Dealing with multipositive unlabeled learning combining metric learning and deep clustering

Amedeo Racanati, Roberto Esposito, Dino Ienco

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Dealing with Multi-Positive Unlabelled learning combining metric learning and deep clustering

AMEDEO RACANATI[1], ROBERTO ESPOSITO[1], and DINO IENCO[2], (Member, IEEE)
[1]University of Turin, Computer Science Department, C.so Svizzera, 185, I-10149 Torino, Italy (e-mail: amedeoracanati@gmail.com, roberto.esposito@unito.it)
[2]INRAE, UMR TETIS, University of Montpellier, 500, Rue Jean François Breton, F-34090 Montpellier, France (e-mail: dino.ienco@inrae.fr)

Corresponding author: Dino Ienco (e-mail: dino.ienco@inrae.fr)

This work was supported in part by the HPC4AI project funded by the Region Piedmont POR-FESR 2014-20 programme (INFRA-P) [1].

ABSTRACT Standard supervised classification methods make the assumption that the training data is fully annotated thus requiring an a-priory labelling process which is both costly and time-consuming. To relax this requirement, many different flavors of weakly supervised learning have been proposed. Among weakly supervised learning strategies, Positive Unlabelled learning (PUL) is gaining attention from the research community due to the wide spectrum of applications it can fit. However, the majority of research studies related to PUL only consider binary classification tasks while real-world applications commonly involve multiple categories. To deal with this limitation, Multi-Positive Unlabelled learning (MPUL) has been recently introduced to learn from examples labelled with multiple positive labels and a single unknown negative label. Up to today, only a limited number of research works were proposed to cope with this more general setting.

In this paper, we propose a new MPUL framework based on deep learning strategies. Our framework, named `ProtoMPUL` (Prototype based Multi-Positive and Unlabelled Learning), combines metric learning and clustering strategies to model the set of positive classes as well as to characterize the unknown negative one.

Experimental evaluations on real-world benchmarks considering recent MPUL competitors demonstrates that the proposed framework achieves state-of-the-art performances, thus supporting the validity of the proposed approach.

INDEX TERMS Multi-Positive Unlabelled learning, weakly supervised learning, tabular data, metric learning, deep clustering

I. INTRODUCTION

Standard supervised classification methods make the assumption that the training data is fully annotated with the whole set of classes of interest, requiring an apriori money- and time-consuming labelling process that can be unaffordable and unrealistic in several real-world scenarios. To relax this strict requirement several weakly supervised learning settings [2] have been proposed. Among them, it is worth mentioning active learning [3], semi-supervised learning [4], multi-instance learning [5], learning with label noise [6] and positive unlabelled learning (PUL) [7].

In recent years, Positive-Unlabelled learning received growing attention from the research community. PUL objective is to learn a classifier considering an *incomplete* training set where only a portion of the positive samples have associated label information while no label is available for samples belonging to the negative class [8]. In other terms, under such a learning setting, the training set is composed of two parts, a labelled one containing only positive samples and an unlabelled one containing both positive and negative samples. This learning setting is of particular importance in many real-world applications [8].

As a practical example, let us consider an automatic diagnosis scenario where a system aims to predict if a patient has a particular disease. In this scenario, patients diagnosed with the disease are labelled as positives, while patients that were not diagnosed with the deseases are unlabelled since *not being diagnosed* is different from *not*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3174590, IEEE Access

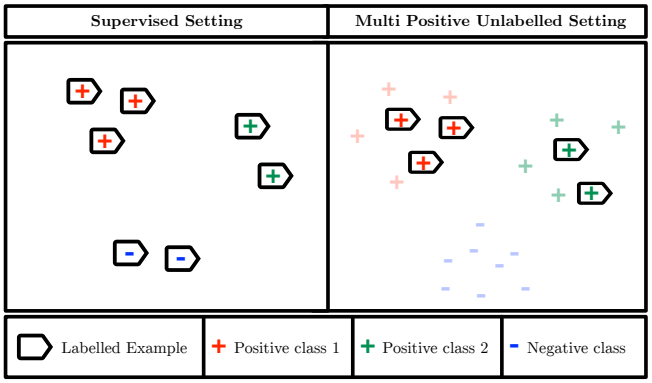

FIGURE 1. Visualization of the difference between supervised learning (SL) approach and the MPUL approach. In the SL approach every example of the training set has a label. In MPUL the training set is composed of both labelled and unlabelled examples. More in detail, labelled examples are available only for a portion of the positive classes.

having the disease [9]. The same problem occurs when a company wants to create an archive of researchers' home pages, using web-crawling techniques. Once downloaded, a web page should be classified to decide whether it is a researcher's home page or some other page. In such a context, the concept of the positive sample is well defined (the researcher's home page) while the negative concept is not well-established [7] because no real characterization of what is not a home page is supplied.

In a medical context, we might want to recognize vascular lesions starting from medical images [10]. In this particular case, accurately labeling vascular lesions could take more than one year and are then often left unlabelled, while it is relatively easy to assign positive labels to healthy individuals.

In all these scenarios, defining a method to exploit both positive and unlabeled samples could save time, money, human labor and the expert may focus his/her effort to only define what is good, avoiding the ungrateful task of recognizing what is not.

In the field of remote sensing, the problem is even more relevant since to deal with tasks like land cover or natural resources mapping from satellite images, samples can be supplied for a particular category of interest (i.e. urban, forest or wheat crop) while it could be seriously hard to identify negative samples that can completely describe the underlying landscape. In this scenario, PUL methods can be effectively used while standard supervised approaches simply cannot [11].

Although existing PUL frameworks are effective and demonstrate promising performance in a plethora of diverse applications, they share the limitation that only binary classification tasks are considered while, in many real-world scenarios multi-class classification problems are involved. As an example, large-scale e-commerce platforms have the objective to detect cyber security attacks, the malicious attacks can be considered as positive samples while benign transactions can be considered as negative ones. As there are normally multiple kinds of cyber security attacks, the positive samples can be categorized into multiple classes. Similar problems occur for personalized email filters. Some systems should allow some spam to pass through the system in addition to non-spam emails, which are also organized into several positive classes. In remote sensing classification analysis, samples can be supplied for multiple categories of interest without providing exhaustive coverage of the different underlying land cover classes. In this case, positive samples coming from many land cover classes are available while the negative class is hard to pinpoint to a single label.

To deal with such scenarios, Multi-Positive Unlabelled learning (MPUL) (see Fig. 1) was recently introduced [12]. In this learning setting, the available training data is composed of a labelled set of data composed of samples spanning over $K-1$ positive classes of interest and an unlabeled set of data that contains samples coming from K classes (the $K-1$ positive classes plus an additional negative one). The goal is to learn a multi-class classification model capable to categorize an unseen test sample, at inference time, in one of the K classes in which the classification problem is defined. While many frameworks and solutions were proposed for the binary setting, unfortunately, only few works [12], [13] have tackled the more general problem related to the multi-class scenario.

To cope with the under-explored Multi-Positive and Unlabelled Learning (MPUL) scenario, in this paper we propose a new deep learning-based framework especially tailored to cope with propositional (or tabular) data. Our framework, named `ProtoMPUL` (Prototype based Multi-Positive and Unlabelled Learning approach), combines metric learning and clustering strategies with the goal of model the set of positive classes on which the label information is available and, simultaneously, supplying also a characterization for the unknown negative one. `ProtoMPUL` involves three different stages: first, an autoencoder is trained to extract an initial data embedding; second, a metric learning strategy is adopted to stretch the manifold in which the data is embedded considering the available (positive) label information and; third, a deep clustering process is used to further refine the separability among the multiple positive classes and the negative one and, simultaneously, provides per-class prototypes. At inference time, the encoder network as well as the learnt prototypes are employed to classify previously unseen test samples.

To assess the behavior of our framework, we provide an experimental analysis of real-world benchmarks coming from different domains. The benchmarks for Multi-Positive and Unlabelled learning were generated by following a similar protocol as the one adopted by [12], [13]. Results show the effectiveness of the proposed framework w.r.t. recent competing methods especially tailored to deal with MPUL scenarios.

The rest of the paper is structured as follows: Section II discusses related work, preliminary definitions and technical background on Multi-Positive Unlabelled Learning are in-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3174590, IEEE Access

troduced in Section III, the `ProtoMPUL` framework is presented in Section IV, Section V describes and discusses the experimental evaluation. Section VI concludes and draws possible future works.

II. RELATED WORK

PU learning has been introduced in [8] where the problem has been defined and motivated as important for many practical applications where positive samples are cheap to acquire. For instance, it might be easy to acquire files of patients who have a particular disease, but it might be difficult to get the files of patients for which we have a negative diagnosis. At the same time, one could easily and cheaply get unlabelled data by using data for patients *without* a diagnosis. Due to its practical usefulness, PUL has recently been used in a wide variety of applicative scenarios [14] and has been further developed in many theoretical papers [8].

Methods to deal with the PUL setting can be roughly divided into three families [15]. The first family [16], [17] employs a two-step method where reliable negative samples are first selected and then used to train a traditional binary classifier. As the second step is a trivial application of binary classification, methods falling in this family mostly differ for the way they determine the reliable negative samples. For instance, in [17] the authors search features of positive samples that have a distribution that is markedly different in the unlabelled set. Reliable negative samples are then found by removing from the unlabelled set the samples having the positive feature signature. In [18] the authors propose to exploit probabilistic generative models to characterize the distribution of the positive samples, and to label as reliable negative samples those that are in the lowest density regions with respect to the positive ones. In addition, the proposed framework creates mixtures of generative models by adopting a bagging mechanism from the discriminative framework as an effective and cheap alternative to the classical Expectation Maximization strategy.

The second family of approaches formulates the PU learning problem as a cost-sensitive task [19], [20] where the errors on positive and negative samples are weighted differently. Here, differences between approaches can be significative, but mainly concern the schemas used for assigning the weights. For instance, in [20] a fixed weight is used on negative samples, while in [19] weights vary depending on the negative sample. In [21] a novel non-negative risk estimator for positive and unlabelled learning setting is introduced. The proposed estimator can be used to evaluate the risk for a set of symmetric losses (e.g., the mean-squared-error reduction) as well as train common binary classifiers for the case of positive and unlabelled learning.

The third family [22] models the unlabeled data as negative samples with label noise, thus PU learning reduces to a binary classification problem with one-sided label noise. [23] proposes an adaptive sampling framework for positive and unlabelled learning and for learning with label noise. The proposed framework iteratively estimates the class mislabeling probability with an adaptive sampling procedure which reduces the risk of selecting mislabeled instances for model training. Subsequently, it is able to construct generalizable models even when a large proportion of mislabeled instances is present in the data.

[12] was the first work to propose an extension of the standard (binary) PU learning to the multi-class scenario, introducing the Multi-positive Unlabelled (MPU) learning setting. The proposed approach is based on a one-step method in which the MPU learning problem is modeled minimizing multiple convex loss functions acting on labelled and unlabelled data.

More recently, [13] presented a new MPU learning approach based on a risk estimator derived from the one proposed in [12]. The authors build on the observation that the risk estimator proposed in [12] was affected by overfitting issues possibly caused by the unboundedness of the estimators. The authors propose a bounded risk estimator that alleviates this problem and avoids possible biases.

In our work, we use deep autoencoders, metric learning and deep clustering to induce an embedding that simplifies separating the negative class from the positive one. It is similar in spirit, albeit very different in practice, to the approaches in the first of the positive unlabelled learning families introduced above. Also, while most of the research efforts in the PU learning literature have been devoted to cope with a binary setting in which only one positive class is available at training time (e.g.,[8]), we concern ourselves with the problem of non-binary PUL settings, i.e., a setting where we have multiple positive classes and one negative class as those proposed in [11], [24], [12]. To this end, we build on the problem definition proposed in [12], but leveraging the combination of three deep learning techniques: the minimization of a metric learning loss, deep autoencoders, and deep clustering. As we show in the ablation study in Section V, each one of the three components we propose provides improved accuracy to the final solution.

III. MULTI-POSITIVE AND UNLABELLED LEARNING

The Positive and Unlabelled learning (PUL) setting [8] considers a scenario in which we dispose of a training dataset $D = \{P \cup U\}$ composed by a set P of positive samples and a set U of unlabelled samples. The unlabelled samples set U contains both positive and negative samples but their label information is not accessible. In this scenario, the PUL setting has the objective to exploit both P and U to learn a binary classification model allowing the assignment of a binary label (positive or negative) to new, previously unseen, samples.

The Multi-Positive and Unlabelled learning (MPUL) setting [12] generalizes the PUL approach to a multi-class scenario in which the positive set P contains samples belonging to $K-1$ classes (with the associated label information) while the set U contains samples (but not labels) of all the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3174590, IEEE Access

K classes, i.e., the $K-1$ positive classes plus samples belonging to the (unknown) negative class. In this scenario, the MPUL setting has the objective to learn a multi-class classification model from both P and U with the aim to classify previously unseen samples to one of the K classes.

More formally, in the MPUL scenario, we denote with $P = (X_l, Y_l)$, the positive examples. P is thus a pair formed by a set of examples $X_l = \{x_i\}_{i=1}^{N_l}$ and a set of labels $Y_l = \{y_i\}_{i=1}^{N_l}$. Each example x_i is a vector in $\mathbb{R}^d$, and the corresponding label y_i is an element of the set $\{1, \ldots, K-1\}$. In this setting, the additional set $U = X_u = \{x_i\}_{i=1}^{N_u}$ contains no label information. With X we indicate the union of labelled and unlabelled samples $X = \{X_l \cup X_u\}$. In the following N, N_l, and N_u represent the number of examples in the sets X, X_l, and X_u respectively. We emphasize that a sample $x_n \in X_u$ can belong to any one of the K classes ($K-1$ positive classes plus the negative one), but the label information is unknown at learning time.

IV. PROTOTYPE BASED MPUL

In this section, we introduce `ProtoMPUL`: a deep neural architecture (see Fig. 2) and the corresponding training algorithm to deal with Multi-Positive Unlabelled learning setting. The current architecture is tailored for propositional (tabular) data (e.g., all involved neural networks are fully connected).

`ProtoMPUL` exploits metric learning and deep clustering based strategies in order to characterize the $K-1$ positive classes as well as the unknown negative class. The result of the learning process consists of a prototype representation for each of the K classes. At the end of the process, classification can be performed by projecting the samples in the learnt embedded space and classifying them with the class corresponding to their closest prototype.

The algorithm to train `ProtoMPUL` goes through three different stages: *Stage 1)* an autoencoder is pre-trained on the full set of available data via a layer-wise incremental procedure [25]; *Stage 2)* the autoencoder is complemented by a metric learning loss allowing the system to integrate the information provided by the labelled samples; *Stage 3)* the class separability is reinforced by adopting a deep learning clustering approach [26] which, ultimately, provides the per-class prototype representation.

The three stages have the objective to modify progressively the manifold in which the original data is projected enforcing a cluster structure to separate all the classes involved in the classification problem.

The training algorithm for `ProtoMPUL` (see Algorithm 1) takes as inputs the set of positive and unlabelled samples with the associated label information (X_l, Y_l, X_u) and the total number of classes K. The algorithm is also parametric with respect to the stopping condition in the loops that characterize each stage. While more sophisticated convergence criteria could be devised[1], in our experiments we simply use a fixed number of epochs for each stage and treat it as a user-defined parameter. The results of the

[1]E.g., the difference in the loss of two consecutive iterations being smaller than a user-given parameter.

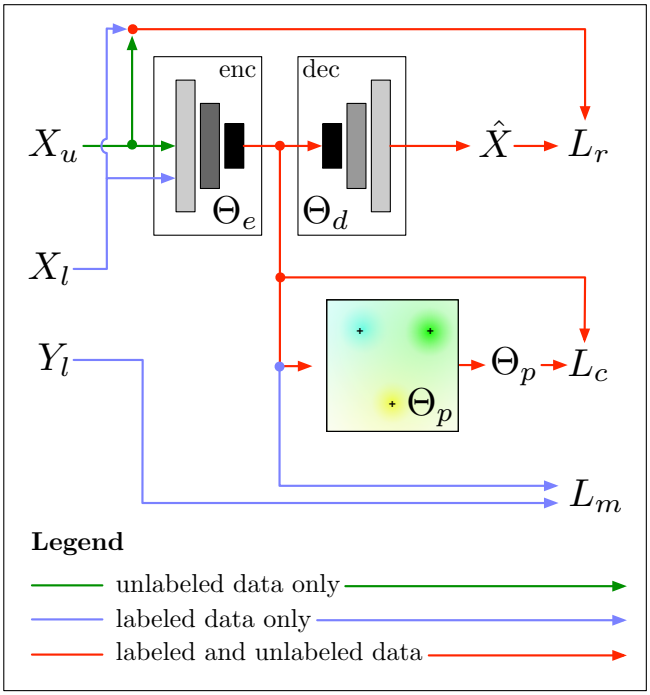

FIGURE 2. `ProtoMPUL` architecture

Algorithm 1: `ProtoMPUL`

Data: $X = X_l \cup X_u$, Y_l, K.

Stage 1
Use greedy layer-wise pretraining to initialize Θ_e, Θ_d by descending the gradient: $\nabla_{\Theta_e,\Theta_d} L_r(X)$;

Stage 2
while *stopping condition 2 not met* **do**
  update Θ_e, Θ_d by descending the gradient: $\nabla_{\Theta_e,\Theta_d} L_r(X)$;
  update Θ_e by descending the gradient: $\nabla_{\Theta_e} \gamma_m L_m(X_l, Y_l)$;
end

Stage 3
Θ_p = K-means(enc$_{\Theta_e}(X)$, K);
while *stopping condition 3 not met* **do**
  every *T iterations* **do**
    update the auxiliary target distribution B using (4);
    break if percentage of changed labels $\leq \delta$;
  end
  update Θ_e, Θ_d, Θ_p by descending the gradient: $\nabla_{\Theta_e,\Theta_d,\Theta_p} L_r(X) + \gamma_c L_c(X)$;
  update Θ_e by descending the gradient: $\nabla_{\Theta_e} \gamma_m L_m(X_l, Y_l)$;
end
Result: Θ_e, Θ_p

algorithm are the parameters Θ_e of the encoder function enc_{Θ_e} and the set of prototypes $\{p_i\}_{i=1}^K$ encoded by Θ_p.

Stage 1 performs a greedy layer-wise pre-training of the autoencoder. The autoencoder will provide the embeddings of the samples that are central to our approach. The goal of this stage is to initialize the autoencoder to a sensible starting point by training it to autoencode the complete set of data ($X = \{X_l \cup X_u\}$). In later stages the autoencoder will be refined using information from the label set Y_l and from the clustering loss. In all our experiments the autoencoder is based on fully connected layers and has shape $500 - 500 - 2000 - 10 - 2000 - 500 - 500$. The central (bottleneck) layer provides a new representation for the samples which are therefore embedded in some subspace of R^{10}. The layers in the autoencoder are based on the ReLU [27] activation function. As previously mentioned, the autoencoder model is trained layer-wise following the procedure proposed in [25]. The layer-wise greedy strategy incrementally trains the encoder and the decoder networks adding one layer at a time, facilitating the parameters optimization of the deep autoencoder. In addition, for this pretraining stage, we adopt a denoising strategy [28] to learn the model parameters in which the autoencoder network has to reconstruct a particular sample from its corrupted version.

The objective function optimized in this stage is the reconstruction loss function L_r over the whole set of samples X:

$$L_r(X) = \frac{1}{N}\sum_{i=1}^{N} \|\text{dec}_{\Theta_d}(\text{enc}_{\Theta_e}(x_i)) - x_i\|_2^2 \qquad (1)$$

where N is the number of labelled and unlabelled samples, $\text{enc}_{\Theta_e}(\cdot)$ (respectively $\text{dec}_{\Theta_d}(\cdot)$) is the encoder (respectively decoder) network with parameters Θ_e (respectively Θ_d). Here and in the following $\|\cdot\|_2$ denotes the L_2 norm.

Stage 2 of the framework refines the autoencoder by alternating a gradient descent on the L_r loss, with a gradient descent on a metric learning loss in which the label information associated with the positive samples is leveraged to learn a label-aware projection of the original data. The metric loss L_m is defined as follows:

$$L_m(X_l, Y_l) = \frac{2}{N_l^2 - N_l}\sum_{n=1}^{N_l}\sum_{n'=n+1}^{N_l} \mathbb{1}_{[y_n=y_{n'}]}[d_{nn'}^2 - \beta_s]_+ + (1 - \mathbb{1}_{[y_n=y_{n'}]})[\beta_d - d_{nn'}^2]_+ \qquad (2)$$

where $\mathbb{1}_{y_n=y_{n'}}$ is an indicator function that returns 1 if x_n and $x_{n'}$ belong to the same class and 0 otherwise. $d_{nn'}^2$ is the squared euclidean distance between the embeddings of x_n and $x_{n'}$ while $[\cdot]_+$ is the ramp function defined as $[z]_+ = max(0, z)$ commonly used in the hinge loss or in ReLU units. β_s and β_d are two margin parameters that allow one to adjust the contribution of the two components of the L_m loss function. The loss induces a penalty when the squared distance $d_{nn'}$ between two samples of the same class ($\mathbb{1}_{[y_n=y_{n'}]}$) is larger than β_s since in this case $[d_{nn'}^2 - \beta_s]_+$ would be positive. Similarly, it induces a penalty when the squared distance of samples of different classes is smaller than β_d. Thus, the smaller β_s is, the closer two samples of the same class are required to be to not incur in a penalty; while the bigger β_d is, the more distant two samples of different classes are required to be. The goal of the L_m loss function is to exploit the available label information Y_l to stretch the geometric manifold induced by the embedded representation with the aim of integrating the class information. In so doing it forces arranging the samples belonging to the same class to be close together (minimizing the term $[d_{nn'}^2 - \beta_s]_+$) and samples belonging to different classes to be far away from each other (minimizing the term $[\beta_d - d_{nn'}^2]_+$). We note that the L_m loss can only be applied on the set of labelled samples X_l, which explains the need of optimizing L_m and L_r separately.

Stage 3 starts with the initialization of the model prototypes with the centroids derived by a clustering step on the current embedded representation of X. In principle any distance-based clustering algorithm could be used, in our experiments we adopted the well-known K-means clustering algorithm [29] setting the number of clusters equal to K. The first $K-1$ centroids are initialized in the center of mass of the embeddings of samples belonging to the $K-1$ positive classes. The last centroid is initialized randomly selecting a sample with a probability proportional to the distance of the closest centroid.

After the initialization of the prototypes, the main loop in Stage 3 relies on an alternate optimization strategy where enc_{Θ_e}, dec_{Θ_d}, and the deep soft-clustering networks are tuned using X by descending the gradient $\nabla_{\Theta_e,\Theta_d,\Theta_p} L_r(X) + \gamma_c L_c(X)$, and enc_{Θ_e} is further refined using the labelled samples by descending the gradient $\nabla_{\Theta_e} L_m(X_l, Y_l)$.

More specifically, given the initial prototypes, in this Stage we exploit the deep clustering strategy pioneered in [26] as a way to further improve the data partitioning. The deep clustering approach starts by computing a soft assignment Q between the embeddings and the prototypes; then, based on these assignments, the embedded data representation as well as the prototypes are updated descending the gradients of the clustering loss L_c. The loss is based on the Kullback-Leibler (KL) divergence between the distribution representing the soft-assignments of samples to the prototypes and an auxiliary target distribution B:

$$L_c(X) = \text{KL}(B\|Q) = \sum_{n=1}^{N}\sum_{k=1}^{K} b_{nk}\log\frac{b_{nk}}{q_{nk}} \qquad (3)$$

The auxiliary target distribution B is computed every T iterations using the formula:

$$b_{nk} = \frac{q_{nk}^2/\bar{q}_k}{\sum_{k'=1}^{K} q_{nk'}^2/\bar{q}_{k'}} \qquad (4)$$

where $\bar{q}_k = \sum_{n=1}^{N} q_{nk}$ is used as a normalization factor so to avoid preferring bigger clusters. As emphasized in

[26], distribution B is defined in terms of Q implying that the minimization of $L_c(X)$ is a form of self-learning. In fact, in a self-learning setting, an initial classifier is used to label an unlabelled dataset in order to train itself on its own high confidence predictions. In our case the distribution B plays the role of the high-confidence predictions and by minimizing the Kullback-Leibler (KL) divergence between B and Q, one gets the information needed to further update the embeddings to improve the partitioning of the samples.

To compute the soft assignment, following [26], [30], we exploit the Student's t-distribution as a kernel to measure the similarity between points in the embedded space and the prototypes [31]:

$$q_{nk} = \frac{(1 + ||\text{enc}_{\Theta_e}(x_n) - p_k||^2)^{-1}}{\sum_{k'=1}^{K}(1 + ||\text{enc}_{\Theta_e}(x_n) - p_{k'}||^2)^{-1}} \quad (5)$$

where $\text{enc}_{\Theta_e}(x_n)$ is the embedded representation of the n-th sample, p_k (respectively $p_{k'}$) is the k-th (respectively k'-th) prototype, and q_{nk} is the soft assignment between sample x_n and prototype p_k. Such distribution forces the assignment to have sharper probabilities (closer to 0 or 1) by squaring the original distribution and then normalizing it [32].

To weight the contribution of metric learning and deep clustering, we multiply the metric loss L_m and the clustering loss L_c by the user-tunable hyper-parameters γ_m and γ_c, respectively.

The algorithm terminates providing the learnt prototypes Θ_p. The prototypes, along with the learnt encoder enc_{Θ_e} allow classifying new samples by first mapping them into the embedded space and then assigning them the class of the nearest prototype:

$$f_{\Theta_e\Theta_p}(x) = \arg \min_{k \in \{1 \dots K\}} ||\text{enc}_{\Theta_e}(x) - p_k||^2.$$

The time complexity of the framework is given by the sum of the complexities of the three stages it is built on. We observe that the time complexity of stages two and three are dominated by the complexity of the gradient descent (backpropagation) algorithm, which is $O(YWNE)$, where Y is the number of layers in the network, $W = O(R^2)$ is the number of weights per layer, R is the maximum number of neurons per layer, N is the number of examples and E is the number of epochs. Stage one is more costly because it involves the training procedure for the greedy layer-wise pretraining strategy, which repeats the training Y times yielding a total complexity of $O(Y^2WNE)$. In summary, the total complexity is just the sum of the three given complexities which is dominated by the $O(Y^2WNE)$ term.

V. EXPERIMENTAL EVALUATION

In this section, we introduce the experimental settings and datasets we have adopted to evaluate the proposed framework, as well as the results and the related discussion.

We provide several quantitative evaluations. In the first one, we provide an ablation study about the different components on which `ProtoMPUL` is built. In the second evaluation, we compare the proposed approach w.r.t recent competitors considering a setting similar to the one reported in [12]. In the third and fourth evaluations, we assess the sensitivity of the different competing approaches to the variation of the number of positive classes as well as to the variation of the number of labelled samples. Finally, we summarize and discuss information about the execution time of the different methodologies.

A. EXPERIMENTAL SETTINGS AND DATASETS

We consider recent state-of-the-art methods as well as reference methods:

- our main competitors are two recent methods proposed in [13], named AREA (Alternative Risk EstimAtor) and UREA (Unbiased Risk EstimAtor). Both methods are based on the concept of an empirical risk estimator. While the former considers an (unbounded) estimator that can suffer from overfitting, the latter solves such a problem allowing a better generalization on unseen data;
- similarly to what was done in [12], we consider a linear Support Vector Machine [33] (named *Linear SVM*) approach learnt on the original multi-class classification problem. This is not a competitor for our approach. Rather, we keep it as a reference method since labels for all the classes (including the negative one) are available at the training stage;
- additionally, with respect to what has been done in previous studies, since deep learning approaches are non-linear methods, we also consider a radial basis function Support Vector Machine [33] ((named *RBF SVM*)) as an additional reference method trained on the fully labelled data set.

It is worth stressing that, while the first two approaches (UREA and AREA) are direct MPU learning competitors (using exactly the same amount of label information as well as the same learning setting as `ProtoMPUL`), the two SVM methods are deployed in the standard supervised setting: all samples in the training set are labelled and they cover the whole set of K classes. Their performances should then be taken as an *upper bound* of the possible performances an MPUL approach might achieve.

We evaluate the performances of the different approaches on nine standard multi-class classification tasks. The datasets characteristics are reported in Table 1.

Given a dataset, for each positive class, we consider a 60/20/20 (train/validation/test) split of the associated samples. Among the training samples of a specific class, only half of them (i.e., 30% of the class samples) are associated to label information, while the rest are assigned to the unlabelled training set. The rest of the class samples belong to the validation and to the test set. Regarding the negative class, half of the samples are assigned to the unlabelled training set, while the rest are assinged to the test set. We

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3174590, IEEE Access

note that at training time one does not know the identity of the negative class, hence hyper-parameters are only estimated using the positive classes and the validation set cannot contain negative examples. As usual, the validation set is used to choose the best hyper-parameter settings for the different competing approahces, while the test set is used to assess the ability of the different methods to generalize on unseen samples.

As evaluation metric, we choose the F1 score [34] since, even if not perfect [35], it is very popular in the evaluation of the predictive performances in class unbalanced scenarios.

To avoid possible bias due to the way in which datasets are split, we repeat the process described above 5 times and we average the obtained results.

TABLE 1. Datasets characteristics

Dataset	# Samples	# Feat.	# Classes
FMnist	70000	784	10
Har	10299	561	6
Landsat	6435	36	6
Mnist	70000	784	10
Optdigits	5620	64	10
Pendigits	10992	16	10
Semeion	1593	256	10
Sonar	208	60	2
Usps	9298	256	10

Except for the SVM methods, all the other approaches (including `ProtoMPUL`) are learnt via stochastic gradient descent through the Adam optimizer [36] with a batch size equals to 256.

For each dataset and each method, we used a grid search to find the best hyper-parameters configuration using a validation set based on the positive samples only. Regarding `ProtoMPUL`, the hyper-parameter values for learning rate, weight decay and γ_c are 10^{-3}, 10^{-4} and 10^{-1}, respectively; γ_s is chosen from { 0.1, 1 }, β_s is set equal to β_d and their value ranges in the set { 1, 10, 100 }. For the convergence criterium, the stopping threshold tol (i.e., the percentage of changed labels in Algorithm 1, Stage 3) is set to 0.1%, while the number of maximum iterations in the three stages is set to $10\,000$.

For what it concerns UREA and AREA, the hyper-parameters λ and η are chosen from the range $\{10^{-4}, ..., 10^{-1}\}$.

For the SVM classifier, the complexity parameter C ranges in the set $\{10^{-1}, ..., 10^{2}\}$ while for *RBF SVM*, the kernel radius varies considering values in the set of possible values $\{10^{-5}, ..., 10^{-1}\}$. For all methods, data are rescaled via z-score normalization.

Experiments are carried out on a workstation equipped with an Intel® Xeon® CPU E5-2643@3.30GHz, with 128 GB of RAM. No Graphical Processing Unit was employed during the experiments.

B. ABLATION STUDY

The first experiment we have conducted has the objective to validate the importance of the components of `ProtoMPUL`.

TABLE 2. Summary of the `ProtoMPUL` ablations involved in the experimental evaluation. "lwp" stands for the layer-wise pretraining of the autoencoder.

	Stage 1	Stage 2		Stage 3		
	lwp	L_r	L_m	L_r	L_m	L_c
$Abla_0$	✗	✓	✓	✓	✓	✓
$Abla_1$	✓	✓	✗	✓	✓	✓
$Abla_2$	✓	✗	✓	✓	✓	✓
$Abla_3$	✓	✓	✓	✓	✓	✗
$Abla_4$	✓	✓	✓	✗	✓	✓
$Abla_5$	✓	✓	✓	✓	✗	✓

To this end, we compare the performances of `ProtoMPUL` to several of its ablations.

To choose the ablation settings, we considered only configurations that involve coherent subsets of the components that our framework adopts.

The summary of the different `ProtoMPUL` ablations are reported in Table 2. The first ablation ($Abla_0$) evaluates the appropriateness of the layer-wise training procedure associated to the first stage of `ProtoMPUL`. Successively, $Abla_1$ and $Abla_2$ are specially tailored to evaluate the importance of the set of loss functions employed in the second stage of our framework while, the remaining ablations ($Abla_3$, $Abla_4$ and $Abla_5$) assess the interplay of the full set of loss functions in the third stage of our proposal.

Table 3 reports the results of the ablation study. We can observe that, generally, `ProtoMPUL` outperforms all its different ablations or it attains comparable performances. The only case in which a different trend is exhibited is related to the *Sonar* dataset. This is probably because this benchmark is the smallest one we have in terms of samples (around 200) and this factor negatively influences the `ProtoMPUL` train procedure. It is also worth mentioning that, as we show in Section V-C, all MPUL methods fail on this dataset, which might indicate a general problem with this dataset that also affect the ablation study.

We can note that the layer-wise strategy seems worthy of interest since $Abla_0$ is dominated most of the time by `ProtoMPUL`. For the rest of the ablations, the complete system is usually better, often by a large margin. This trend is violated only by $Abla_5$ on the OptDigits dataset, but the same ablation setting is much worse than the complete system on all the other datasets. Based on this evidence we conclude that all the components of `ProtoMPUL` are important and their interplay contributes to the state-of-the-art results that we show in the next Section.

C. EVALUATION OF COMPETING APPROACHES

Table 4 summarizes the results, in terms of F1 score, obtained by the different competing approaches on the set of benchmarks introduced in Section V. We can observe that `ProtoMPUL` outperforms the direct competing methods (AREA and UREA) on the majority of the datasets. When this does not happen (*FMnist*), performances are still largely comparable. Regarding the comparison between `ProtoMPUL` and the two SVM models, we note that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3174590, IEEE Access

TABLE 3. Results (in terms of F1 score) of the different ProtoMPUL ablations as well as the performances of ProtoMPUL. For each method, both average and standard deviation are reported. Best results are highlighted in **blue**.

Dataset	ProtoMPUL	Abla_0	Abla_1	Abla_2	Abla_3	Abla_4	Abla_5
FMnist	**0.818±0.024**	0.802±0.021	0.238±0.120	0.427±0.133	0.747±0.053	0.645±0.041	0.750±0.011
Har	**0.967±0.006**	0.935±0.009	0.264±0.138	0.507±0.178	0.775±0.161	0.835±0.142	0.841±0.179
Landsat	**0.842±0.017**	0.821±0.020	0.752±0.186	0.711±0.174	0.551±0.082	0.674±0.141	0.682±0.129
Mnist	**0.954±0.006**	0.923±0.010	0.059±0.042	0.776±0.031	0.873±0.076	0.825±0.077	0.858±0.060
Optdigits	0.973±0.004	0.943±0.005	0.643±0.060	0.889±0.106	0.899±0.112	0.921±0.113	**0.976±0.008**
Pendigits	0.924±0.011	**0.929±0.013**	0.763±0.142	0.745±0.070	0.794±0.081	0.906±0.025	0.890±0.015
Semeion	**0.849±0.025**	**0.849±0.025**	0.047±0.056	0.769±0.110	0.775±0.072	0.809±0.120	0.796±0.113
Sonar	0.346±0.049	0.405±0.052	0.439±0.192	0.105±0.000	0.473±0.129	**0.481±0.076**	0.308±0.089
Usps	**0.929±0.008**	0.915±0.008	0.047±0.040	0.682±0.121	0.875±0.061	0.920±0.015	0.863±0.098

RBF SVM achieves better performances on almost all the datasets compared to *Linear SVM*. This clearly shows that the former represents a more robust and effective upper bound for the MPU learning approaches. It is also interesting to note that ProtoMPUL outperforms the *Linear SVM* method on some datasets (*Landsat*, *Mnist* and *Optdigits*) even though *Linear SVM* has complete knowledge about the involved set of classes. When compared to the other methods with complete knowledge of the classes (*RBF SVM*), ProtoMPUL achieves results that are not so far from them, thus demonstrating the quality of the proposed framework.

We assess the statistical significance of the obtained results using two statistical tests. We compute a Friedman test [37] to assess if the difference in the observed accuracies is statistically significant and, successively we set up a one-tail independent Student's-T test to assess if the observed average F1 scores can support the hypothesis $\mu_1 > \mu_2$, where μ_1 is the performance of the ProtoMPUL and μ_2 is the performance of a direct competitor (AREA or UREA). In all cases, we consider the test passed when it supports the alternate hypothesis at the 0.05 confidence level.

The Friedman test comfortably rejects the null hypothesis (the F1 scores would be the same regardless of the algorithm) at the confidence level 0.05 since the non-parametric statistical test provides us a p-value of 0.00178. Table 5 summarizes the results of the independent Student's-T test. Green values indicate results with a significance level better than 0.05 and red values indicate results that are not statistically significant. We can observe that the difference in the observed mean values are always statistically significant (at the given confidence level) when ProtoMPUL is compared with UREA. In the case of the AREA approach, the difference in the observed mean is not significant only 2 times out of 9. It is worth pointing out that: *i)* the first of these two cases (FMnist dataset, comparison with AREA) corresponds to the one result where ProtoMPUL is not better than the competitor, i.e., the result is actually a positive outcome for us (the test $\mu_2 > \mu_1$ also fails the test at the 0.05 significance level); *2)* the second case (Sonar dataset) is one where all MPUL methods fail to learn anything useful.

D. SENSITIVITY TO THE NUMBER OF POSITIVE CLASSES

In this experiment, we evaluate the sensitivity of the approaches to the number of positive classes. To this end, we have chosen two datasets among those having the maximal number of labels (Pendigits and Semeion) and evaluate how varying the number of positive classes impacts on the classification performances. The number of positive classes has been varied using values from the set {2,4,6,8,9} (9 being the largest possible value given the labels in the unmodified dataset). Results are reported in Fig. 3.

These results show that ProtoMPUL achieves superior performances with respect to the competitors (AREA and UREA) in both benchmarks and almost all cases. When this is not the case, i.e., for the Pendigits dataset and number of classes equal to 4 and 6, the results are still largely comparable. More in general, we observe that all the MPUL methods have decreasing performances as the number of positive classes increases, but the decrease appears to be less severe in the case of ProtoMPUL.

E. SENSITIVITY TO THE AMOUNT OF LABELLED SAMPLES

In this experiment, we evaluate the sensitivity of the algorithms to the number of labelled samples on Pendigits and Semeion. We chose these datasets based on the following factors: *i)* these are the same two datasets we used in the previous experiment, this simplify the experimentation, allows for easier reporting, and keeping the choice fixed seems fairer; *ii)* the two datasets are both hard ones, as mentioned in the previuos section, all MPUL approaches have decreasing performances as the number of positive classes grows and these datasets are among the ones with the largest number of classes; *iii)* they are a large and a small dataset, allowing us to assess how the algorithms work in the two regimes.

In the experiment we varied the amount of labelled samples for the positive classes. Specifically, we let the percentage (with respect to the positive examples in the experiment reported in Section V-C) of positively labelled examples to vary in the set: $\{20\%, 40\%, 60\%, 80\%, 100\%\}$. Results are provided in Fig. 4.

We can observe that ProtoMPUL outperforms the competitors (AREA and UREA) no matter the percentage of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3174590, IEEE Access

TABLE 4. Results (in terms of F1 score) of the different competing approaches on the set of considered benchmarks. For each method, both average and standard deviation are reported. Best results for MPU learning approaches are highlighted in bold. Best results for SVM baselines are emphasized in a light blue tonality.

Dataset	*Linear SVM*	*RBF SVM*	AREA	UREA	ProtoMPUL
FMnist	0.858±0.001	0.909±0.001	**0.836±0.003**	0.749±0.006	0.818±0.024
Har	0.986±0.002	0.989±0.001	0.960±0.005	0.917±0.011	**0.967±0.006**
Landsat	0.816±0.008	0.925±0.008	0.815±0.008	0.773±0.024	**0.842±0.017**
Mnist	0.904±0.002	0.981±0.000	0.874±0.005	0.729±0.002	**0.954±0.006**
Optdigits	0.958±0.004	0.988±0.003	0.932±0.004	0.864±0.050	**0.973±0.004**
Pendigits	0.925±0.006	0.994±0.001	0.879±0.008	0.780±0.027	**0.924±0.011**
Semeion	0.863±0.018	0.931±0.012	0.659±0.024	0.619±0.041	**0.849±0.025**
Sonar	0.744±0.033	0.105±0.000	0.269±0.160	0.167±0.049	**0.346±0.049**
Usps	0.933±0.005	0.978±0.003	0.902±0.004	0.893±0.014	**0.929±0.008**

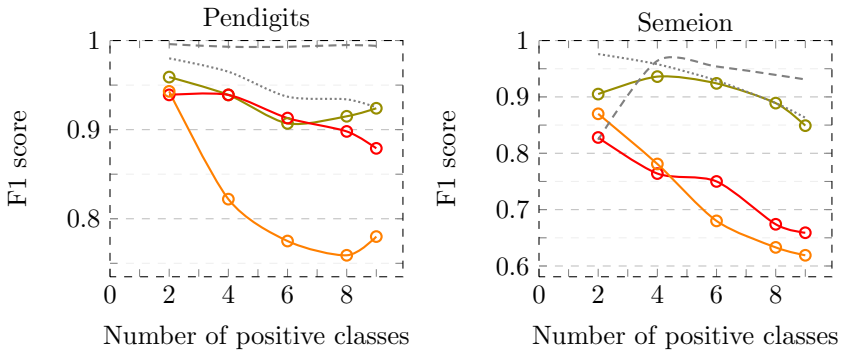

FIGURE 3. Performances, as the number of positive classes varies, of the tested methods on the Pendigits and Semeion datasets for the ProtoMPUL (in olive), AREA (in red) and UREA (in orange) algorithms. Gray lines show the performances of the reference algorithms: Linear SVM (dotted) and RBF SVM (dashed).

TABLE 5. p-values results of the paired Student's-T test with null hypothesis $\mu_1 > \mu_2$ (μ_1 is the average performance of ProtoMPUL). Green values indicate statistically significant results with significance level of 0.05 while red values indicate not statistically significant results.

Dataset	ProtoMPUL vs AREA	ProtoMPUL vs UREA
FMnist	9.33e-01	1.25e-04
Har	4.00e-02	9.87e-06
Landsat	6.18e-03	3.89e-04
Mnist	7.00e-09	3.48e-13
Optdigits	1.06e-07	6.29e-04
Pendigits	3.82e-05	2.01e-06
Semeion	9.11e-07	2.54e-06
Sonar	1.67e-01	2.08e-04
Usps	7.25e-05	5.32e-04

labelled samples from positive classes is employed. In addition, it exhibits a more stable behavior than the competitors.

F. COMPUTATIONAL COSTS

In this section, we provide an overview of the computational cost of training the tested approaches. The training time (in seconds) is reported in Table 6. We do not provide figures for the inference time because, as usual for neural networks, the inference time is negligible when compared to the training time. Also, for what it concerns ProtoMPUL specifically, the total cost is given by the cost of encoding the example using enc_{Θ_e} and by the (negligible) cost of comparing the result with the set of learned prototypes. All methods have been run on CPU as already mentioned in Section V-A. ProtoMPUL is clearly the algorithm with the highest computational requirements, but it is worth noticing that it is also the one with the smallest variability between experiments. This is due to the fact that the time complexity of ProtoMPUL is dominated by the layer-wise pretraining we perform in the first phase, and this is largely affected by the number of layers; a parameter that is kept fixed in our experiments. In our opinisson, while the time performances are clearly not favorable to ProtoMPUL, they are still within reason and can be largely justified by the better performances of the learnt model.

TABLE 6. Execution time in seconds. LSVM is a shorthand for *Linear SVM*, RSVM is a shorthand for *RBF SVM*.

Dataset	LSVM	RSVM	AREA	UREA	ProtoMPUL
FMnist	688	57 180	44 304	46 224	34 650
Har	28	960	5 056	5 168	8 118
Landsat	1	1	400	416	9 384
Mnist	652	125 660	45 264	46 672	34 896
Optdigits	1	1	512	528	4 206
Pendigits	1	1	464	496	4 656
Semeion	1	1	464	464	2 568
Sonar	1	1	112	112	144
Usps	8	200	2 240	2 288	7 464

VI. CONCLUSION

In this work, we have presented a new framework for Multi-Positive and Unlabelled learning (MPUL) for the classification of propositional (or tabular) data. Our framework, ProtoMPUL, combines deep metric learning and deep clustering approaches with the goal to model the set of $K-1$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3174590, IEEE Access

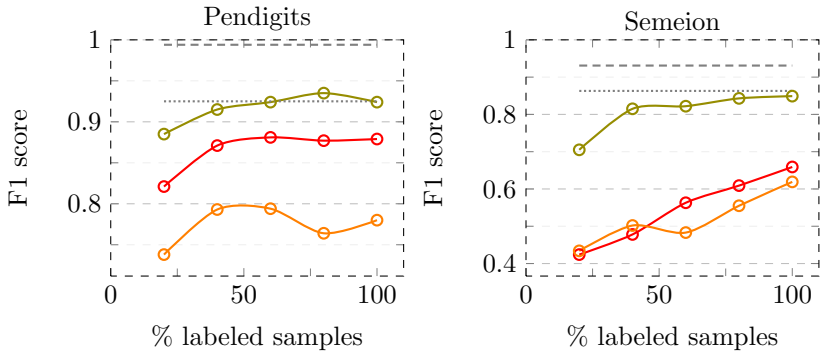

FIGURE 4. Performances, as the number of positive examples varies, of the tested methods on the Pendigits and Semeion datasets for the ProtoMPUL (in olive), AREA (in red) and UREA (in orange) algorithms. Gray lines show the performances of the reference algorithms: Linear SVM (dotted) and RBF SVM (dashed).

positive classes on which label information is available and, simultaneously, providing also a characterization of the unknown negative class. The training algorithm for ProtoMPUL is based on three stages. In the first one an autoencoder is incrementally trained to extract a preliminary embedding of the data. Successively, the learning procedure is complemented by a metric loss function with the aim to involve the available label information. The last stage integrates a deep learning clustering process to further enforce class separability and extract a set of prototypes (one for each of the $K-1$ positive classes plus an additional one for the negative class). At inference time, the encoder network as well as the learnt prototypes are employed to classify previously unseen test samples.

The experimental comparison with the state of the art MPUL competitors on standard propositional datasets has demonstrated the quality of the proposed solution, while the in depth ablation analysis has highlighted that all the different components of ProtoMPUL play an important role in its performances. The experiments with a varying number of positive classes have shown that all MPUL methods tend to have decreasing performances as the number of positive classes grows, but also that ProtoMPUL is remarkably robust to this issue. The same kind of observations can be made about the performances of the algorithms when the number of labelled examples decreases: again ProtoMPUL has better overall performances and appears to be more robust than the competitors. All these benefits have to be counterbalanced by a larger computational demand for training the model.

Several possible research ramifications are possible for future works. Among them, we plan to extend our framework to work with other kinds of data (e.g., images, multivariate time-series) adapting the autoencoder network to the specificity of the input data. We also intend to extend the proposed methodology to situations and scenarios in which multiple unknown negative classes can be present. Finally, to reduce the computational demands of the algorithm, we plan to optimize phase 1, either by replacing the layer-wise pretraining with some alternative strategy less computationally demanding, or by changing the neural architecture to avoid the pretraining altogether.

REFERENCES

[1] M. Aldinucci, S. Rabellino, M. Pironti, F. Spiga, P. Viviani, M. Drocco, M. Guerzoni, G. Boella, M. Mellia, P. Margara *et al.*, "HPC4AI: an AI-on-demand federated platform endeavour," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*, 2018, pp. 279–286.

[2] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1, pp. 44–53, 08 2017.

[3] C. C. Aggarwal, X. Kong, Q. Gu, J. Han, and P. S. Yu, "Active learning: A survey," in *Data Classification: Algorithms and Applications*, C. C. Aggarwal, Ed. CRC Press, 2014, pp. 571–606.

[4] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, 2020.

[5] M. Carbonneau, V. Cheplygina, E. Granger, and G. Gagnon, "Multiple instance learning: A survey of problem characteristics and applications," *Pattern Recognit.*, vol. 77, pp. 329–353, 2018.

[6] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 5, pp. 845–869, 2014.

[7] H. Yu, J. Han, and K. C. Chang, "PEBL: positive example based learning for web page classification using SVM," in *KDD*. ACM, 2002, pp. 239–248.

[8] J. Bekker and J. Davis, "Learning from positive and unlabeled data: a survey," *Mach. Learn.*, vol. 109, no. 4, pp. 719–760, 2020.

[9] M. Claesen, F. D. Smet, P. Gillard, C. Mathieu, and B. D. Moor, "Building classifiers to predict the start of glucose-lowering pharmacotherapy using belgian health expenditure data," *CoRR*, vol. abs/1504.07389, 2015.

[10] M. A. Zuluaga, D. R. Hush, E. J. F. D. Leyton, M. H. Hoyos, and M. Orkisz, "Learning from only positive and unlabeled data to detect lesions in vascular CT images," in *MICCAI*, vol. 6893, 2011, pp. 9–16.

[11] W. Li, Q. Guo, and C. Elkan, "One-class remote sensing classification from positive and unlabeled background data," *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.*, vol. 14, pp. 730–746, 2021.

[12] Y. Xu, C. Xu, C. Xu, and D. Tao, "Multi-positive and unlabeled learning," in *IJCAI*, 2017, pp. 3182–3188.

[13] S. Shu, Z. Lin, Y. Y., and L. Li, "Learning from multi-class positive and unlabeled data," in *ICDM*. IEEE, 2020, pp. 1256–1261.

[14] K. Jaskie and A. Spanias, "Positive and unlabeled learning algorithms and applications: A survey," in *IISA*. IEEE, 2019, pp. 1–8.

[15] C. Zhang, C. Gong, T. Liu, X. Lu, W. Wang, and J. Yang, "Online positive and unlabeled learning," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 2248–2254.

[16] D. Ienco and R. G. Pensa, "Positive and unlabeled learning in categorical data," *Neurocomputing*, vol. 196, pp. 113–124, 2016.

[17] H. Yu, J. Han, and K. C.-C. Chang, "Pebl: positive example based learning for web page classification using svm," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 239–248.

[18] T. M. A. Basile, N. D. Mauro, F. Esposito, S. Ferilli, and A. Vergari, "Ensembles of density estimators for positive-unlabeled learning," *J. Intell. Inf. Syst.*, vol. 53, no. 2, pp. 199–217, 2019.

[19] C. Elkan, "The foundations of cost-sensitive learning," in *IJCAI*, B. Nebel, Ed. Morgan Kaufmann, 2001, pp. 973–978.

[20] W. S. Lee and B. Liu, "Learning with positive and unlabeled examples using weighted logistic regression," in *ICML*, vol. 3, 2003, pp. 448–455.

[21] R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama, "Positive-unlabeled learning with non-negative risk estimator," in *NIPS*, 2017, pp. 1675–1685.

[22] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari, "Cost-sensitive learning with noisy labels," *J. Mach. Learn. Res.*, vol. 18, pp. 155:1–155:33, 2017.

[23] P. Yang, J. T. Ormerod, W. Liu, C. Ma, A. Y. Zomaya, and J. Y. H. Yang, "Adasampling for positive-unlabeled and label noise learning with bioinformatics applications," *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1932–1943, 2019.

[24] T. Peng, W. Zuo, and F. He, "SVM based adaptive learning method for text classification from positive and unlabeled documents," *Knowl. Inf. Syst.*, vol. 16, no. 3, pp. 281–301, 2008.

[25] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[26] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*. PMLR, 2016, pp. 478–487.

[27] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[28] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.

[29] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[30] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation." in *Ijcai*, 2017, pp. 1753–1759.

[31] L. van der Maaten and G. E. Hinton, "Visualizing high-dimensional data using t-sne," *JMLR*, vol. 9, pp. 2579–2605, 2008.

[32] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39 501–39 514, 2018.

[33] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[34] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[35] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *arXiv preprint arXiv:2010.16061*, 2020.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations, ICLR*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[37] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.