ConCE loss is roughly similar to negative sampling in word2vec [39] but adapted to contrastive learning over label embeddings. The architecture of ConCE (Fig. 6) is similar to ConLE for both the training and prediction phases, the main difference with ConLE being that the distance computation is now a dot product with an additional sigmoid function to scale the output in the range of values [0,1] ($\widehat{y}_{i,L_j}$). This value will be the output value of ConCE, and is interpreted as the a posteriori probability that the features (x_i) and the label (L_j) (a single label) given as inputs correspond to a true pair, meaning that the label is the right one for the sample. The Cross Entropy loss will try to reduce the distance between embeddings for true pairs and increase the distance for false pairs. During prediction we need to try all labels to obtain the one with the smallest distance to each sample, and this operation requires as many forward-passes as labels.

3.4. Representative Label Based Contrastive Learning (RLB-CL) (Type III)

There are three groups of proposed architectures for Representative Label Based Contrastive Learning (RLB-CL) solutions (Type III solutions, Fig. 1): (1) Representative Label Contrastive architectures: **(λ)MM, Con(λ), E(λ)MS, E2(λ)MS**, (2) Cross Entropy over Labels with Contrastive Regularization architectures: **CE+(λ)MM, CE+E(λ)MS**, and (3) Cross Entropy over Labels and Distances with Contrastive Regularization architectures: **CEDist+(λ)MM, CEDist+E(λ)MS**. These architectures implement a contrastive learning framework between the sample features and its positive label and a single representative for the cluster of negative labels.

3.4.1. Representative label contrastive

The **Representative Label Contrastive architectures ((λ)MM, Con(λ), E(λ)MS, E2(λ)MS)** Fig. 7) is a novel group of architectures based in a combination of max-margin, exponential and contrastive losses. They are based on novel losses that depend exclusively on the distances to the representative labels, that is, to the positive label and a representative point for the cluster of negative labels: $\mathscr{D}_P$, $\mathscr{D}_{NN}$, $\mathscr{D}_{AN}$ (Eqs. (8),(10) and (11)) (Fig. 3). The group of positive and negative labels are established for each sample. These architectures have three types of losses: (a) Max-margin: (λ)MM, (b) Max-Margin+Min-Separation: Con(λ) (c) Max-Separation+Min-Separation: E(λ)MS, E2(λ)MS. Where (λ) stands for a

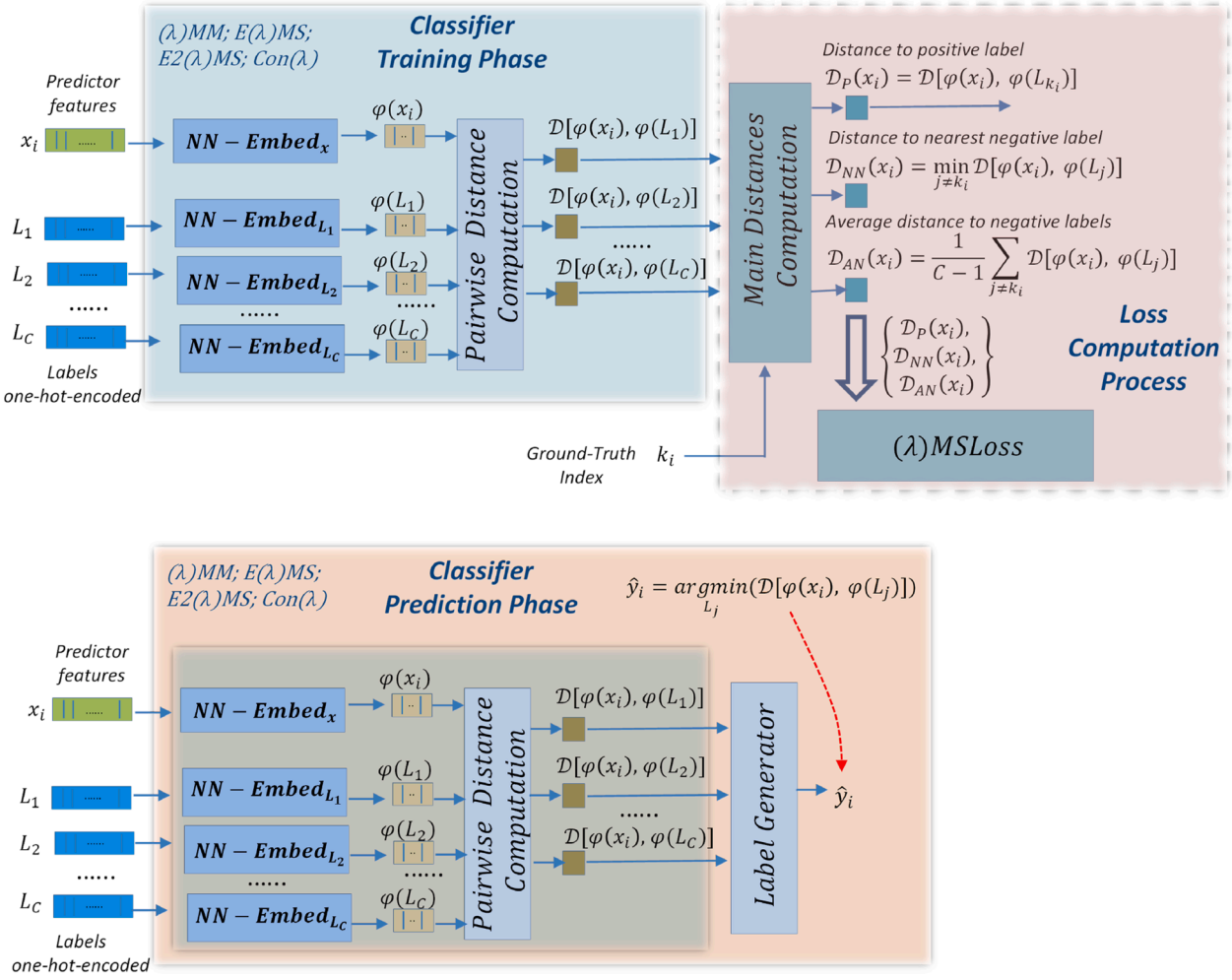

Fig. 7. ***(λ)MM, E(λ)MS, E2(λ)MS and Con(λ) architectures:*** Each of these models use a different related loss function with different distances defined for the negative labels. The training phase (**Upper chart**) requires C+2 inputs per sample: sample features (x_i), all C labels (L_1, L_2,..L_C) and the integer index ($k_i \in \{1...C\}$) of the positive label. In the prediction phase (**Lower chart**) we need a single forward pass to obtain the label with the smallest distance to the sample.

dummy letter representing the loss function (see description below).

All these models share the same fundamental architecture Fig. 7) with different implementations for their loss function (Eqs. (15)–(24)). The architecture can be differentiated between its training and prediction phases Fig. 7). During training, the architecture requires C+2 inputs: one input is the sample features (x_i), other inputs are the one-hot-encoded labels (L_1, L_2,..L_C) and, finally, an integer k_i acting as the index of the positive label in the set {1...C}. By providing all the necessary inputs, it is not necessary to perform several rounds of training for each sample as it happens in the other models (Type I and II). The features and the C labels are the inputs to C different neural networks (NN): $NN-Embed_x$,$NN-Embed_{L_1}$, ..$NN-Embed_{L_C}$. These NNs have different entry sizes and the same small dimension output size. The outputs of the NNs are the respective features and label embeddings: $\varphi(x_i)$, $\varphi(L_1)$,..,$\varphi(L_C)$. Then, a distance function (Euclidean) is applied to the label embeddings to compute the distances between the features and each label ($\mathscr{D}[\varphi(x_i), \varphi(L_1)], .., \mathscr{D}[\varphi(x_i), \varphi(L_C)]$). These distances are the output of the architecture, and are also used by a function ("Main Distance Computation", Fig. 7) that knowing the index of the positive label (k_i) will obtain the distance to the positive label ($\mathscr{D}_P$), the distance to the nearest negative label ($\mathscr{D}_{NN}$) and the average distance to all the negative labels ($\mathscr{D}_{AN}$). These three distances are used as the inputs for all subsequent losses (Eqs. (15)–(24)), and are also the output of the architectures. The loss functions Eqs. (15)–(24) provide an average loss for all N samples.

In the prediction phase (Fig. 7), the inputs to the model are the features of the test sample and all the labels. The C output distances, between the sample features and each label, will be used by a label generator module that will simply compare the distances and choose the smallest as the distance associated with the predicted label. We need a single forward pass to obtain the label with the shortest distance to the test sample. This approach substantially reduces the prediction times, as corroborated by experimental results (c.f. Section 4).

The Representative Label Contrastive architectures share the same base architecture (Fig. 7) and their differences correspond to different loss functions:

- **Max-margin losses:** The loss functions for the **(λ)MM** models Fig. 7) are the following; where (λ) stands for a dummy letter to be replaced by N, A, NA or WNA) (Eqs. (15)–(18)). They are novel max-margin losses for the difference of distance between the positive label ($\mathscr{D}_P$) and one of the selected representatives of the negative labels ($\mathscr{D}_{NN}$, $\mathscr{D}_{AN}$). These losses try to make the distance difference greater than a margin (Max-margin)(Fig. 1). NAMM proposes a loss combining the distance to both $\mathscr{D}_{NN}$ and $\mathscr{D}_{AN}$, while WNAMM extends NAMM with configurable weights for each distance term. NMM and AMM consider a single distance difference, while NAMM simultaneously considers two distance differences:
  - Positive to Nearest negative label Max-Margin Loss (*NMM Loss*):

$$NMM\ Loss = \frac{1}{N}\sum_{i=1}^{N}\max[(\mathscr{D}_P(x_i) - \mathscr{D}_{NN}(x_i) + 1),\ 0] \tag{15}$$

  - Positive to Average negative label Max_Margin Loss (*AMM Loss*):

$$AMM\ Loss = \frac{1}{N}\sum_{i=1}^{N}\max[(\mathscr{D}_P(x_i) - \mathscr{D}_{AN}(x_i) + 1),\ 0] \tag{16}$$

  - Positive to Nearest and Average negative labels Max-Margin Loss (*NAMM Loss*):

$$NAMM\ Loss = \frac{1}{N}\left(\sum_{i=1}^{N}\max[(\mathscr{D}_P(x_i) - \mathscr{D}_{NN}(x_i) + 1),\ 0] + \sum_{i=1}^{N}\max[(\mathscr{D}_P(x_i) - \mathscr{D}_{AN}(x_i) + 1),\ 0]\right) \tag{17}$$

  - Weighted Positive to Nearest and Average negative labels Max-Margin Loss (*WNAMM Loss*):

$$WNAMM\ Loss = \frac{1}{N}\left(\sum_{i=1}^{N}\max[(\mu_0\mathscr{D}_P(x_i) - \mu_1\mathscr{D}_{NN}(x_i) + 1),\ 0] + \sum_{i=1}^{N}\max[(\mu_2\mathscr{D}_P(x_i) - \mu_3\mathscr{D}_{AN}(x_i) + 1),\ 0]\right), \quad \text{with weights } \mu_0,\ \mu_1,\ \mu_2,\ \mu_3\ > 0 \tag{18}$$

- **Max-Margin+Min-Separation losses:** The loss functions for the **Con(λ)** models Fig. 7) are the following; where (λ) stands for a dummy letter to be replaced by N, A) (Eqs. (19),(20)). They are novel contrastive losses including the distance to the positive label ($\mathscr{D}_P$) and one of the selected representatives of the negative labels (either $\mathscr{D}_{NN}$ or $\mathscr{D}_{AN}$). These losses try to make the distance to the positive label as small as possible, while keeping the distance to one of the representatives of the negative labels greater than a margin (Max-Margin+Min-Separation)(Fig. 1):
  - Contrastive Loss with positive and nearest negative labels (Con*N Loss*):

$$ConN\ Loss = \frac{1}{N}\sum_{i=1}^{N}\mathscr{D}_P(x_i) + \max[(1 - \mathscr{D}_{NN}(x_i)),\ 0] \tag{19}$$

  - Contrastive Loss with positive and average negative labels (Con*A Loss*):

$$ConA\ Loss = \frac{1}{N}\sum_{i=1}^{N}\mathscr{D}_P(x_i) + \max[(1 - \mathscr{D}_{AN}(x_i)),\ 0] \tag{20}$$

- **Max-Separation+Min-Separation losses:** The loss functions for the **E(λ)MS** and **E2(λ)MS** models Fig. 7) are the following; where (λ) stands for a dummy letter to be replaced by N, NA, WNA) (Eqs. (21)–(24)). They are novel exponential or squared exponential losses for the difference of distance between the positive label ($\mathscr{D}_P$) and one of the selected representatives of the negative labels ($\mathscr{D}_{NN}$, $\mathscr{D}_{AN}$). The exponential function (*exp*) grows rapidly for increasing positive values (difference of distances) and moves to zero also rapidly for negative values. The loss produces small values of $\mathscr{D}_P$ and large values for $\mathscr{D}_{NN}$ or $\mathscr{D}_{AN}$, which results in trying to separate, as much as possible, the positive and negative labels from the reference sample (Max-Separation+Min-Separation) (Fig. 1). The squared exponential function ($(exp())^2$) makes this behavior even more

pronounced. The loss E2NAMS combines the $\mathscr{D}_{NN}$ and $\mathscr{D}_{AN}$ distances, and E2WNAMS is its weighted version:

- Positive to Nearest negative label Maximum Separation with Exponential Loss (*ENMS Loss*):

$$ENMS\ Loss = \frac{1}{N}\sum_{i=1}^{N} exp[\mathscr{D}_P(x_i) - \mathscr{D}_{NN}(x_i)] \quad (21)$$

- Positive to Nearest negative label Maximum Separation with squared Exponential Loss (*E2NMS Loss*):

$$E2NMS\ Loss = \frac{1}{N}\sum_{i=1}^{N} (\exp[\mathscr{D}_P(x_i) - \mathscr{D}_{NN}(x_i)])^2 \quad (22)$$

- Positive to Nearest and Average negative label Maximum Separation with squared Exponential Loss (*E2NAMS Loss*):

$$E2NAMS\ Loss = \frac{1}{N}\sum_{i=1}^{N} (\exp[\mathscr{D}_P(x_i) - (\mathscr{D}_{NN}(x_i) + \mathscr{D}_{AN}(x_i))])^2 \quad (23)$$

- Weighted Positive to Nearest and Average negative labels Maximum Separation with squared Exponential Loss (*E2WNAMS Loss*):

$$E2WNAMS\ Loss = \frac{1}{N}\sum_{i=1}^{N} (\exp[\mu_0\mathscr{D}_P(x_i) - \mu_1\mathscr{D}_{NN}(x_i) - \mu_2\mathscr{D}_{AN}(x_i)])^2, \quad (24)$$

with weights $\mu_0,\ \mu_1,\ \mu_2\ > 0$

3.4.2. Cross entropy over labels with contrastive regularization

The previous models use the distance between embeddings as the transformed features used for classification in a contrastive framework. In these models we will use the embeddings and their distances as the features used in a classic supervised classification framework based in a cross-entropy (CE) loss and a softmax nonlinear activation function. In this latter case, we will use the max-margin and max/min-separation losses (*e.g.*, NMM, AMM, ENMS...) as a **regularizer** input to a combined loss function that incorporate these losses together with the cross-entropy loss. The objective here is to reduce the error between expected and predicted labels (achieved by the CE loss) while keeping the distances between positive/negative labels as close/separate as possible from the samples (regularization term). This approach corresponds to the CE+(λ)MM and CE+E(λ)MS architectures (Fig. 8).

The **Cross Entropy over Labels with Contrastive Regularization architectures (CE+(λ)MM, CE+E(λ)MS)** Fig. 8) is based in extending the previously presented losses with a categorical Cross Entropy (CE) (Eq. (25)) term. The novel losses for the CE+(λ)MM and CE+E(λ)MS

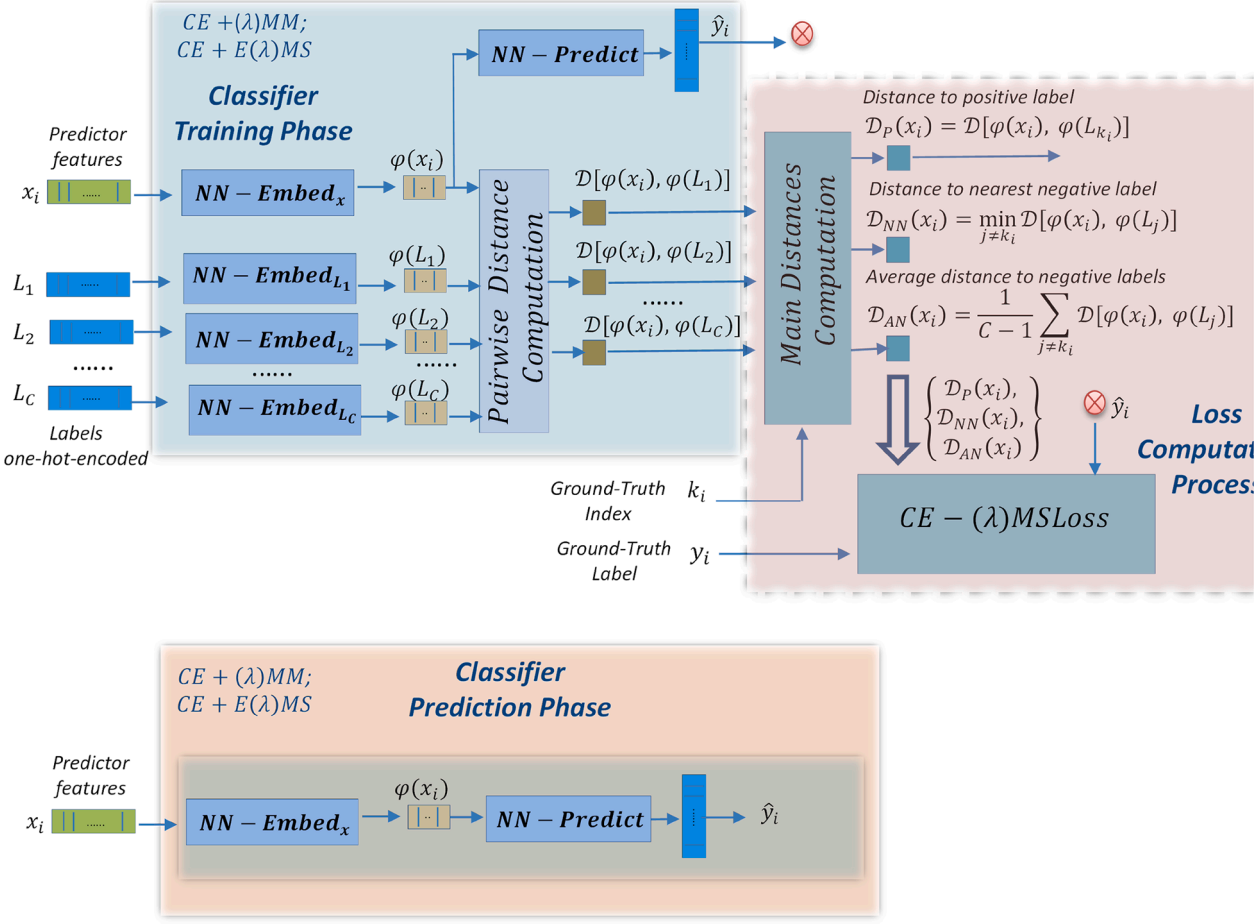

Fig. 8. ***CE+(λ)MM and CE+E(λ)MS architectures:*** General architecture for the models with a cross entropy term added to the loss function. The addition of this extra term requires to use the ground-truth label (y_i) for network training (**Upper chart**) and to produce a label prediction ($\hat{y}_i$). This model directly produces a label prediction facilitating the prediction phase (**Lower chart**)

architectures (Fig. 8) are presented in Eqs. (26)–(29); where, as above, (λ) stands for a dummy letter to be replaced by N, A or NA:

$$y_i^j = \begin{cases} 1 \text{ if } j = k_i \\ 0 \text{ if } j \neq k_i \end{cases} \text{with } \widehat{y}_i^j \in [0,1] \text{ and } \sum_{j=1..C} \widehat{y}_i^j = 1$$

$$CrossEntropy\left(y_i, \widehat{y}_i\right) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_i^j \log\left(\widehat{y}_i^j\right) \quad (25)$$

- CE Loss plus Positive to Nearest negative label Max-Margin Loss (*CE + NMM Loss*):

$$CE + NMM\ Loss = NMMLoss + CrossEntropy\left(y_i, \widehat{y}_i\right) \quad (26)$$

- CE Loss plus Positive to Average negative label Max-Margin Loss (*CE + AMM Loss*):

$$CE + AMM\ Loss = AMM\ Loss + CrossEntropy\left(y_i, \widehat{y}_i\right) \quad (27)$$

- CE Loss plus Positive to Nearest and Average negative labels Max-Margin Loss (*CE + NAMM Loss*):

$$CE + NAMM\ Loss = NAMM\ Loss + CrossEntropy\left(y_i, \widehat{y}_i\right) \quad (28)$$

- CE Loss plus Positive to Nearest negative Maximum Separation with Exponential Loss (*CE + ENMSLoss*):

$$CE + ENMS\ Loss = ENMS\ Loss + CrossEntropy\left(y_i, \widehat{y}_i\right) \quad (29)$$

Note: In the above expressions: $\widehat{y}_i$ is the predicted label for sample x_i. It is a vector of dimension C (*i.e.*, $\widehat{y}_i|_{j=1..C}$), generated by a neural network (*NN*) *i.e.*, $\widehat{y}_i = NN - Predict[\ \varphi(x_i)]$ (Fig. 8) with a softmax activation in the last layer. Each vector component ($\widehat{y}_i^j$) is interpreted as the probability that label L_j is associated with sample x_i. Additionally, y_i is the ground-truth label for x_i, where y_i^j is its one-hot-encoded representation with $y_i^j = 1$ if $j = k_i$ and $y_i^j = 0$ if $j \neq k_i$, where k_i corresponds to the index of the positive label of x_i

These models inherit the separation strategy provided by the contrastive part of the loss function, for example, a CE+(λ)MM model will implement a Max-margin separation strategy (same as (λ)MM), while a CE+E(λ)MS will implement a Max-Separation+Min-Separation separation strategy (same as E(λ)MS). The CE+(λ)MM and CE+E(λ)MS architectures Fig. 8) are identical to their counterparts: (λ)MM and E(λ)MS, with the addition of a small neural network: NN − Predict, which has as inputs the features embedding and generate the predicted label ($\widehat{y}_i$). The distances between embeddings plus the predicted and ground-truth label will be the inputs to the compound losses for this architecture (Eqs. (26)–(29)). During training, the model requires C+3 inputs per sample: the sample features, all C labels (one-hot-encoded), an integer k_i acting as the index of the positive label in the set {1… C} and the

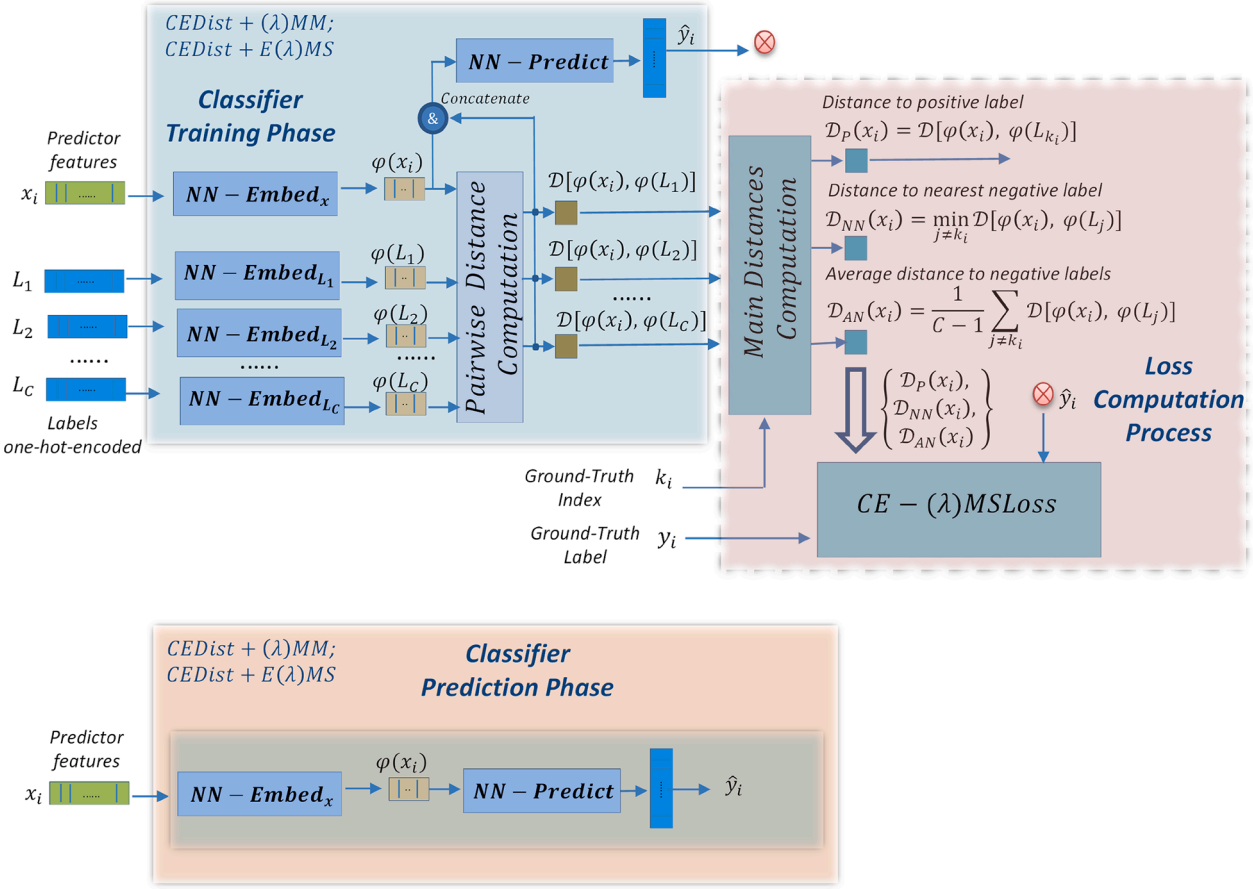

Fig. 9. ***CEDist+(λ)MM and CEDist+E(λ)MS architectures:*** Similar architecture to CE+(λ)MM and CE+E(λ)MS, where the neural network that generates the predicted label ($\widehat{y}_i$) takes as input the features embedding plus the distance of x_i to all labels.

Table 1
Application scenarios of the proposed models: comparison of global characteristics of the different models.

		Global evaluation	Complexity	Execution times
LB-CL	*Max Margin over Label Embeddings* (*Section 3.3.1*)	Very poor prediction performance	Low	Binary: Intermediate Multiclass: High
	Contrastive over Label Embeddings (Section 3.3.2)	- Best prediction performance of all models for binary classification - High prediction times for multiclass classification which increase with the number of classes - Increasing the dimensionality of the embedding space does not have a consistent impact on performance. Models provide excellent performance with a dimension of 2 - For multi-class classification they require many more training samples than alternative RLB-CL models. They behave poorly for small datasets.	Low	Binary: Intermediate Multiclass: High
	Contrastive with Cross Entropy (Section 3.3.3)		Low	Binary: Intermediate Multiclass: High
RLB-CL	*Representative Label Contrastive* (*Section 3.4.1*)	- These models behave in general quite similarly, providing the best prediction performance for multiclass classification and with the best specific model depending on the dataset. - These models share three main architectures with different variants depending on the loss function that is applied. - Low prediction times for multiclass classification and invariant with the number of classes - Using exclusively the average point of the cluster of negative labels provides the worst performance (e.g., AMM). - Using the nearest point to the cluster of negative labels or a combination of representative points provides the best performance (e.g., NMM, ENMS, E2NMS, NAMM,). - Computing the average point of the cluster of negative labels has an impact on training and prediction times. It is faster to compute the nearest point to this cluster. - Models that include cross-entropy provide some of the best prediction results. They have the highest complexity without a significant impact on their prediction times. - Increasing the dimensionality of the embedding space does not have a consistent impact on performance. Models provide excellent performance with a dimension of 2. - They require fewer training samples than LB-CL models and are more robust against small datasets - The two separation strategies (Fig. 1): Max-margin and Max-Separation+Min-Separation, seem to have quite similar prediction performance results. Meanwhile, the strategy: Max-Margin+Min-Separation seems to have a lower prediction performance.	Intermediate	Low
	Cross Entropy over Labels with Contrastive Regularization (*Section 3.4.2*)		High	Low
	Cross Entropy over Labels and Distances with Contrastive Regularization (*Section 3.4.3*)		High	Low

ground-truth label (y_i). This latter input is required functionally but not physically since can be extracted by knowing the index of the positive label (k_i). In the prediction phase (Fig. 8), we only require the sample features as input to the model and the features embedding NN ($NN-Embed_x$) and the prediction NN ($NN-Predict$) to generate the predicted label. This approach reduces even more the prediction times, as corroborated by experimental results (c.f. Section 4).

3.4.3. Cross entropy over labels and distances with contrastive regularization

The **Cross Entropy over Labels and Distances with Contrastive Regularization architectures (CEDist+(λ)MM, CEDist+E(λ)MS)** (Fig. 9), are a variant of their counterpart CE+(λ)MM and CE+E(λ)MS models, where $\hat{y}_i$ is a function of the features embedding ($\varphi(x_i)$) and the distances between each label embedding ($\varphi(L_j)|_{j=1..C}$) and the features embedding. *i.e.*, $\hat{y}_i = NN-Predict[\varphi(x_i), \mathscr{D}[\varphi(x_i), \varphi(L_1)], \ldots \mathscr{D}[\varphi(x_i), \varphi(L_C)]]$. To generate the predicted label ($\hat{y}_i$), we concatenate all the inputs to the neural network. Apart from this difference, this architecture is completely similar to its CE+(λ)MM and CE+E(λ)MS counterparts.

The complexity of the neural networks implementing the embedding functions are quite low. For the $NN-Embed_L$: 1 hidden layer with 10 nodes, ReLU activation and linear activation for the output layer. For the $NN-Embed_x$: 2 hidden layers with 100/50 nodes, ReLU activation and linear activation for the output layer. All the embedding NNs share the same configuration. Likewise, for the models with added Cross Entropy, the NN that implements the final classification stage ($NN-Predict$), is also small: 2 hidden layers with 20/20 nodes, ReLU activation and softmax activation for the output layer.

3.5. Summary of proposed models and application scenarios

This Section presents a summary of the main characteristics of the proposed models with a comparison between them Table 1 in terms of prediction performance, complexity and execution times (prediction phase). This information is extracted from their architectures and from the results obtained with the two datasets used in the experiments

Table 2
Taxonomy of contrastive learning solutions: unified classification of the solutions following the presentation given in Figs. 1 and 3.

	Type of embeddings	Separation strategy	Labels availability	
			No: Unsupervised/Self-supervised (Type I.A)	**Yes:** Supervised
Type I: Published research	*Using exclusively feature embeddings*	*Max-margin*	Triplet [7], RankedList [13]	SupCon [4] (single negative label) (Type I.B)
		Max-Margin + Min-Separation	Contrastive [8,14], Lifted [9]	Solutions not available
		Max-Separation + Min-Separation	N-pair [5], NT-Xent [15], ProxyNCA [10,16], InfoNCE [6]	SupCon [4] (several negative labels) (Type I.B)
Type II: LB-CL	*Using feature embeddings and label embeddings (with all labels)*	*Max-margin*	N/A	- Max Margin over Label Embeddings (MMoLE
		Max-Margin + Min-Separation		- Contrastive over Label Embeddings (ConLE),
		Max-Separation + Min-Separation		- Contrastive with Cross Entropy (ConCE)
Type III: RLB-CL	*Using feature embeddings and label embeddings (with two labels only: sample's positive label and a single proxy-label as a representative of all its negative labels)*	*Max-margin*		- Max-Margin ((λ)MM) - Cross Entropy with Max-Margin (CE+(λ)MM) - Cross Entropy plus Distances with Max-Margin (CEDist+(λ)MM)
		Max-Margin + Min-Separation		- Contrastive (Con(λ))
		Max-Separation + Min-Separation		- Exponential loss for Max-Separation (E(λ)MS), - Squared Exponential loss for Max-Separation (E2(λ)MS), - Cross Entropy with Exponential loss for Max-Separation (CE+E(λ)MS) - Cross Entropy plus Distances with Exponential loss for Max-Separation (CEDist+E(λ)MS)

(Section 4). The comparison provided in Table 1 can be useful to establish the application scenarios of the different models.

Table 2 provides a unified summary classification of the different contrastive learning solutions which were presented under two different points of view in Figs. 1 and 3. The summary classification in Table 2 offers a taxonomy of the contrastive learning solutions based on the following differentiation parameters: a) the types of embeddings used, b) the separation strategy, and c) labels availability (whether the models are either supervised or unsupervised/self-supervised)

All the models presented in this work have been implemented with neural networks trained with gradient descent, with a batch size of 100, 100 epochs and early-stopping with a waiting period of 10 epochs and a validation set of 20% of the training set. The optimizer employed was Adam with the original default parameters [41]. The source code for the most representative models of the proposed solutions is provided in a freely accessible repository [42].

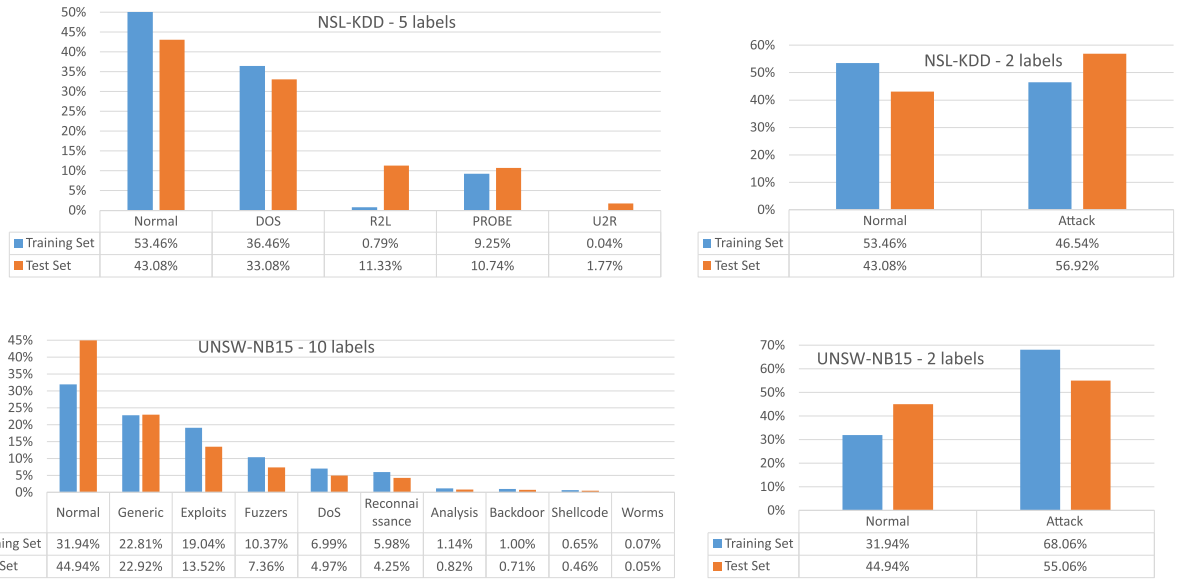

Fig. 10. Labels distribution in the training and test sets for NSL-KDD (**Upper charts**) and UNSW-NB15 (**Lower charts**) for multi-class (5 or 10 labels) and binary (2 labels) classification scenarios.

4. Results

4.1. Selected datasets

We have selected two network intrusion detection datasets (NSL-KDD and UNSW-NB15) as experimental benchmarks to apply the proposed models. They are good representatives of a field characterized by noisy and unbalanced datasets.

The NSL-KDD dataset [43] is a well-known intrusion detection (ID) dataset. It contains 125,973 training samples and 22,544 test samples with 122 features (after one-hot encoding categorical features). The continuous features have been scaled in the [0,1] range. It has 40 labels with a dissimilar frequency distribution in the training and test sets. These 40 labels are usually aggregated into 5 or 2 labels [43] (Fig. 10). The UNSW-NB15 [44] is a newer ID dataset. It contains 2,540,044 training samples and 82,332 test samples. After a similar pre-processing (one-hot encoding and scaling) to that performed for NSL-KDD, the final number of features is 196. It has 10 distinct labels which can be aggregated into 2 (normal and attack) (Fig. 10). NSL-KDD and UNSW-NB15 present very different label distributions for the training and test sets with the common characteristic of being noisy and strongly unbalanced. The scenarios presented by the two datasets under their different multi-class and binary configurations allow exploring the performance of the proposed model in a wide range of challenging contexts.

4.2. Performance metrics

We have used a complete set of metrics to compare results for the two datasets with the different models. We provide metrics for classification, clustering and training/prediction execution times. The classification metrics applied are: accuracy, F1-score, precision, recall, and Matthews Correlation Coefficient (MCC) with their usual definitions [45,46]. MCC is particularly useful for unbalanced datasets. The metric for clustering quality are Normalized Mutual Information (NMI) [5,9,10] and Silhouette coefficient [47]. NMI is invariant to label permutation, which is needed to perform a correct clustering evaluation, it is also a label based metric assessing the correct identification of labels to clusters. Silhouette coefficient is an unsupervised metric that is not based on knowing the ground-truth label; it provides a good indication of the separation between clusters. To identify the execution times of the algorithms, we present the training and prediction times as a comparative indicator of complexity and computational load, knowing that these values cannot be considered in an absolute but relative way to help the comparison between algorithms, since their absolute values depend on the nature and capacity of the processor used.

All the proposed metrics are associated to better results in a monotonic increasing way; their range of values is [0,1], with the exception of MCC which is a correlation coefficient with a range [-1,1] (where +1 indicates a high correlation between ground-truth and predicted results and -1 total disagreement between them) and the Silhouette coefficient with also a range [-1,1] (where +1 indicates a high separation between clusters and -1 highly mixed clusters).

Clustering scores have not been considered for the ML models because it is not the purpose of these models to perform clustering and, additionally, the high dimensionality of the original feature space (>100) makes clustering a challenging task in itself, with different applicable techniques and interpretation of results [48]. It is precisely the translation of the original features into a low-dimension embedding space (in LB-CL and RLB-CL models) that allows the classification task to be interpreted from a clustering perspective.

4.3. NSL-KDD results

The results for the NSL-KDD dataset are divided by the type of classification into binary classification (2 labels) and multi-class classification (5 labels) (Section 4.1). We focus on providing the results of several representative LB-CL and RLB-CL models (Sections 3.3 and 3.4) in comparison with several well-known machine learning (ML) models: logistic regression, random forest, Gradient Boosting Machine (GBM), Support Vector Machine with radial kernel (SVM-RBF), Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN) and a Linear Model with a Kernel Approximation (LM+KA). The last two models have been selected for having obtained very good performance results in recent works for both NSL-KDD and UNSW-NB15 datasets [49], and CNN models are some of the most widely used deep learning models in IDS [20,21,50]. In particular, we will use a CNN with a one-dimensional kernel (CNN-1D) due to the one-dimensional vector structure of the features in the two datasets used in this work. The LM+KA models [49] correspond to a recent trend towards fast shallow linear models that can handle non-linearities through a feature transformation that approximates a kernel SVM. The interest of these models lies in being especially fast. We have not considered applying recurrent networks (*e.g.*, long short term memory (LSTM)) as the selected datasets are not based on sequential data.

Two types of results are provided: (1) classification results using the models (ML, LB-CL and RLB-CL) and (2) improvement of the classification results of the ML models when using the original features versus the transformed features obtained by the LB-CL and RLB-CL models.

4.3.1. Classification with proposed models

A comparison of classification and clustering performance metrics for a representative number of ML, LB-CL and RLB-CL models is provided in Fig. 11. The metrics are separated into two groups by the number of labels to detect (2 and 5 labels). For the LB-CL and RLB-CL models, we identify the dimension of the embedding space and the distance adopted between embeddings (Cosine or Euclidean). All results are computed over the test set (Section 4.1). The two rightmost columns in Fig. 10 provide the number of trainable weights and the number of Floating Point Operations (Flops) required for each model. These two metrics, along with the training and prediction execution times, offer a good insight into the computational complexity and performance of the models. A more detailed analysis of the complexity of the models is provided in Section 4.7.

Fig. 11 shows the two best results in bold-italic text style. Additionally, a color code is used where a dark-green is associated with better results and a dark-red with worse results, with an interpolated color-palette that is used for intermediate values. The color code and the best-two values are applied column-wise and independently (separately) for the blocks of 2 and 5 labels.

2 labels

For binary classification, the LB-CL models (Type II) present best overall results (F1 and MCC) for classification and clustering. Model complexity is also smaller for LB-CL models. Considering training and prediction times, the RLB-CL models have the best times, which is as expected since the number of comparisons is less than for LB-CL models. The reduction in the execution times of the proposed models is at least an order of magnitude compared with most of the classic ML models. Interestingly, the training and prediction times of the RLB-CL models are better than those of the logistic regression and LM + KA models which are specially designed to be fast.

5 labels

For multi-class classification (5 labels) the RLB-CL models (Type III) present some of the best classification metrics after the CNN-1D and LM+KA models which have previously shown particularly good performance for this dataset [49]. The RLB-CL models have also some of the best clustering metrics (NMI metric). The best prediction times are for the RLB-CL models and second best for the training times (after LM+KA model). The best RLB-CL model is ENMS considering F1 and MCC as our main classification metric. It is interesting to note the poor behavior of AMM, which incorporates exclusively the average distance to the group of negative labels. The unsupervised clustering metric (Silhouette)

	Type	Classifier	Embed. Dim.	Distance	Classification scores: Accuracy	F1	Precision	Recall	MCC	Clustering scores: Silhouette	NMI	Exec. Times (min): Training	Prediction	Nº of Weights	Flops
2 Labels	ML	Logistic Regression			0.7639	0.7576	0.9116	0.6481	0.5700			26.06	0.012		
		Random Forest			0.7497	0.7258	***0.9642***	0.5819	0.5771			13.68	0.344		
		GBM			0.7951	0.7859	***0.9700***	0.6605	0.6439			44.66	0.065		
		SVM-RBF			0.7616	0.7257	0.8085	0.7616	0.6487			105.69	16.113		
		MLP			0.7957	0.7956	0.9242	0.6984	0.6228			68.74	0.066		
		CNN-1D			0.8056	0.7994	0.9688	0.6804	0.6327			***14.04***	***0.046***		
		LM+KA			0.8870	0.8944	0.9559	0.8403	0.7395			***1.79***	***0.009***		
	LB-CL	ConCE	2	Cosine	***0.9050***	***0.9176***	0.9061	0.9294	***0.8058***	0.6538	0.9807	4.46	0.021	***17506***	***34689***
		ConCE	4	Cosine	***0.9094***	***0.9222***	0.9018	0.9436	***0.8151***	0.7053	0.9812	2.67	0.024	17630	34929
		ConCE	6	Cosine	0.8892	0.9110	0.8393	***0.9962***	0.7871	0.7189	0.9828	2.74	0.021	17754	35169
		ConCE	8	Cosine	0.8950	0.9145	0.8520	***0.9870***	0.7946	0.6772	0.9835	5.43	0.022	17878	35409
		ConCE	10	Cosine	0.8600	0.8697	0.9248	0.8208	0.7256	0.6931	0.9821	2.86	0.025	18002	35649
		ConLE	2	Euclid.	0.8829	0.8935	0.9264	0.8628	0.7662	***0.9930***	***0.9931***	8.20	0.022	***17504***	***34687***
		ConLE	10	Euclid.	0.8911	0.9050	0.8989	0.9112	0.7776	***0.9932***	***0.9934***	8.59	0.027	18000	35647
	RLB-CL	E2NMS	2	Euclid.	0.8872	0.9036	0.8798	0.9286	0.7695	0.7542	0.9737	2.72	0.013	17556	34768
		E2NMS	8	Euclid.	0.8816	0.8936	0.9144	0.8738	0.7612	0.7779	0.9691	***1.64***	0.013	17994	35608
		ENMS	2	Euclid.	0.8763	0.8949	0.8668	0.9249	0.7474	0.7327	0.9760	2.88	0.011	17556	34769
		NMM	2	Euclid.	0.8125	0.8093	0.9615	0.6987	0.6653	0.7880	0.9782	7.87	0.011	17556	34769
		NMM	8	Euclid.	0.8112	0.8072	0.9640	0.6943	0.6645	0.7741	0.9795	4.27	0.012	17994	35609
		CE+NMM	2	Euclid.	0.8770	0.8969	0.8576	0.9400	0.7501	0.7689	0.9858	4.08	***0.007***	20358	40176
		CEDist+NMM	2	Euclid.	0.8857	0.9029	0.8744	0.9333	0.7668	0.7084	0.9902	15.01	0.014	20458	40376
		CEDist+NMM	8	Euclid.	0.8369	0.8454	0.9178	0.7836	0.6844	0.7514	0.9878	5.73	0.014	21196	41816
		ConN	2	Euclid.	0.8268	0.8332	0.9218	0.7601	0.6697	0.9915	0.9857	7.49	0.020	17556	34771
		ConN	8	Euclid.	0.8240	0.8300	0.9215	0.7551	0.6652	0.9911	0.9846	6.09	0.019	17994	35611
5 Labels	ML	Logistic Regression			0.7652	0.7358	0.7702	0.7652	0.6518			32.16	0.010		
		Random Forest			0.7276	0.6845	0.7860	0.7276	0.6053			14.13	0.398		
		GBM			0.7625	0.7291	0.8053	0.7625	0.6549			238.38	0.272		
		SVM-RBF			0.7616	0.7257	0.8085	0.7616	0.6487			108.97	16.120		
		MLP			0.7618	0.7317	0.7538	0.7618	0.6466			87.33	0.067		
		CNN-1D			0.7875	***0.7633***	0.8094	0.7875	***0.6985***			13.35	0.043		
		LM+KA			***0.8068***	***0.7930***	***0.8179***	***0.8068***	***0.7125***			***1.80***	***0.010***		
	LB-CL	ConCE	2	Cosine	0.7597	0.7279	0.7405	0.7597	0.6416	0.3539	0.9678	6.40	0.058	***17536***	***34749***
		ConCE	10	Cosine	0.7775	0.7430	***0.8219***	0.7775	0.6766	0.6549	0.9635	7.57	0.054	18032	35709
		ConLE	2	Euclid.	0.7547	0.7183	0.7885	0.7547	0.6362	***0.9950***	***0.9947***	15.73	0.059	***17534***	***34747***
		ConLE	10	Euclid.	0.7729	0.7474	0.8104	0.7729	0.6653	***0.9899***	0.9920	23.03	0.057	18030	35707
	RLB-CL	E2NMS	2	Euclid.	0.7831	0.7516	0.7846	0.7831	0.6837	0.7473	0.9827	3.88	0.018	17862	35314
		E2NMS	10	Euclid.	0.7725	0.7429	0.8106	0.7725	0.6647	0.7153	0.9854	3.46	0.019	18710	36914
		E2NAMS	2	Euclid.	0.1702	0.1501	0.1412	0.1702	-0.0397	0.2459	0.2026	14.26	0.019	17862	35314
		E2WNAMS (1,0.5,0.5)	2	Euclid.	0.7851	0.7528	0.7703	0.7851	0.6862	0.6971	0.9763	3.61	0.019	17862	35314
		ENMS	6	Euclid.	0.7637	0.7207	0.7998	0.7637	0.6525	0.7080	0.9795	2.83	0.019	18286	36115
		ENMS	2	Euclid.	***0.7906***	0.7530	0.7931	***0.7906***	0.6955	0.7034	0.9807	6.39	0.027	17862	35315
		ENMS	10	Euclid.	0.7859	0.7536	0.7970	0.7859	0.6876	0.6225	0.9738	***2.80***	0.020	18710	36915
		CE+ENMS	2	Euclid.	0.7803	0.7454	0.8111	0.7803	0.6804	0.7768	0.9831	7.36	***0.008***	23467	46222
		NMM	2	Euclid.	0.7551	0.7376	0.7808	0.7551	0.6383	0.7391	0.9925	24.19	0.020	17862	35315
		NMM	10	Euclid.	0.7796	0.7491	0.7904	0.7796	0.6765	0.6291	0.9932	11.21	0.017	18710	36915
		CE+NMM	2	Euclid.	0.7495	0.7158	0.7572	0.7495	0.6291	0.6638	***0.9946***	28.83	0.017	23467	46222
		CEDist+NMM	2	Euclid.	0.7458	0.7096	0.7240	0.7458	0.6223	0.6073	0.9939	22.81	0.027	24367	47922
		CEDist+NMM	4	Euclid.	0.7763	0.7457	0.7968	0.7763	0.6691	0.6138	0.9919	7.86	0.024	24879	48922
		WNAMM (1,0.5,1,0.5)	2	Euclid.	0.7555	0.7214	0.8032	0.7555	0.6392	0.9258	0.9929	12.73	0.023	18262	36017
		NAMM	2	Euclid.	0.7716	0.7442	0.7730	0.7716	0.6628	0.6018	0.9907	6.14	0.018	18262	36017
		NAMM	10	Euclid.	0.7597	0.7327	0.7844	0.7597	0.6461	0.5411	0.9924	8.90	0.012	19510	38417
		AMM	2	Euclid.	0.6079	0.5943	0.6315	0.6079	0.4434	0.3703	0.3794	3.56	0.018	17862	35315
		AMM	10	Euclid.	0.4499	0.3885	0.5074	0.4499	0.2875	0.2599	0.2665	3.03	0.018	18710	36915
		ConN	2	Euclid.	0.7749	0.7287	0.8138	0.7749	0.6737	0.9876	0.9917	12.18	0.011	17862	35317
		ConN	10	Euclid.	0.7609	0.7280	0.7881	0.7609	0.6440	0.9877	0.9897	7.91	0.010	18710	36917

Fig. 11. NSL-KDD dataset: comparison of classification and clustering performance metrics between a selection of classic ML models vs. a representative set of the proposed LB-CL and RLB-CL models. A color code is used where dark-green is for better results and dark-red for worse results, an interpolated color-palette is used for intermediate values. The best-two values are in bold-italic. The color code and the best-two values are applied column-wise and separately for the blocks of 2 and 5 labels. All results are computed over the test set (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.).

obtains its maximum with contrastive models (ConLE) in both binary and multi-class scenarios. The best overall performance results are for the LM+KA model which was selected for being particularly effective with this dataset and being a tough competitor to other models. Nevertheless, considering the prediction performance and prediction speed together, the ENMS model has the third best prediction result with very short prediction times, while a similar model (CE+ENMS) provides the shortest prediction times.

4.3.2. *Improvement of ML models*

As mentioned in the introduction, the embedding of the features and the distances between them and each of the label embeddings can be

	Features	Nº features	Classifier	Classification Scores: Accuracy	F1	Precision	Recall	MCC	Exec. Times (sec): Training	Prediction	Features generation
2 Labels	Original features: x_i	122	Logistic Regression	0.7639	0.7576	0.9116	0.6481	0.5700	26.06	0.012	N/A
		122	Random Forest	0.7497	0.7258	***0.9642***	0.5819	0.5771	***13.68***	0.344	
		122	GBM	0.7951	0.7859	***0.9700***	0.6605	0.6439	44.66	0.065	
		122	SVM-RBF	0.7616	0.7257	0.8085	***0.7616***	0.6487	105.69	16.113	
		122	MLP	0.7957	0.7956	0.9242	0.6984	0.6228	68.74	0.066	
			Average	0.7732	0.7581	0.9157	0.6701	0.6125	51.76	3.320	
			Std. Dev.	0.0210	0.0327	0.0650	0.0662	0.0370	36.60	7.153	
	Features embedding + Distances: $\varphi(x_i), \{\mathcal{D}[\varphi(x_i), \varphi(L_j)]\}_{j=1..2}$ Generated with NMM (embedding dimension: 2)	4	Logistic Regression	***0.8152***	***0.8126***	***0.9606***	***0.7042***	***0.6688***	***1.35***	***0.003***	0.72
		4	Random Forest	***0.8155***	***0.8150***	0.9496	***0.7138***	***0.6643***	26.74	***0.212***	
		4	GBM	***0.8172***	***0.8163***	0.9534	***0.7137***	***0.6685***	***29.86***	***0.035***	
		4	SVM-RBF	***0.8116***	***0.8081***	***0.9619***	0.6966	***0.6641***	***6.18***	***0.516***	
		4	MLP	***0.8193***	***0.8190***	***0.9528***	***0.7181***	***0.6714***	***19.02***	***0.055***	
			Average	***0.8157***	***0.8142***	***0.9556***	***0.7093***	***0.6674***	***16.63***	***0.164***	
			Std. Dev.	***0.0028***	***0.0041***	***0.0053***	***0.0087***	***0.0031***	***12.50***	***0.213***	
5 Labels	Original features: x_i	122	Logistic Regression	0.7652	0.7358	0.7702	0.7652	0.6518	32.16	***0.010***	N/A
		122	Random Forest	0.7276	0.6845	0.7860	0.7276	0.6053	14.13	0.398	
		122	GBM	0.7625	0.7291	***0.8053***	0.7625	0.6549	238.38	0.272	
		122	SVM-RBF	0.7616	0.7257	***0.8085***	0.7616	0.6487	108.97	16.120	
		122	MLP	0.7618	0.7317	0.7538	0.7618	0.6466	87.33	0.067	
			Average	0.7557	0.7214	0.7848	0.7557	0.6415	96.20	3.373	
			Std. Dev.	0.0158	0.0209	0.0232	0.0158	0.0204	88.45	7.127	
	Features embedding + Distances: $\varphi(x_i), \{\mathcal{D}[\varphi(x_i), \varphi(L_j)]\}_{j=1..5}$ Generated with NMM (embedding dimension: 10)	15	Logistic Regression	***0.7845***	***0.7607***	***0.7928***	***0.7845***	***0.6831***	***26.09***	0.038	1.71
		15	Random Forest	***0.7801***	***0.7477***	***0.7900***	***0.7801***	***0.6785***	***5.62***	***0.026***	
		15	GBM	***0.7749***	***0.7399***	0.7733	***0.7749***	***0.6708***	***195.81***	***0.118***	
		15	SVM-RBF	***0.7814***	***0.7500***	0.7924	***0.7814***	***0.6799***	***50.35***	***1.628***	
		15	MLP	***0.7809***	***0.7514***	***0.7903***	***0.7809***	***0.6790***	***37.50***	***0.062***	
			Average	***0.7803***	***0.7499***	***0.7877***	***0.7803***	***0.6783***	***63.07***	***0.374***	
			Std. Dev.	***0.0035***	***0.0075***	***0.0082***	***0.0035***	***0.0045***	***76.00***	***0.702***	

Fig. 12. NSL-KDD dataset: Performance classification metrics when using the original features versus the transformed ones produced by the NMM model (RLB-CL/Type III). The comparison is done with a representative set of classic ML models used as a reference. Bold-italic is used to mark the best value in a pair-wise comparison between each metric using original and transformed features. All results are computed over the test set.

used as a replacement for the original features, implementing a de facto supervised dimensionality reduction. These transformed features can be used as original features in a subsequent classifier, implementing a stacked-ensemble configuration [51]. In a stacked-ensemble, it is important not to incur in data leakage or compromise the test set in the sequence of classifiers used. To avoid using the test set in the learning process we use always the same LB-CL/RLB-CL model as base model (level 0 classifier) [51] and one of several classic ML models as meta-model (level 1 classifier) [51]. We do not use the best base model (Fig. 11) using the results of the test data, as it would involve using the information from the test set during training. Our proposed model for stacked generalization is also different from [51] because we do not use the predictions of the base model as inputs to the metamodel, but rather an intermediate transformation of the original features created by the base model.

Fig. 12 provides the results when using the original features versus the transformed ones produced by one of our proposed models (used as base model). The base model selected is the NMM model (RLB-CL/Type III). This model is the one used for all experiments in this Section and Section 4.4.2. We can observe how using the transformed features improves the prediction performance of all classic ML models used as a reference (logistic regression, random forest, GBM, SVM and MLP) compared with the same models using the original features. To facilitate the comparison, we have marked in bold-italic the best value in a pair-wise comparison between each metric using original and transformed features.

We identified F1 and MCC as the two metrics that best represent the overall performance of the models, and considering these metrics we observe (Fig. 12) the improvement obtained when using the transformed features. This improvement is achieved (for F1 and MCC) for all ML models. The average performance is also improved in all cases, with a significant reduction in the dispersion of values for the different models (standard deviation).

4.4. UNSW-NB15 results

The presentation of results for the UNSW-NB15 dataset follows the same principles that for the NSL-KDD dataset (Section 4.3). Therefore, we will not repeat the details provided in Section 4.3, focusing directly on the analysis of results.

4.4.1. Classification with proposed models

A comparison of classification and clustering performance metrics for a representative number of ML, LB-CL and RLB-CL models is provided in Fig. 13. The metrics are separated into two groups by the number of labels to detect (either 2 or 10 labels).

2 labels

For binary classification, the LB-CL models (Type II) present best results for classification and clustering. The training and prediction times are also among the smallest, principally for training, but at prediction the RLB-CL models have smaller times, which is as expected since the number of comparisons is less than those for LB-CL models. The reduction in execution times for the proposed models is more striking compared to the classic ML models, with a reduction of at least an order of magnitude compared to most ML models. The RLB-CL models provide shorter prediction times than even LM+KA.

10 labels

For multi-class classification (10 labels) the RLB-CL models (Type III) present the best classification metrics and some of the best clustering metrics (NMI metric). The best training and prediction times are also for

					Classification scores					Clustering scores		Exec. Times (min)			
	Type	Classifier	Embed. Dim.	Distance	Accuracy	F1	Precision	Recall	MCC	Silhouette	NMI	Training	Prediction	Nº of Weights	Flops
2 Labels	ML	Logistic Regression			0.8428	0.8673	0.8100	0.9334	0.6870			26.84	0.08		
		Random Forest			0.8619	0.8871	0.8069	0.9850	0.7377			3.04	0.23		
		GBM			0.8551	0.8822	0.7985	0.9855	0.7257			85.57	0.43		
		SVM-RBF			0.8087	0.8517	0.7430	***0.9975***	0.6512			3027.14	464.30		
		MLP			0.8493	0.8778	0.7926	0.9836	0.7146			588.59	0.52		
		CNN-1D			0.8976	0.9134	0.8552	0.9799	0.8132			11.74	0.11		
		LM+KA			0.8848	0.8946	0.9010	0.8883	0.8054			3.10	0.03		
	LB-CL	ConCE	2	Cosine	0.9090	0.9186	0.9049	0.9327	0.8159	0.5951	0.6736	***2.06***	0.07	***24906***	***49489***
		ConCE	4	Cosine	0.9092	0.9173	0.9201	0.9146	0.8168	0.5689	0.6739	2.53	0.14	25030	49729
		ConCE	6	Cosine	0.9061	0.9118	0.9439	0.8818	0.8138	0.6205	0.6743	3.06	0.09	25154	49969
		ConCE	8	Cosine	0.9059	0.9152	0.9086	0.9218	0.8097	0.5791	0.6740	2.54	0.08	25278	50209
		ConCE	10	Cosine	***0.9119***	0.9203	0.9174	0.9232	***0.8219***	0.5697	0.6849	4.58	0.08	25402	50449
		ConLE	2	Euclid.	0.9089	***0.9206***	0.8854	0.9587	0.8178	***0.8848***	***0.6924***	5.11	0.07	***24904***	***49487***
		ConLE	10	Euclid.	***0.9116***	***0.9203***	0.9131	0.9277	***0.8211***	***0.8639***	***0.6890***	4.33	0.08	25400	50447
	RLB-CL	E2NMS	2	Euclid.	0.8692	0.8659	***0.9936***	0.7673	0.7660	0.5981	0.6717	2.48	***0.03***	24956	49568
		E2NMS	8	Euclid.	0.8181	0.8577	0.7532	0.9960	0.6664	0.5698	0.6725	3.01	0.03	25394	50408
		ENMS	2	Euclid.	0.8370	0.8702	0.7748	0.9923	0.6971	0.5925	0.6759	4.54	0.03	24956	49569
		NMM	2	Euclid.	0.8125	0.8543	0.7465	***0.9986***	0.6585	0.8474	0.6729	2.76	0.03	24956	49569
		NMM	8	Euclid.	0.8163	0.8565	0.7516	0.9953	0.6630	0.8059	0.6672	***2.33***	0.03	25394	50409
		CE+NMM	2	Euclid.	0.8972	0.9029	0.9406	0.8682	0.7970	0.6903	0.6673	2.54	***0.03***	27758	54976
		CEDist+NMM	2	Euclid.	0.9078	0.9135	***0.9443***	0.8847	0.8169	0.5301	0.6793	6.11	0.03	27858	55176
		CEDist+NMM	8	Euclid.	0.8994	0.9076	0.9185	0.8969	0.7975	0.6348	0.6657	2.89	0.03	28596	56616
		ConN	2	Euclid.	0.8409	0.8709	0.7870	0.9749	0.6960	0.8620	0.6691	3.86	0.03	24956	49571
		ConN	8	Euclid.	0.8215	0.8591	0.7597	0.9884	0.6681	0.8625	0.6676	2.76	0.03	25394	50411
10 Labels	ML	Logistic Regression			0.6362	0.6937	0.8254	0.6362	0.5657			246.67	0.11		
		Random Forest			0.7037	0.7376	0.7899	0.7037	0.6145			***4.53***	0.41		
		GBM			0.7218	0.7296	0.7911	0.7218	0.6457			885.82	3.05		
		SVM-RBF			0.6605	0.6753	0.7844	0.6605	0.5920			4390.77	1410.22		
		MLP			0.7283	0.7446	0.8122	0.7283	0.6585			656.11	0.38		
		CNN-1D			***0.7821***	***0.7766***	0.8247	***0.7821***	0.6810			31.76	0.11		
		LM+KA			***0.7779***	0.7262	0.7634	***0.7779***	0.6510			4.86	***0.04***		
	LB-CL	ConCE	2	Cosine	0.4832	0.4536	0.4818	0.4832	0.2787	0.0086	0.3515	5.79	0.30	***24986***	***49649***
		ConCE	10	Cosine	0.6883	0.7151	0.8101	0.6883	0.6237	0.4377	0.6521	8.40	0.32	25482	50609
		ConLE	2	Euclid.	0.6881	0.7185	0.8134	0.6881	0.6237	***0.7674***	0.6770	27.44	0.58	***24984***	***49647***
		ConLE	10	Euclid.	0.7143	0.7472	***0.8362***	0.7143	0.6527	***0.7995***	0.6850	21.16	0.40	25480	50607
	RLB-CL	E2NMS	2	Euclid.	0.7494	0.7215	0.7109	0.7494	0.6546	0.0209	0.6300	5.61	***0.04***	26172	51824
		E2NMS	10	Euclid.	0.7428	0.7501	0.8063	0.7428	0.6737	0.2645	0.6877	7.07	0.04	27460	54224
		E2NAMS	2	Euclid.	0.6696	0.5530	0.4874	0.6696	0.5407	0.4024	0.5206	15.56	0.05	26172	51824
		E2WNAMS (1,0.5,0.5)	2	Euclid.	0.6674	0.6937	0.7385	0.6674	0.5683	0.2615	0.5669	***3.61***	0.06	26172	51824
		ENMS	6	Euclid.	0.7402	0.7612	0.8143	0.7402	0.6703	0.1576	0.6206	10.94	0.06	26816	53025
		ENMS	2	Euclid.	0.7311	0.7313	0.7494	0.7311	0.6420	0.1637	0.6177	8.90	0.05	26172	51825
		ENMS	10	Euclid.	0.7097	0.7281	0.8084	0.7097	0.6434	0.2506	0.6634	5.92	0.05	27460	54225
		CE+ENMS	2	Euclid.	0.7577	0.7606	0.8176	0.7577	***0.6886***	0.2929	0.6939	14.17	0.05	32032	63232
		NMM	2	Euclid.	0.7406	0.7414	0.7995	0.7406	0.6719	0.3204	0.6893	8.92	0.07	26172	51825
		NMM	10	Euclid.	0.7444	0.7669	0.8260	0.7444	0.6791	0.3996	0.6621	9.61	0.08	27460	54225
		CE+NMM	2	Euclid.	0.7606	0.7631	0.8171	0.7606	***0.6931***	0.2855	***0.7091***	20.08	0.06	32032	63232
		CEDist+NMM	2	Euclid.	0.7545	0.7583	0.8164	0.7545	0.6874	0.0690	***0.7087***	25.99	0.08	34332	67632
		CEDist+NMM	4	Euclid.	0.7303	0.7404	***0.8292***	0.7303	0.6693	0.3089	0.6967	9.58	0.09	35054	69032
		WNAMM (1,0.5,1,0.5)	2	Euclid.	0.7075	0.7349	0.8079	0.7075	0.6291	0.5089	0.5733	8.15	0.09	27472	54470
		NAMM	2	Euclid.	0.6958	0.6954	0.7831	0.6958	0.6266	0.3185	0.6544	17.56	0.10	27472	54470
		NAMM	10	Euclid.	0.7567	***0.7771***	0.8237	0.7567	0.6874	0.3008	0.6852	29.09	0.10	29560	58227
		AMM	2	Euclid.	0.0244	0.0170	0.0389	0.0244	-0.1131	0.3037	0.4232	8.50	0.12	***26172***	***51825***
		AMM	10	Euclid.	0.0619	0.0820	0.3519	0.0619	0.0215	0.3723	0.4190	7.46	0.13	27460	54225
		ConN	2	Euclid.	0.7362	0.7441	0.8024	0.7362	0.6689	0.5081	0.6915	12.17	0.05	***26172***	***51827***
		ConN	10	Euclid.	0.7499	0.7535	0.8135	0.7499	0.6818	0.6729	0.6938	11.18	0.05	27460	54227

Fig. 13. UNSW-NB15 dataset: comparison of classification and clustering performance metrics between a selection of classic ML models vs. a representative set of the proposed LB-CL and RLB-CL models. A color code is used where dark-green is for better results and dark-red for worse results, an interpolated color-palette is used for intermediate values. The best-two values are in bold-italic. The color code and the best-two values are applied column-wise and separately for the blocks of 2 and 10 labels. All results are computed over the test set (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.).

the RLB-CL models. The LM+KA model also has very short prediction times (as expected). The best RLB-CL models are NMM/NAMM or their cross-entropy counterparts depending on taking F1 or MCC as our main classification metric. An interesting finding is again the poor behavior of AMM, which indicates that exclusively using the average distance to the group of negative labels is not as good as considering the distance to the nearest negative label, or a combination of both distances. Similar to what happens with NSL-KDD, the unsupervised clustering metric

	Features	Nº features	Classifier	Classification Scores: Accuracy	F1	Precision	Recall	MCC	Exec. Times (sec): Training	Prediction	Features generation
2 Labels	Original features: x_i	196	Logistic Regression	0.8428	0.8673	0.8100	0.9334	0.6870	26.74	0.081	N/A
		196	Random Forest	***0.8619***	***0.8871***	***0.8069***	***0.9850***	***0.7377***	***3.00***	0.227	
		196	GBM	0.8551	0.8822	0.7985	***0.9855***	0.7257	82.14	0.449	
		196	SVM-RBF	0.8087	0.8517	0.7430	***0.9975***	0.6512	3027.14	464.299	
		196	MLP	0.8493	0.8778	0.7926	***0.9836***	0.7146	539.51	0.513	
			Average	0.8435	0.8732	0.7902	0.9770	0.7032	735.70	93.114	
			Std. Dev.	0.0207	0.0141	0.0273	0.0250	0.0346	1299.60	207.499	
	Features embedding + Distances: $\varphi(x_i), \{\mathcal{D}[\varphi(x_i), \varphi(L_j)]\}_{j=1..2}$ Generated with NMM (embedding dimension:8)	10	Logistic Regression	***0.8933***	***0.9086***	***0.8602***	***0.9628***	***0.7887***	***0.71***	***0.004***	6.77
		10	Random Forest	0.8577	0.8836	0.8041	0.9805	0.7284	4.23	***0.109***	
		10	GBM	***0.8696***	***0.8923***	***0.8180***	0.9814	***0.7500***	***26.36***	***0.088***	
		10	SVM-RBF	***0.8605***	***0.8863***	***0.8040***	0.9874	***0.7362***	***174.39***	***41.041***	
		10	MLP	***0.8724***	***0.8944***	***0.8214***	0.9818	***0.7553***	***50.35***	***0.239***	
			Average	***0.8707***	***0.8930***	***0.8215***	***0.9788***	***0.7517***	***51.21***	***8.296***	
			Std. Dev.	***0.0140***	***0.0097***	***0.0230***	***0.0094***	***0.0233***	***71.67***	***18.305***	
10 Labels	Original features: x_i	196	Logistic Regression	0.6362	0.6937	0.8254	0.6362	0.5657	246.67	0.109	N/A
		196	Random Forest	0.7037	0.7376	0.7899	0.7037	0.6145	***4.53***	0.407	
		196	GBM	0.7218	0.7296	0.7911	0.7218	0.6457	885.82	3.051	
		196	SVM-RBF	0.6605	0.6753	0.7844	0.6605	0.5920	4390.77	1410.223	
		196	MLP	0.7283	0.7446	0.8122	0.7283	0.6585	656.11	0.383	
			Average	0.6901	0.7162	0.8006	0.6901	0.6153	1236.78	282.835	
			Std. Dev.	0.0401	0.0301	0.0174	0.0401	0.0381	1796.30	630.230	
	Features embedding + Distances: $\varphi(x_i), \{\mathcal{D}[\varphi(x_i), \varphi(L_j)]\}_{j=1..10}$ Generated with NMM (embedding dimension:10)	20	Logistic Regression	***0.6942***	***0.7422***	***0.8424***	***0.6942***	***0.6272***	***95.05***	***0.015***	7.61
		20	Random Forest	***0.7270***	***0.7534***	***0.8050***	***0.7270***	***0.6494***	7.00	***0.171***	
		20	GBM	***0.7553***	***0.7586***	***0.8091***	***0.7553***	***0.6864***	***591.35***	***1.456***	
		20	SVM-RBF	***0.7571***	***0.7560***	***0.8203***	***0.7571***	***0.6900***	***952.37***	***205.703***	
		20	MLP	***0.7581***	***0.7602***	***0.8206***	***0.7581***	***0.6919***	***257.20***	***0.364***	
			Average	***0.7383***	***0.7541***	***0.8195***	***0.7383***	***0.6690***	***380.59***	***41.542***	
			Std. Dev.	***0.0279***	***0.0071***	***0.0145***	***0.0279***	***0.0292***	***389.77***	***91.771***	

Fig. 14. UNSW-NB15 dataset: Performance classification metrics when using the original features versus the transformed ones produced by the NMM model (RLB-CL/Type III). The comparison is done with a representative set of classic ML models used as a reference. Bold-italic is used to mark the best value in a pair-wise comparison between each metric using original and transformed features. All results are computed over the test set.

(Silhouette) obtains its maximum with contrastive models (ConLE) in both binary and multi-class scenarios. The LM+KA model presents poor prediction results for this dataset and the CNN-1D model (with the second best prediction results) has long training and prediction times.

4.4.2. Improvement of ML models

Following an analysis similar to that performed in Section 4.3.2 for NSL-KDD, Fig. 14 shows the improvement in classification performance when using the transformed features generated by the NMM model (used as a reference base model) instead of the original features. The comparison is done with a set of classic ML models used as a reference (logistic regression, random forest, GBM, SVM and MLP). To facilitate the comparison in Fig. 14, we have marked in bold-italic the best value in a pair-wise comparison between each metric using original and transformed features.

Considering F1 and MCC as the two metrics that best represent the overall performance of the models, we observe (Fig. 14) the improvement obtained when using the transformed features. This improvement is also achieved for all average performance metrics with a reduction in the standard deviation of values.

4.5. Detection of unknown intrusions

In this section we present a comparison of the capabilities of some of the models discussed in Sections 4.3 and 4.4 to detect unknown intrusions. We propose to assess the ability of a model to detect unknown intrusions with a methodology based on removing a particular intrusion from the training set and to evaluate the ability of the model to detect this intrusion as an anomaly in the test set. The details of the proposed methodology are based on the following steps:

1 Select some of the best performing and representative models for the two datasets.
2 Select the two most frequent attacks for the two datasets (*i.e.*, DOS and PROBE for NSL-KDD; and, Generic and Exploits for UNSW-NB15)
3 For each selected attack in each dataset:
   a Remove the samples corresponding to this attack in the training set. The resulting training set will have either 4 labels (NSL-KDD) or 9 labels (UNSW-NB15) since we have removed one of the attacks.
   b Keep the samples corresponding to this attack in the test set. The test set does not change.
   c Train the model with the new training set. Prior to training, we collapse all attacks into a single anomaly label. Therefore, we train the model with a binary classification scheme (normal/anomaly).
   d Predict results for the test set. The prediction will be a binary label (normal/anomaly). Using this prediction, we now obtain the detection rate (recall metric) for each original class in the test set *i.e.*, the percentage of samples detected as anomaly (with the binary classifier) among the samples of each original class.

As a particular example, assuming we select to remove the DOS attack from the NSL-KDD dataset. The steps mentioned above will be: a) We remove the DOS samples in the training set. b) We keep the test set unchanged. c) We train a binary classifier (normal/anomaly labels) with the resulting training set after collapsing all remaining attacks in the training set (PROBE, R2L and U2R) to a single anomaly label. d) We

				Global classification scores			Detection Rate for a specific class in the Test Set		
	Type	Classifier	Attacks in the Training Set	Accuracy	F1	MCC	NORMAL	DOS	PROBE
NSL-KDD	ML	Random Forest	All attacks	0.750	0.726	0.577	0.971	0.805	0.703
			DOS missing	0.693	0.640	0.500	0.976	0.586	0.611
			PROBE missing	0.738	0.705	0.569	0.986	0.789	0.372
	ML	CNN-1D	All attacks	0.806	0.799	0.633	0.917	0.861	0.928
			DOS missing	0.738	0.708	0.562	0.975	0.671	0.786
			PROBE missing	0.780	0.763	0.630	0.989	0.856	0.335
	LB-CL	ConCE	All attacks	0.905	0.918	0.806	0.873	0.967	0.939
			DOS missing	0.800	0.801	0.628	0.921	0.782	0.866
			PROBE missing	0.834	0.834	0.699	0.966	0.909	0.605
	RLB-CL	ENMS	All attacks	0.876	0.895	0.747	0.812	0.982	0.970
			DOS missing	0.804	0.808	0.630	0.910	0.744	0.944
			PROBE missing	0.801	0.811	0.614	0.872	0.894	0.815
	RLB-CL	CEDist+NMM	All attacks	0.837	0.845	0.684	0.907	0.920	0.868
			DOS missing	0.784	0.783	0.601	0.914	0.715	0.899
			PROBE missing	0.822	0.824	0.671	0.941	0.883	0.686

				Global classification scores			Detection Rate for a specific class in the Test Set		
	Type	Classifier	Attacks in the Training Set	Accuracy	F1	MCC	NORMAL	GENERIC	EXPLOITS
UNSW-NB15	ML	Random Forest	All attacks	0.862	0.887	0.738	0.740	0.999	0.989
			GENERIC missing	0.816	0.840	0.628	0.741	0.765	0.991
			EXPLOITS missing	0.859	0.880	0.719	0.754	0.998	0.901
	ML	CNN-1D	All attacks	0.898	0.913	0.813	0.849	0.999	0.987
			GENERIC missing	0.861	0.873	0.719	0.854	0.755	0.990
			EXPLOITS missing	0.894	0.910	0.792	0.804	0.999	0.972
	LB-CL	ConLE	All attacks	0.909	0.921	0.818	0.848	0.998	0.990
			GENERIC missing	0.878	0.885	0.757	0.906	0.813	0.966
			EXPLOITS missing	0.880	0.891	0.759	0.876	0.995	0.862
	RLB-CL	ENMS	All attacks	0.837	0.870	0.697	0.647	1.000	0.999
			GENERIC missing	0.835	0.868	0.690	0.648	0.992	0.998
			EXPLOITS missing	0.836	0.868	0.688	0.662	1.000	0.985
	RLB-CL	CEDist+NMM	All attacks	0.899	0.908	0.798	0.902	1.000	0.990
			GENERIC missing	0.899	0.907	0.796	0.897	0.993	0.963
			EXPLOITS missing	0.862	0.877	0.721	0.820	0.997	0.865

Fig. 15. Detection rates for specific subsets of attacks in the test set for the datasets: NSL-KDD (**Upper chart**) and UNSW-NB15 (**Lower chart**). Detection rates are obtained separately when all samples in the training set are used and when samples corresponding to selected attacks are removed. Different types of classifiers are used in a binary classification scheme to predict each sample in the test set as normal/anomaly. The detection rate for a class in the test set is the percentage of samples correctly identified as normal/anomaly for that class. A color code is used where a dark-green is associated with better results and a dark-red with worse results, with an interpolated color-palette that is used for intermediate values. The color code is applied column-wise and independently (separately) for the results of each model (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.).

predict the normal/anomaly label for each sample in the (unchanged) test set. Then, we select the subset of samples from the test set that had the DOS attack as their original label (ground-truth label). Using this subset, we obtain the percentage of correct detections as anomaly (using the binary classifier) for all elements in this subset. This percentage of correct detections is also known as detection rate or recall. Likewise, we create similar subsets of the test set for the samples associated with the PROBE and NORMAL labels (ground-truth). For these additional subsets we also obtain their corresponding detection rates. The reason for including the detection rate for NORMAL samples (even though the NORMAL samples are never removed from the training set) is to check how this metric changes by altering the distribution of attacks in the training set. At the end, we present the detection rates for the three subsets of samples (for the PROBE, DOS, and NORMAL labels) for each attack elimination exercise. We also provide the global binary classification scores obtained with the binary classifier.

The methodology mentioned above aims to evaluate the ability of the models to classify a sample as carrying an attack even when its associated training samples have been removed from the training set. To perform the experiments, it is necessary to use different levels of granularity in the hierarchy of security attacks (*e.g.*, attack $->$ DOS $->$ Neptune) since otherwise a multi-class classifier trained without the knowledge of a particular class cannot produce a classification for that class. However, by playing at different levels of granularity, we can predict a sample as an anomaly, since the anomaly class has been used at training time.

Following this methodology, we have obtained expected results, such as having the smallest detection rate for each attack in the test set when the corresponding attack is missing in the training set. However, it is interesting to note that the proposed models behave better than

		Type	Classifier	Embed. Dim.	Nº of training samples	Classification scores: Accuracy	F1	MCC
NSL-KDD	2 labels	LB-CL	ConCE	2	100	0.7178	0.6756	0.5406
					1000	0.8029	0.8040	0.6346
					10000	0.8117	0.8152	0.6471
					50000	0.8345	0.8422	0.6818
					250000	0.9050	0.9176	0.8058
		RLB-CL	ENMS	2	100	0.7546	0.7443	0.5582
					1000	0.7724	0.7667	0.5868
					10000	0.7798	0.7756	0.5988
					50000	0.7751	0.7687	0.5939
					125000	0.8763	0.8949	0.7474
	5 labels	LB-CL	ConCE	2	100	0.5090	0.4214	0.2087
					1000	0.6773	0.6007	0.5024
					10000	0.7512	0.7027	0.6314
					50000	0.7609	0.7117	0.6463
					625000	0.7597	0.7279	0.6416
		RLB-CL	ENMS	2	100	0.6689	0.6149	0.4921
					1000	0.7488	0.7080	0.6203
					10000	0.7777	0.7394	0.6736
					50000	0.7780	0.7388	0.6772
					125000	0.7906	0.7530	0.6955
UNSW-NB15	2 labels	LB-CL	ConLE	2	100	0.7512	0.8150	0.5505
					1000	0.8502	0.8771	0.7114
					10000	0.8499	0.8775	0.7128
					100000	0.8625	0.8867	0.7359
					350000	0.9089	0.9206	0.8178
		RLB-CL	NMM	8	100	0.7812	0.8241	0.5703
					1000	0.8058	0.8493	0.6442
					10000	0.8164	0.8560	0.6608
					100000	0.8164	0.8564	0.6624
					170000	0.8163	0.8565	0.6630
	10 labels	LB-CL	ConLE	10	100	0.2264	0.1511	0.1829
					1000	0.6495	0.6419	0.5208
					10000	0.6728	0.6948	0.6080
					100000	0.7397	0.7468	0.6633
					1700000	0.7143	0.7472	0.6527
		RLB-CL	NMM	10	100	0.6111	0.6111	0.4766
					1000	0.6139	0.6232	0.4812
					10000	0.6734	0.6720	0.5463
					100000	0.6828	0.6795	0.5587
					170000	0.7444	0.7669	0.6791

Fig. 16. Main classification metrics for the two datasets and some representative models, when the number of training samples is reduced by several orders of magnitude (from the complete training set to 100 training samples). Training set reduction is done by random sampling, maintaining the proportions of the labels as far as possible (stratified sampling).

alternative ML models (Fig. 15), showing in general better detection rates for the missing attacks. It is also interesting that these detection rates are quite high in many cases (*e.g.*, a detection rate of 0.992 for the Generic attack in UNSW-NB15, with the ENMS model) considering the absence of these attacks in the training set.

In Fig. 15 we can see that the best global classification scores are obtained with the complete training set (all attacks present), being the worst when we remove the most frequent attack (DOS and Generic, respectively). As mentioned above, the smallest detection rate is obtained when the corresponding attack is missing from the training set. For the NORMAL samples, the detection rate tends to increase when one of the attacks is missing, which could be understood as having less difficulty discriminating the attacks.

4.6. Behavior with small datasets

Considering the particularities of the proposed models, based on optimizing the number of comparisons required to implement the contrastive mechanism, it is interesting to evaluate the behavior of these models with a reduced number of training samples. In this Section we present the classification results for the most representative models, for both datasets, with a reduction of several orders of magnitude in the number of training samples, while keeping the test sets unchanged (for both datasets). For each dataset and model, the results are presented for the complete dataset and for several reduced versions of the dataset (Fig. 16). Training sets reduction is done by stratified sampling (random sampling keeping the proportions of the labels as far as possible). In its smallest version, the training sets have been reduced to just 100 samples. In general, as expected, the results improve monotonically with the number of training samples, but in general the results are extremely good with a number of samples above 1000–10000 (Fig. 17).

Fig. 17 presents in two graphs the evolution of the F1-score for the two datasets and for several of the proposed models. The horizontal-axis of the graphs has a logarithmic scale, since we have obtained the results by reducing the number of samples by several orders of magnitude. The rightmost points in these graphs (Fig. 17) represent the nominal number

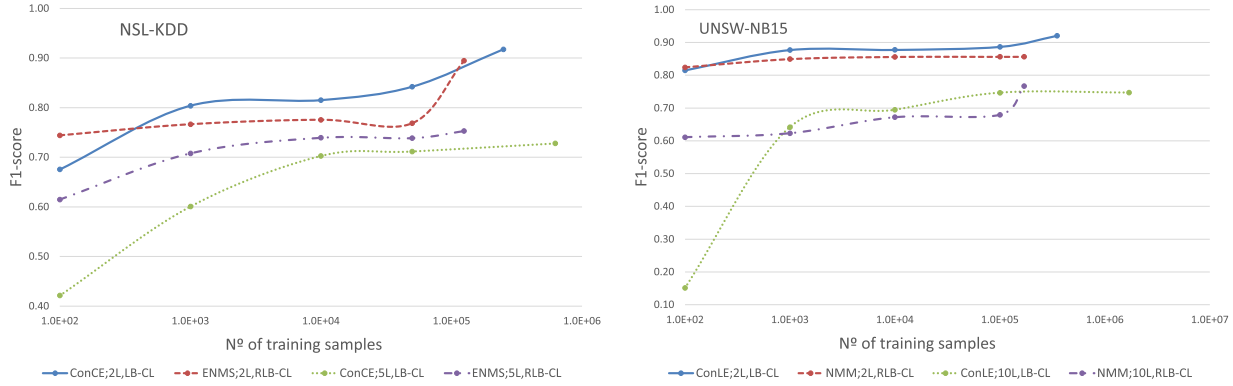

Fig. 17. Evolution of the F1-score for several models and the two datasets, as the number of training samples increases (horizontal-axis in logarithmic scale). Each model is identified by the model name, the configuration by number of labels used in the dataset (*e.g.*, 5L for the 5-labels configuration), and the model category (*i.e.*, LB-CL or RLB-CL). The graphs are dataset dependent: (**Left chart**) NSL-KDD and (**Right chart**) UNSW-NB15.

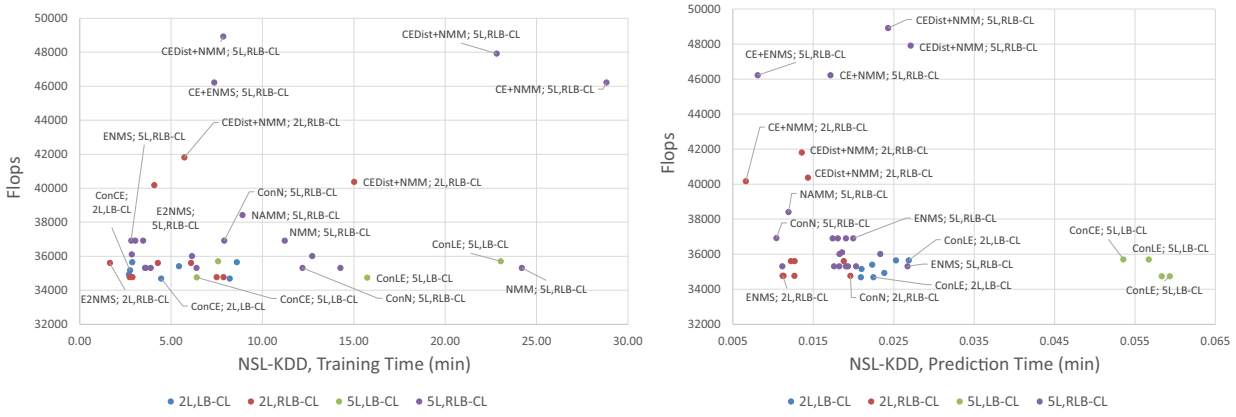

Fig. 18. Comparison of the proposed models in terms of Flops vs. Training Time (**Left chart**) and Flops vs. Prediction Time (**Right chart**) for the NSL-KDD dataset. Each model is identified by the model name, the configuration by number of labels used in the dataset (*e.g.*, 5L for the 5-labels configuration), and the model category (*i.e.*, LB-CL or RLB-CL).

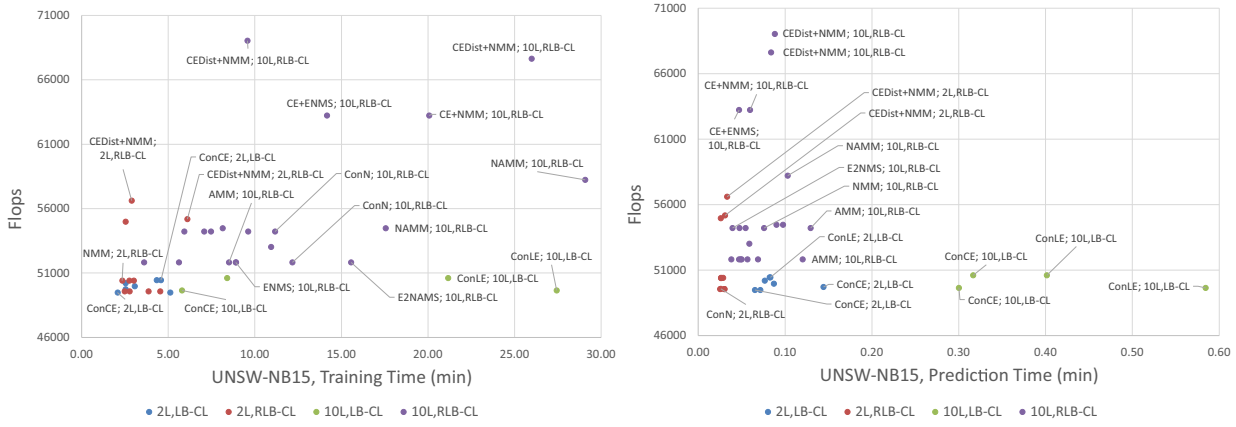

Fig. 19. Comparison of the proposed models in terms of Flops vs. Training Time (**Left chart**) and Flops vs. Prediction Time (**Right chart**) for the UNSW-NB15 dataset. Each model is identified by the model name, the configuration by number of labels used in the dataset (*e.g.*, 10L for the 10-labels configuration), and the model category (*i.e.*, LB-CL or RLB-CL).

of training samples (complete training sets). We can observe how this number varies widely between models. The LB-CL models require a significantly higher number of training samples, as these models need to compare each sample with all available labels. While the RLB-CL models only require comparing each sample with the positive label and a single representative of the negative ones (two comparisons in total). As the number of classes/labels increases, this difference becomes more important.

It is very interesting how the LB-CL models when applied to multi-class classification rapidly deteriorate below a certain number of

training samples. This deterioration is much stronger than for the RLB-CL models and occurs earlier in the graph, that is, the LB-CL models, when doing multi-class classification, require many more training samples than the alternative RLB-CL models. Collapsing all negative labels into their best representative avoids the need to make a comparison between the sample and all negative labels (number that increases with the number of labels) and this more effective representation also makes the training process more efficient.

4.7. Computational complexity comparison

Computational complexity information, for the proposed models, is provided in this Section and in Figs.11 and 13. The number of trainable weights and number of Floating Point Operations (Flops) required for each model are given in detail in Figs. 11 and 13. These figures also contain the training and prediction times per model. The number of Flops is calculated for the complete computational graph [52] for each model. This Section shows an overview comparison between the different models considering the number of Flops in relation to the training and prediction times (for the two datasets). Figs. 18 and 19 provide these comparison with a separate graph for Flops vs. Training Time and Flops vs. Prediction Time. Points in the graphs represent particular models. For reasons of space, not all models are identified, only the most representative ones.

The location of the models on the charts (Figs. 18 and 19) follows a noisy pattern for the charts that include training times. Conversely, we find very stable patterns in the charts that consider prediction times (even for different data sets).Training times are noisy due to the early-stopping criteria used during training, which is stochastic in nature (Section 3.5). The charts of Flops vs. Prediction Time have a well-established pattern for the distribution of the different models:

- High prediction times with a low Flops number is for LB-CL models with the multi-label configuration.
- Large Flops number with low prediction times is for the models based on cross-entropy with contrastive regularization (RLB-CL) with the multi-label configuration (*e.g.*, CE+NMM, CEDist+NMM).
- Low Flops number with low prediction times is for LB-CL models with binary-labels and most RLB-CL models with binary and multi-label configurations. It is worth mentioning the particularly low Flops number and prediction times, with the multi-label configuration, for the following models: NMM, AMM, ConN, ENMS and E2NMS.
- It is interesting that there is no model with both high prediction times and large Flops number, which means that the proposed models compensate a complex architecture with a faster operation (fewer comparisons required).

The number of trainable weights perfectly correlates with the number of Flops needed for each model. The number of Flops is approximately twice the number of trainable weights, as expected considering the well-established relationship between Flops and matrix/vector dimensions [53]. It is interesting that the main contribution to the Flops required by each model is determined mainly by its number of weights and much less, comparatively, by the operations necessary to calculate the loss function.

The LB-CL models, for multi-class classification, show extremely low Flops requirements but the highest prediction times. This is due to its architecture, which at the time of prediction requires comparing each sample with each of the labels to find the one closest to the sample; these multiple comparisons, at the time of prediction, strongly penalize them.

4.8. Generic results

It is interesting how the increase in the dimensionality of the embeddings does not necessarily produce an increase in performance, being the opposite in many cases. It is also interesting that consistently good performances are obtained with very low dimensionality embeddings.

With a reduced number of classes (labels), the LB-CL models have the best classification and clustering performance, but, when the number of classes increases, the RLB-CL models provide the best results. This is the expected behavior because, as we replace the negative labels with their best individual representative, we make the label separation process more effective. The MMoLE architecture provides the worst performance, and its results are not shown to concentrate the presentation area to the best models. The idea behind providing the different loss functions (Sections 3.3 and 3.4) was to exploit the taxonomy given in Fig. 1 and Table 2, trying to create models that implemented all the different types of separation strategies (columns in Fig. 1). That is the reason for creating the proposed loss functions as each one fits into different separation strategies. The original aim was to check whether the separation strategy had a significant impact on performance, which appears not to be the case, as the best results are fairly evenly distributed among the different separation strategies. What makes a clear difference is to use LB-CL or RLB-CL methods, with a great impact depending on whether we are doing binary or multiclass classification; with the LB-CL models

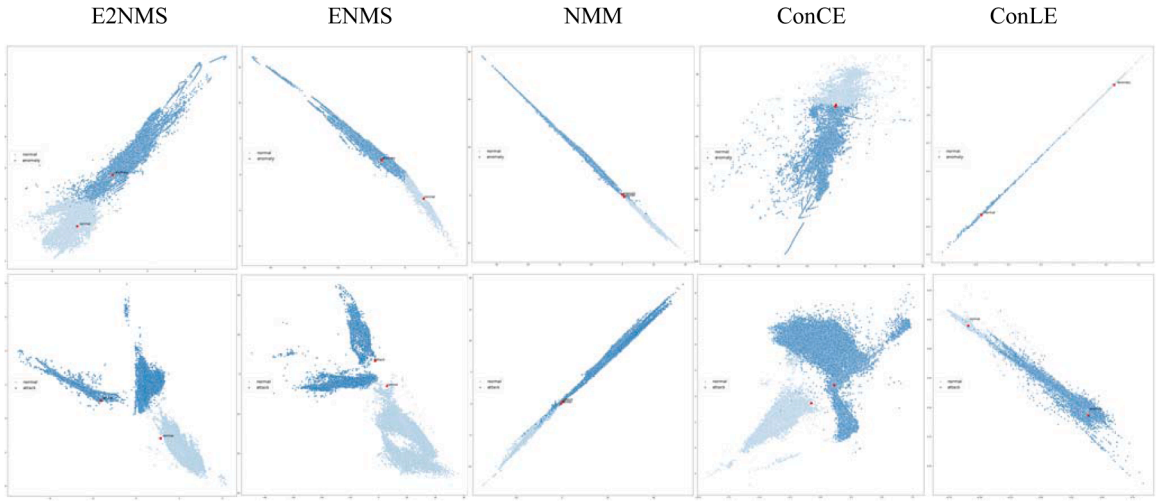

Fig. 20. Binary clusters: Shapes adopted by the clusters formed by the samples in embedding space around their corresponding label embeddings (points in red-bold). The clusters correspond to the binary-class scenarios for NSL-KDD (**Upper row**) and UNSW-NB15 (**Lower row**). The columns correspond to different LB-CL/RLB/CL models used to create the embeddings. All models use a 2-dimensional embedding space.

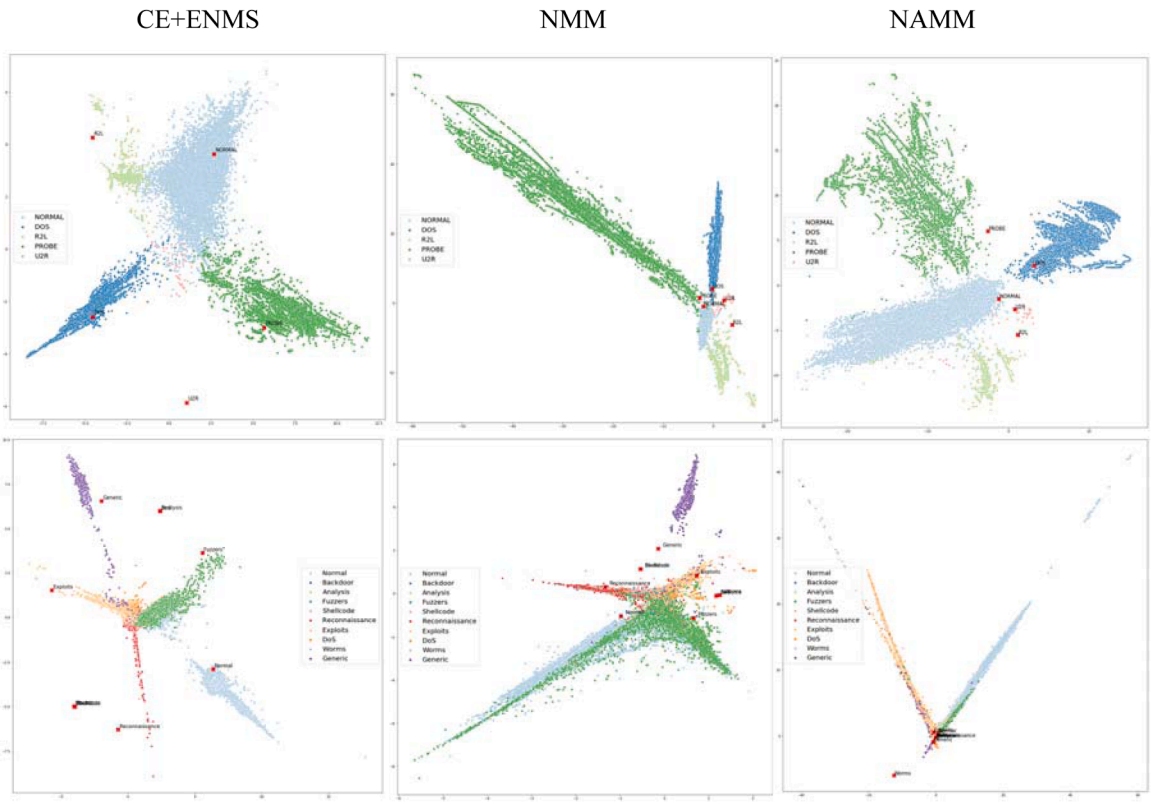

Fig. 21. Multi-class clusters: Shapes adopted by the clusters formed by the samples in embedding space around their corresponding label embeddings (points in red-bold). The clusters correspond to the multi-class scenarios for NSL-KDD (**Upper row**) with 5 labels and UNSW-NB15 (**Lower row**) with 10 labels. The columns correspond to different LB-CL/RLB/CL models used to create the embeddings. All models use a 2-dimensional embedding space.

excelling in binary classification and the RLB-CL models in multiclass classification.

Considering the results presented in the above sections, we can conclude that the proposed models offer a competitive and alternative framework to build classifiers for noisy and unbalanced datasets. The reason for this can be found in the ability of these models to not only bring the samples closer to their positive label, but also separate them from their negative label representatives. That means that, for each training iteration, all negative labels are considered at all times. Among the negative labels there is a high probability of presence for the infrequent ones, which implies that for each training iteration we will be acting not only on the most frequent labels (which will appear more often as a positive label), but also on the less frequent ones (which will necessarily appear more often as part of the negative label sets). Robustness to noise can also be explained by this constant iteration between each sample and the full set of labels. This constant iteration between a reduced set of label representatives with all samples provides an averaging effect that limits the influence of labeling errors or sample features noise.

While providing best classification performance, these models also offer the possibility to employ the generated embeddings (in a low dimensional space) to enhance other models when these embeddings are used as a replacement for the original features. The embeddings can also be used to create visual representation of the distribution of labels and samples in embedding space. Considering that the models performance is very remarkable even with 2-dimensional embeddings, we can use them to perform visualizations without any additional dimensionality reduction (*e.g.*, PCA, t-SNE). Figs. 20 and 21 provide some examples of the clusters formed by the samples in embedding space around their corresponding label embeddings. These visualizations could be used to facilitate the location in embedding space of a new sample and identify outliers. We can also observe the different distributions of samples around their labels depending on the model; with the exponential and contrastive losses giving a more clustered distribution around the label and the NMM a star-shaped distribution of samples around the labels which occupy a central position in the graph.

All neural network models were implemented in python with Tensorflow/Keras [52]. To obtain the performance metrics we have used the scikit-learn python package [54]. All computations were performed in a commercial PC (i7-4720-HQ, 16 GB RAM).

5. Conclusion

This work presents a set of novel architectures and loss functions applied to supervised contrastive learning, with the following characteristics:

- Map sample features and labels to the same representation space.
- Create a type of classifiers based on ranking distances between sample features and their labels, such that same-class samples will gather around their label prototype in a common representation space (embedding). We demonstrate that these classifiers outperform other leading alternative algorithms in a variety of scenarios.
- Create an implicit dimensionality reduction technique by adopting the sample embeddings and their distances to the label embeddings as a substitute to the original features. We demonstrate that these

low-dimensionality features increase the prediction performance of a wide range of machine learning algorithms.
- Significatively reduce the number of pair-wise comparisons required by alternative contrastive learning methods (sample-wise methods) by ranking each sample against its positive/negative labels, rather than its positive/negative samples, which is a much larger set.
- Introduce a way to further reduce the required comparisons by replacing all negative labels with a single proxy-label to represent them.
- Provide excellent prediction results with a very low dimensionality of the embedding space, even with a dimension as low as 2.

We present a taxonomy of contrastive learning solutions to compare the proposed methods with those existing in the literature. This taxonomy is articulated using these differentiation parameters: a) the types of embeddings used, b) the separation strategy, and c) labels availability (whether the models are either supervised or unsupervised/self-supervised). Following this taxonomy, the proposed models are grouped into two categories according to whether all labels are used (LB-CL) or only two labels are used (positive label and a proxy-label representative of all negative labels) (RLB-CL). Several models and architectures are proposed for each of these categories, analyzing their properties and behavior under different conditions: small datasets, detection of unknown labels and computational complexity.

The experiments performed show that the proposed models are consistently among the best in all experiments, being competitive in prediction performance and low execution times. We show that the proposed models can be particularly useful for noisy and unbalanced multi-class classification problems by applying them to two well-known intrusion detection datasets: NSL-KDD and UNSW-NB15.

We believe that the line of research presented here could yield fruitful results in various fields. For future research, we intend to explore further developments of the proposed framework in the following directions: combine it with knowledge distillation for contrastive learning [55], create more sophisticated weighted losses and extend it to structured outputs [56].

CRediT authorship contribution statement

Manuel Lopez-Martin: Investigation, Formal analysis, Software, Writing – original draft. **Antonio Sanchez-Esguevillas:** Resources, Writing – review & editing. **Juan Ignacio Arribas:** Funding acquisition, Validation, Writing – review & editing. **Belen Carro:** Funding acquisition, Supervision, Writing – review & editing.

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.

Acknowledgment

The preparation of the article and the study of algorithms were funded with Grant RTI2018-098958-B-I00 from Proyectos de I+D+i «Retos investigación», Programa Estatal de I+D+i Orientada a los Retos de la Sociedad, Plan Estatal de Investigación Científica, Técnica y de Innovación 2017-2020. Spanish Ministry for Science, Innovation and Universities; the Agencia Estatal de Investigación (AEI) and the Fondo Europeo de Desarrollo Regional (FEDER).

Appendix. A

We will show the relationship between (a) the distance to a cluster's centroid and (b) the average distance to all elements in the cluster. We refer to the reference vector to which we want to compute the distance as A. $\overline{X}$ is the centroid of N vectors X_i *i.e.* $\overline{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$. We represent the distance between vectors A and B as $\| A - B \|$ which is the norm of the difference of the vectors, where $\| A \| = A^T.A$

Distance to centroid:

$\| \overline{X} - A \| = (\overline{X} - A)^T.(\overline{X} - A) = \overline{X}^T.\overline{X} - 2\overline{X}^T A + A^T.A = \| \overline{X} \| - 2\overline{X}^T A + \| A \| = \| \overline{X} \| - 2A^T\overline{X} + \| A \|$ since $\overline{X}^T A$ is a scalar.

Therefore, $\| \overline{X} - A \| = \| \overline{X} \| - 2A^T\overline{X} + \| A \| = \frac{1}{N}\| \sum_{i=1}^{N} X_i \| - 2A^T\overline{X} + \| A \|$

Average distance to the N elements of the cluster:

$$\frac{1}{N}\sum_{i=1}^{N} \| X_i - A \| = \frac{1}{N} \sum (X_i - A)^T.(X_i - A) = \frac{1}{N} \sum {X_i}^T.X_i - 2A^T\left(\frac{1}{N}\sum X_i\right) + \frac{1}{N}\sum A^T.A = \frac{1}{N}\sum \| X_i \| - 2A^T\overline{X} + \| A \|$$

By the triangle inequality: $\sum_{i=1}^{N} \| X_i \| \geq \| \sum_{i=1}^{N} X_i \|$, therefore:

$$\frac{1}{N}\sum_{i=1}^{N} \| X_i - A \| \geq \frac{1}{N} \| \sum_{i=1}^{N} X_i \| - 2A^T\overline{X} + \| A \| = \| \overline{X} - A \|$$

That means, that the average distance to the points of the cluster is an upper bound for the distance to the centroid of the cluster, and a minimization of that distance will also imply a minimization of the distance to the centroid.

References

[1] P.H. Le-Khac, G. Healy, A.F. Smeaton, Contrastive Representation Learning: A Framework and Review, IEEE Access, 2020, https://doi.org/10.1109/access.2020.3031549.
[2] A. Jaiswal, A.R. Babu, M.Z. Zadeh, D. Banerjee, F. Makedon, A survey on contrastive self-supervised learning, (2020). http://arxiv.org/abs/2011.00362 (accessed January 23, 2021).
[3] S. Arora, H. Khandeparkar, M. Khodak, O. Plevrakis, N. Saunshi, A theoretical analysis of contrastive unsupervised representation learning, (2019). http://arxiv.org/abs/1902.09229 (accessed January 22, 2021).
[4] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, D. Krishnan, Supervised contrastive learning, (2020). http://arxiv.org/abs/2004.11362 (accessed December 14, 2020).
[5] K. Sohn, Improved deep metric learning with multi-class n-pair loss objective, in: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (*NIPS'16*), Curran Associates, Inc., Red Hook, NY, USA, 2016, pp. 1857–1865, in: https://proceedings.neurips.cc/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf.
[6] A. van den Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, (2018). http://arxiv.org/abs/1807.03748 (accessed January 23, 2021).
[7] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: a unified embedding for face recognition and clustering, in: Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823, https://doi.org/10.1109/CVPR.2015.7298682.
[8] R. Hadsell, S. Chopra, Y. LeCun, Dimensionality reduction by learning an invariant mapping, in: Proceedings of the 2006 IEEE Computer Society Conference on

Computer Vision and Pattern Recognition, 2006, pp. 1735–1742, https://doi.org/10.1109/CVPR.2006.100.
[9] H.O. Song, Y. Xiang, S. Jegelka, S. Savarese, Deep metric learning via lifted structured feature embedding, (2015). http://arxiv.org/abs/1511.06452 (accessed February 16, 2021).
[10] Y. Movshovitz-Attias, A. Toshev, T.K. Leung, S. Ioffe, S. Singh, No fuss distance metric learning using proxies, (2017). https://arxiv.org/abs/1703.07464 (accessed February 16, 2021).
[11] Z. Akata, F. Perronnin, Z. Harchaoui, C. Schmid, Label-embedding for image classification, IEEE Trans. Pattern Anal. Mach. Intell. 38 (2015) 1425–1438, https://doi.org/10.1109/TPAMI.2015.2487986.
[12] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, L. Carin, Joint embedding of words and labels for text classification, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, Long Pap., Association for Computational Linguistics, 2018, pp. 2321–2331, https://doi.org/10.18653/v1/P18-1216. Volume 1.
[13] X. Wang, Y. Hua, E. Kodirov, G. Hu, R. Garnier, N.M. Robertson, Ranked list loss for deep metric learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 5202–5211, https://doi.org/10.1109/CVPR.2019.00535.
[14] S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, pp. 539–546, https://doi.org/10.1109/CVPR.2005.202, vol. 1.
[15] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, (2020). https://arxiv.org/abs/2002.05709 (accessed January 22, 2021).
[16] J. Goldberger, S. Roweis, G.E. Hinton, R. Salakhutdinov, Neighbourhood Components Analysis, NIPS, 2004.
[17] K.A.P. da Costa, J.P. Papa, C.O. Lisboa, R. Munoz, V.H.C. de Albuquerque, Internet of things: a survey on machine learning-based intrusion detection approaches, Comput. Netw. 151 (2019) 147–157, https://doi.org/10.1016/j.comnet.2019.01.023.
[18] S. Gamage, J. Samarabandu, Deep learning methods in network intrusion detection: a survey and an objective comparison, J. Netw. Comput. Appl. 169 (2020), 102767, https://doi.org/10.1016/j.jnca.2020.102767.
[19] S. Mahdavifar, A.A. Ghorbani, Application of deep learning to cybersecurity: a survey, Neurocomputing 347 (2019) 149–176, https://doi.org/10.1016/j.neucom.2019.02.056.
[20] H. Liu, B. Lang, Machine learning and deep learning methods for intrusion detection systems: a survey, Appl. Sci. 9 (2019), https://doi.org/10.3390/app9204396.
[21] D. Berman, A. Buczak, J. Chavis, C. Corbett, A survey of deep learning methods for cyber security, Information 10 (2019) 122, https://doi.org/10.3390/info10040122.
[22] A. Aldweesh, A. Derhab, A.Z. Emam, Deep learning approaches for anomaly-based intrusion detection systems: a survey, taxonomy, and open issues, Knowl. Based Syst. 189 (2020), 105124, https://doi.org/10.1016/j.knosys.2019.105124.
[23] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, J. Mach. Learn. Res. 10 (2009) 207–244.
[24] B. Ghojogh, F. Karray, M. Crowley, Fisher and kernel fisher discriminant analysis: tutorial, (2019). http://arxiv.org/abs/1906.09436 (accessed February 24, 2021).
[25] J. Snell, K. Swersky, R.S. Zemel, Prototypical networks for few-shot learning, (2017). https://arxiv.org/abs/1703.05175 (accessed March 5, 2021).
[26] C.Y. Chuang, J. Robinson, L. Yen-Chen, A. Torralba, S. Jegelka, Debiased contrastive learning, (2020). https://arxiv.org/abs/2007.00224 (accessed December 14, 2020).
[27] J. Li, P. Zhou, C. Xiong, R. Socher, S.C.H. Hoi, Prototypical contrastive learning of unsupervised representations, ArXiv. (2020). http://arxiv.org/abs/2005.04966 (accessed January 22, 2021).
[28] M. Kim, J. Tack, S.J. Hwang, Adversarial self-supervised contrastive learning, (2020). https://arxiv.org/abs/2006.07589 (accessed December 14, 2020).
[29] H. Hindy, C. Tachtatzis, R. Atkinson, D. Brosset, M. Bures, I. Andonovic, C. Michie, X. Bellekens, Leveraging Siamese networks for one-shot intrusion detection model, (2020). http://arxiv.org/abs/2006.15343 (accessed January 23, 2021).
[30] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot, Sensors 17 (2017), https://doi.org/10.3390/s17091967 (Switzerland).
[31] N. Pappas, J. Henderson, GILE: a generalized input-label embedding for text classification, (2018). http://arxiv.org/abs/1806.06219 (accessed December 14, 2020).
[32] H. Zhang, L. Xiao, W. Chen, Y. Wang, Y. Jin, Multi-task label embedding for text classification, (2017). https://arxiv.org/abs/1710.07210 (accessed January 23, 2021).
[33] C. Chen, H. Wang, W. Liu, X. Zhao, T. Hu, G. Chen, Two-stage label embedding via neural factorization machine for multi-label classification, in: Proceedings of the AAAI Conference on Artificial Intelligence. 33(01), AAAI Press, 2019, pp. 3304–3311, https://doi.org/10.1609/aaai.v33i01.33013304.
[34] J. Tack, S. Mo, J. Jeong, J. Shin, CSI: Novelty detection via contrastive learning on distributionally shifted instances, (2020). http://arxiv.org/abs/2007.08176 (accessed February 27, 2021).
[35] S.F. Yilmaz, S.S. Kozat, Unsupervised anomaly detection via deep metric learning with end-to-end optimization, (2020). https://arxiv.org/abs/2005.05865 (accessed February 3, 2021).
[36] P. Bedi, N. Gupta, V. Jindal, Siam-IDS: handling class imbalance problem in intrusion detection systems using Siamese neural network, Proced. Comput. Sci. 171 (2020) 780–789, https://doi.org/10.1016/j.procs.2020.04.085.
[37] H. Jmila, M. Ibn Khedher, G. Blanc, M.A. El Yacoubi. Siamese network based feature learning for improved intrusion detection, Springer International Publishing, Cham, 2019, pp. 377–389, https://doi.org/10.1007/978-3-030-36708-4_31.
[38] S. Moustakidis, P. Karlsson, A novel feature extraction methodology using Siamese convolutional neural networks for intrusion detection, Cybersecurity 3 (2020) 16, https://doi.org/10.1186/s42400-020-00056-4.
[39] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, (2013). https://arxiv.org/abs/1310.4546 (accessed March 2, 2021).
[40] Y. Gao, N. Fei, G. Liu, Z. Lu, T. Xiang, S. Huang, Contrastive prototype learning with augmented embeddings for few-shot learning, (2021). https://arxiv.org/abs/2101.09499 (accessed February 3, 2021).
[41] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, ArXiv. (2014). https://arxiv.org/abs/1412.6980 (accessed January 15, 2021).
[42] M. Lopez-Martin, Mlopezm/Supervised-contrastive-learning-over-prototype-label-embeddings: code for the paper: “supervised contrastive learning over prototype-label embeddings for network intrusion detection,” (n.d.). https://github.com/mlopezm/Supervised-contrastive-learning-over-prototype-label-embeddings (accessed June 2, 2021).
[43] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: Proceedings of the Second IEEE International Conference Computer Intelligence Security Definition Application, IEEE Press, 2009, pp. 53–58.
[44] N. Moustafa, J. Slay, UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), in: Proceedings of the 2015 Military Communications and Information Systems Conference, 2015, pp. 1–6, https://doi.org/10.1109/MilCIS.2015.7348942.
[45] C.D. Manning, P. Raghavan, H. Schutze, Introduction to Information Retrieval, Cambridge University Press, 2008, https://doi.org/10.1017/cbo9780511809071.
[46] D.M.W. Powers, Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation, (2020). https://arxiv.org/abs/2010.16061 (accessed March 10, 2021).
[47] J.O. Palacio-Niño, F. Berzal, Evaluation metrics for unsupervised learning algorithms, (2019). http://arxiv.org/abs/1905.05667 (accessed March 11, 2021).
[48] M. Steinbach, L. Ertöz, V. Kumar, The challenges of clustering high dimensional data. New Directions in Statistical Physics, Springer Berlin, 2004, pp. 273–309, https://doi.org/10.1007/978-3-662-08968-2_16.
[49] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, Shallow neural network with kernel approximation for prediction problems in highly demanding data networks, Expert Syst. Appl. 124 (2019) 196–208, https://doi.org/10.1016/j.eswa.2019.01.063.
[50] A.M. Aleesa, B.B. Zaidan, A.A. Zaidan, N.M. Sahar, Review of intrusion detection systems based on deep learning techniques: coherent taxonomy, challenges, motivations, recommendations, substantial analysis and future directions, Neural Comput. Appl. 32 (2020) 9827–9858, https://doi.org/10.1007/s00521-019-04557-3.
[51] D.H. Wolpert, Stacked generalization, Neural Netw. 5 (1992) 241–259, https://doi.org/10.1016/S0893-6080(05)80023-1.
[52] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, Tensorflow: large-scale machine learning on heterogeneous distributed systems, ArXiv. (2016). https://arxiv.org/abs/1603.04467 (accessed September 17, 2020).
[53] R. Tibshirani, Flops for basic operations. Convex Optimization, Carnegie Mellon University, 2015. http://www.stat.cmu.edu/~ryantibs/convexopt-F18/scribes/Lecture_19.pdf (accessed June 7, 2021).
[54] F. Pedregosa, V. Michel, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, J. Vanderplas, D. Cournapeau, F. Pedregosa, G. Varoquaux, A. Gramfort, B. Thirion, O. Grisel, V. Dubourg, A. Passos, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830. http://scikit-learn.sourceforge.net (accessed September 17, 2020).
[55] Y. Tian, D. Krishnan, P. Isola, Contrastive representation distillation, (2019). https://arxiv.org/abs/1910.10699 (accessed March 5, 2021).
[56] S. Belharbi, R. Hérault, C. Chatelain, S. Adam, Deep neural networks regularization for structured output prediction, Neurocomputing 281 (2018) 169–177, https://doi.org/10.1016/j.neucom.2017.12.002.