Enhancing Graph-Based Semisupervised Learning via Knowledge-Aware Data Embedding

Enhancing Graph-based Semi-supervised Learning via Knowledge-aware Data Embedding

Dino Ienco[1], and Ruggero G. Pensa[2]
[1]IRSTEA, UMR TETIS, LIRMM, Univ. of Montpellier, Montpellier
[2]University of Turin – Dept. of Computer Science, Turin, Italy

Semi-supervised learning is a family of classification methods conceived to reduce the amount of required labeled information in the training phase. Graph-based methods are among the most popular semi-supervised strategies: a nearest neighbor graph is built in such a way that the manifold of the data is captured and the labeled information is propagated to target samples along the structure of the manifold. Research in graph-based semi-supervised learning has mainly focused on two aspects: i) the construction of the k-nearest neighbors graph and/or ii) the propagation algorithm providing the classification. Differently from the previous literature, in this paper we focus on data representation with the aim of incorporating semi-supervision earlier in the process. To this end, we propose an algorithm that learns a new knowledge-aware data embedding via an ensemble of semi-supervised autoencoders to enhance a graph-based semi-supervised classification. The experiments carried out on different classification tasks demonstrate the benefit of our approach.

***Index Terms*—semi-supervised learning, autoencoders, data embedding, ensembles**

I. Introduction

Data labeling is a time-consuming and cost-prohibitive task. To cope with this issue, semi-supervised learning (SSL) methods were conceived to reduce as much as possible the amount of labeled information required to train a classification model by taking advantage of the abundant and rich information provided by unlabeled samples. Among the different algorithmic solutions proposed, those based on graph-based label propagation constitute one of the main families of transductive semi-supervised techniques. In graph-based semi-supervised learning (GBSSL), the data samples constitute the nodes of a nearest neighbor graph that is constructed so as to capture the manifold of the data. The classification is then performed by propagating the information from labeled to unlabeled samples along the edges of the graph.

Standard GBSSL pipelines usually involve two steps [1]: i) construction of a k-Nearest Neighbors (kNN) graph without using the information supplied by the labels, and ii) propagation of the label information across the graph through a learning algorithm. Figure 1(a) describes the standard pipeline adopted for the GBSSL scenario where the kNN graph is derived directly from the data and the labels are only introduced as input of the classification algorithm. Most research efforts related to GBSSL mainly address these two aforementioned points. For instance, [1] proposes a solution to the problem of over sparsification affecting the construction of kNN graphs. Similarly, many approaches were proposed [2], [3], [4] to address the label propagation problem, and the main differences among them are related to which structural characteristic of the graph is considered as predominant.

Manuscript received XXX; revised XXX. Corresponding author: Dino Ienco (email: dino.ienco@irstea.fr).

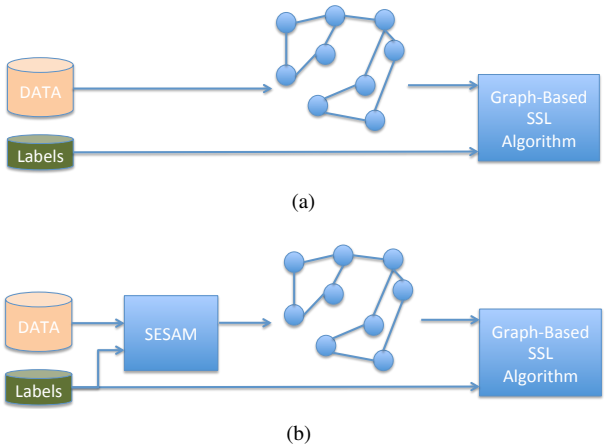


Fig. 1. Standard GBSSL pipeline (a) and knowledge-aware GBSSL pipeline (b) where label information is considered earlier in the process.

Despite the huge research efforts in improving SSL algorithms, to the best of our knowledge, with the exception of the work presented in [5], none of the previous work on GBSSL addresses the challenge of integrating label information at an early stage of the pipeline. In this work, we tackle exactly this problem: Figure 1(b) depicts our proposed strategy according to which the label information is exploited earlier in the pipeline. The kNN graph, in fact, is built upon the new embedding representation learnt by our approach, called *SESAM* (SEmi-Supervised Autoencoder enseMble). Our contribution consists in learning a knowledge-aware data embedding that successively feeds a standard transductive GBSSL pipeline. To this purpose, we propose to learn such knowledge-aware embedding using a semi-supervised autoencoder (SSAE). Autoencoders are neural network models mainly employed to learn new data representations setting up a reconstruction task that encodes and decodes the original data. The encoder compresses the data into an embedded representation while the decoder reconstructs the original data from such embedding. The neural network layer that produces the embedding is called bottleneck layer. A semi-supervised autoencoder extends the autoencoder by integrating a classification task alongside reconstruction. More precisely,

the bottleneck layer is exploited for both reconstruction and classification. Most of the previous literature involving semi-supervised autoencoders [6], [7] makes the assumption that all the examples, used to feed the deep learning model, are associated to a label information and, successively, the reconstruction as well as the classification task are managed at the same time over all the data. Here, differently from those strategies [8], [7], we do not have labels for all the data seen by the model. More precisely, we have only a small portion of the data with associated label information. To cope with this scenario, we propose a different strategy that does not require fully labeled data. More in detail, at each training epoch, two learning mechanisms are alternated: the first performs a reconstruction task involving the whole set of data, while the second tackles the reconstruction as well as the classification task considering only the portion of labeled data. Furthermore, we leverage an ensemble of diverse SSAEs to build the final data embedding. The intuition behind this choice is that an ensemble of network models captures the multifaceted relationships that naturally exist in the data [9], [10]. Our experimental study demonstrates the effectiveness of our proposal: by comparing the performances of *SESAM* in conjunction with a state-of-the-art transductive GBSSL algorithm w.r.t. the same GBSSL algorithm coupled with other strategies, we show that injecting as soon as possible the label information in the transductive classification process leads to better classification results.

II. Background

In a common transductive semi-supervised learning scenario, we can distinguish two sets of examples: $X^u = \{x_i\}_{i=1}^N$ that is a set of N examples without associated class information, and $X^l = \{(x_j, y_j)\}_{j=1}^M$ of M examples with an associated class information. More in detail, each data example $x_j \in X^l$ has an associated label variable $y_j \in C$, where C is the set of possible labels. Generally, the set of unlabeled examples is much bigger than the set of labeled ones ($N >> M$) with the more realistic scenario where only very few examples (e.g., one) per class exist. The objective of the methods dealing with the semi-supervised learning scenario is to propagate the label information from the labeled set X^l to the unlabeled X^u resulting in the classification of all the unlabeled examples in the dataset according to the predefined set of classes in C. Note that, in a transductive learning scenario both training and test examples are available when the model is learnt.

GBSSL is a class of approaches that enables classification when very few information about labels exists. An algorithm of this family first builds a nearest neighbor graph $\mathbb{G} = (V, E)$ where the set V of nodes in the graph corresponds to the whole set of examples ($X^u \cup X^l$), and E is a set of edges such that $(x_i, x_j) \in E$ iff x_j is among the k nearest neighbors of x_i. Successively, it propagates the information from labeled nodes to unlabeled ones through the edges E of $\mathbb{G}$ in a transductive setting. Finally, a class value is assigned to each unlabeled sample/node $x_i \in X^u$ according to the result of the propagation process. It is worth noting that state-of-the-art methods essentially differ on how propagation is performed.

One of the first label propagation algorithms for GBSSL was proposed in [11]. The method iteratively propagates the information from the labeled node to the unlabeled ones leveraging a smooth diffusion kernel. In [12], the authors proposed a propagation algorithm that handles the correlation between labels for node classification. A different strategy is proposed in [3]. Here, the algorithm assumes that nodes with a large number of neighbors can be confidently classified. Recently, [2] introduced a new algorithm that integrates most advantages of the previous works. More in detail, the proposed method is confidence-aware (it assumes that nodes with a large number of neighbors can be confidently classified) and the approach is applicable to both homophily and heterophily networks.

III. The SESAM algorithm

In this section we first provide an overview of our framework and successively we describe *SESAM*, an ensemble learning approach that uses semi-supervised autoencoders to learn knowledge-aware representations with the aim of enhancing transductive GBSSL strategies. One of the key point related to *SESAM* is that it does not require that the data are fully labeled since we set up a learning strategy, reported in Algorithm 2, that alternates a reconstruction step involving the whole set of data and a multi-task step (reconstruction and classification) only considering labeled data.

A. Overview of the approach

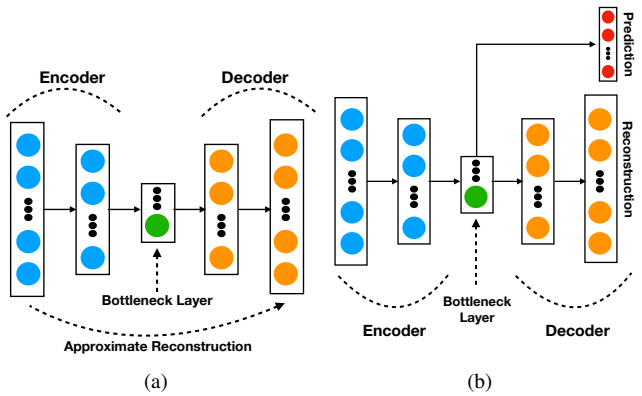
The main goal of our framework is to introduce the use of available knowledge as early as possible in the semi-supervised learning process. Our contribution is mainly focused on the knowledge-aware representation step, which can run independently of any standard k-nearest neighbors graph construction approaches. To this purpose, differently from standard pipelines (Figure 1(a)) where graph $\mathbb{G}$ is build on top of the original data, here (Figure 1(b)), we propose to build $\mathbb{G}$ on top of the knowledge-aware embeddings learnt by an ensemble of semi-supervised autoencoders leveraging both labeled and unlabeled information. Figure 2 depicts the structure of a classical autoencoder as well as the structure of our SSAE. In the remainder of this section we will first introduce some basic notions about autoencoders, then we will detail our contribution.

B. Autoencoders

An autoencoder is a multi-layer neural network that operates successive linear (or non linear) transformations of the input data with the goal to reconstruct the original data. It is composed of two modules: i) an encoder network that extracts the embeddings from the input data X and ii) a decoder network that, from the low-dimensional embedding, reconstructs the original data. Commonly, the last layer of the encoder (usually denoted as bottleneck layer) provides the embeddings of the data.

The function optimized by the autoencoder is as follows:

$$L_{ae}(\Theta_1, \Theta_2) = \frac{1}{|X|} \sum_{i=1}^{|X|} ||X_i - AE(X_i, \Theta_1, \Theta_2)||^2 \quad (1)$$



(a) (b)

Fig. 2. Autoencoder (a) and Semi-Supervised Autoencoder (b) architectures.

where $|X|$ is the cardinality of the example set X, $||\cdot||$ is the ℓ^2 norm, Θ_1 and Θ_2 are the parameters of the encoder and decoder part of the autoencoder, respectively. *AE* is the function representing the autoencoder. The goal is to learn the model parameters (Θ_1, Θ_2) such that *AE* reconstructs the original data X. Typically, an autoencoder has as many neurons in the output layer as the number of neurons in the input layer. This is due to the fact that the autoencoder goal is to reconstruct the original data. Furthermore, the autoencoder network is layered and symmetric. In the encoder module, the number of neurons in the hidden layers decreases gradually while, symmetrically, the number of neurons increases in the decoder module. Efficient reconstruction is obtained exploring the search space considering the network parameters Θ_1, Θ_2.

C. Semi-supervised Autoencoder Ensemble

Inspired by the strategy proposed in [9], [10] where, instead of a single autoencoder model, an ensemble of models is leveraged, in our approach we rely on an ensemble learning setting that train a set of semi-supervised autoencoders (SSAE) with the goal of learning several diverse embeddings from the same set of data. First, we learn a set of diverse SSAE from the set of data X. Second, we extract and juxtapose together all the different low-dimensional embeddings to obtain the new data representation. Successively, such a new data representation is exploited to construct $\mathbb{G}$, the kNN Graph, finally handled by the semi-supervised graph-based classification algorithm.

1) Semi-Supervised Autoencoder

A Semi-Supervised Autoencoder [10] (SSAE) is a multi-task neural network architecture that extends simple autoencoder to solve two different tasks, simultaneously: i) an unsupervised task in which the network reconstructs the original data via an encoding-decoding schema (Equation 1), and ii) a supervised task in which the bottleneck layer of the autoencoder is exploited to perform classification. The loss function associated to the prediction task is the classical categorical cross-entropy between the labels and the predictions:

$$L_{cl}(\Theta_1, \Theta_3) = -\frac{1}{|X^l|}\sum_{j=1}^{|X^l|}\sum_{c=1}^{|C|} y_{jc} \cdot log(\hat{y}_{jc}) \qquad (2)$$

where $y_{j\cdot}$ is the one hot encoding of the label information for the example j, $\hat{y}_{j\cdot} = SSAE(X_j, \Theta_1, \Theta_3)$ is the probability distribution over the set of classes, Θ_1 are the parameters of the encoder (similarly to Equation 1), and Θ_3 are the parameters employed to make the supervised prediction. X^l is the subset of labeled examples. Merging the unsupervised and the supervised components, the (multi-task) loss function related to the SSAE is defined as:

$$L_{SSAE}(\Theta_1, \Theta_2, \Theta_3) = L_{ae} + \lambda L_{cl} \qquad (3)$$

where

$$L_{ae} = \frac{1}{|X|}\sum_{i=1}^{|X|} ||X_i - AE(X_i, \Theta_1, \Theta_2)||^2 \qquad (4)$$

$$L_{cl} = -\frac{1}{|X^l|}\sum_{j=1}^{|X^l|}\sum_{c=1}^{|C|} y_{jc} \cdot log(\hat{y}_{jc}) \qquad (5)$$

We underline that, conversely to most previous works [8], [7], the multi-task loss function involves two different subsets of data. The unsupervised loss (L_{ae}) considers the whole set of examples X while the supervised loss (L_{cl}) only considers the subset of the data X_l to which the label information is associated. We remind that $X_l \subset X$. The joint optimization of the reconstruction and classification tasks helps to learn embeddings that effectively resume the information available in the original data also considering the background knowledge carried out by the small amount of labeled samples available at training time. Here, we describe the strategy we adopt to optimize Equation 3 that is inherited from a recent work considering the use of SSAE for semi-supervised clustering [10]. For all the layers of our neural network architecture we employ the *ReLU* [13] activation function. The only exception concerns the output layer associated to the reconstruction task in which we use a sigmoid activation function due to the fact that data attributes are rescaled in the interval $[0, 1]$. Considering the classification task, the bottleneck layer of our encoder-decoder architecture is directly linked to an output layer to provide classification. In this case, we adopt a linear activation function combined with a SoftMax normalization. Algorithm 1 reports the optimization procedure we employ. In a generic epoch, the procedure optimizes: i) the unsupervised loss associated to data reconstruction (Θ_1 and Θ_2) on the set of data X (line 5-6), and ii) both the unsupervised and supervised loss (Θ_1, Θ_2 and Θ_3), corresponding to the reconstruction and classification tasks respectively, considering the set of data X_l (line 7-8). The learning of parameters Θ_1, Θ_2 and Θ_3 is achieved via mini-batches gradient descent based approaches.

2) Ensemble Construction

SESAM produces the new data representation exploiting an ensemble of diverse SSAE. Diversity is a crucial characteristic when an ensemble learning schema is proposed [9]. Stacking together the embeddings obtained by a set of Semi-Supervised Autoencoder with exactly the same network structure will induce redundancy in the new data representation. To deal with this aspect, we introduce diversity by generating a set of SSAE with different sizes of both the hidden and bottleneck layers. Varying the size of such layers, for each SSAE model,

Algorithm 1 SSAE optimization

Require: X^u: the set of unlabeled examples, X^l: the set of labeled examples, N_EPS: the number of epochs, fl_size: the first hidden layer size, b_size: the bottleneck layer size
Ensure: $\Theta_1, \Theta_2, \Theta_3$.
1: $k = 0$
2: $X = X^u \cup \{x|(x, y) \in X^l\}$
3: $initSSAE(fl\_size, b\_size)$
4: **while** k < N_EPS **do**
5: Update Θ_1 and Θ_2 by descending the gradient:
6: $\nabla_{\Theta_1,\Theta_2} \frac{1}{|X|} \sum_{i=1}^{|X|} ||X_i - SSAE(X_i, \Theta_1, \Theta_2)||^2$
7: Update Θ_1, Θ_2 and Θ_3 by descending the gradient:
8: $\nabla_{\Theta_1,\Theta_2,\Theta_3} \frac{1}{|X^l|} \sum_{j=1}^{|X^l|} ||X_j - SSAE(X_j, \Theta_1, \Theta_2)||^2 - \lambda\Big(\frac{1}{|X^l|} \sum_{j=1}^{|X^l|} y_j \cdot log(SSAE(X_j, \Theta_1, \Theta_3))\Big)$
9: $k++$
10: **end while**
11: **return** $\Theta_1, \Theta_2, \Theta_3$

Algorithm 2 Ensemble Construction Strategy

Require: X: the whole set of examples, X^l: the set of labeled examples, ENS_SIZE: the size of the ensemble
Ensure: X_{newR}: the concatenated representation generated by the ensemble of SSAE
1: $X_{newR} = \emptyset$
2: $p = getNumAttributes(X)$
3: $k = 0$
4: **while** k < ENS_SIZE **do**
5: $fl\_size = rand(p/2, p)$
6: $b\_size = rand(p/4, p/2)$
7: $SSAE = constructSSAE(X, X^l, fl\_size, b\_size)$
8: $t_{rep} = extractEmbedding(SSAE, X, X^l)$
9: $X_{newR} = concat(X_{newR}, t_{rep})$
10: $k++$
11: **end while**
12: **return** X_{newR}

provides a new data representation that include embeddings carrying out information at different granularities.

For each SSAE model of the ensemble, diversity is implemented by picking at random the size of the different hidden layers of the network architecture. Each model has four layers and we have that the output and input layers have the same size that is equal to the number of attributes in the data. Conversely, we can vary the size of the three other layers: the first, the second (bottleneck) and the third hidden layers. Due to the symmetric structure of the SSAE, the first and third hidden layers may have the same size. Such constraint restricts the number of randomly picked parameters to two: the size of the bottleneck layer and the size of the first/third hidden layers. Given the fact that the number of attributes describing our data is equal to p, we sample uniformly at random the size of the first/third hidden layer in the range $[p/2, p)$ while we sample uniformly at random the size of the bottleneck layer in the range $[p/4, p/2)$. To prevent possible information loss due to high level of compression in the bottleneck layer, if $p/4 < 3$, the bottleneck layer size is set to 3. The ranges $[p/2, p)$ and $[p/4, p/2)$ are chosen since they represent the two biggest and not overlapping intervals from which the size of the hidden layers can be picked up. Once the set of diverse SSAE is constructed and trained, each model is applied to the original data and, successively, the new data representation X_{newR} is obtained by concatenating together all the different embeddings extracted by the SSAE neural networks (line 4-9 of Algorithm 2). We underline that our approach is different from *dropout* [14] and other similar strategies. The main goal of *dropout* is to inhibit the different connections avoiding the co-adaptation of neurons belonging to the same layer. In our case, the models in the ensemble set are completely independent from each other and the embeddings extracted by each neural network are successively combined. Furthermore, the reasoning behind our strategy is to provide a new data representation exploiting an ensemble of diverse models that are capable to extract embeddings at different granularities describing coarse relationships as well as fine-grained interactions.

IV. Experiments

In this section, we first assess the performances of our embedding strategy for GBSSL classification by comparing it with several state-of-the-art competitors and baselines. Then, we report the results of the sensitivity analysis w.r.t. the ensemble size. Successively, we compare the alternating optimization strategy introduced in Algorithm 1 with a sequential alternative. Finally, we provide some insights about the visual inspection of the representations learnt by *SESAM*.

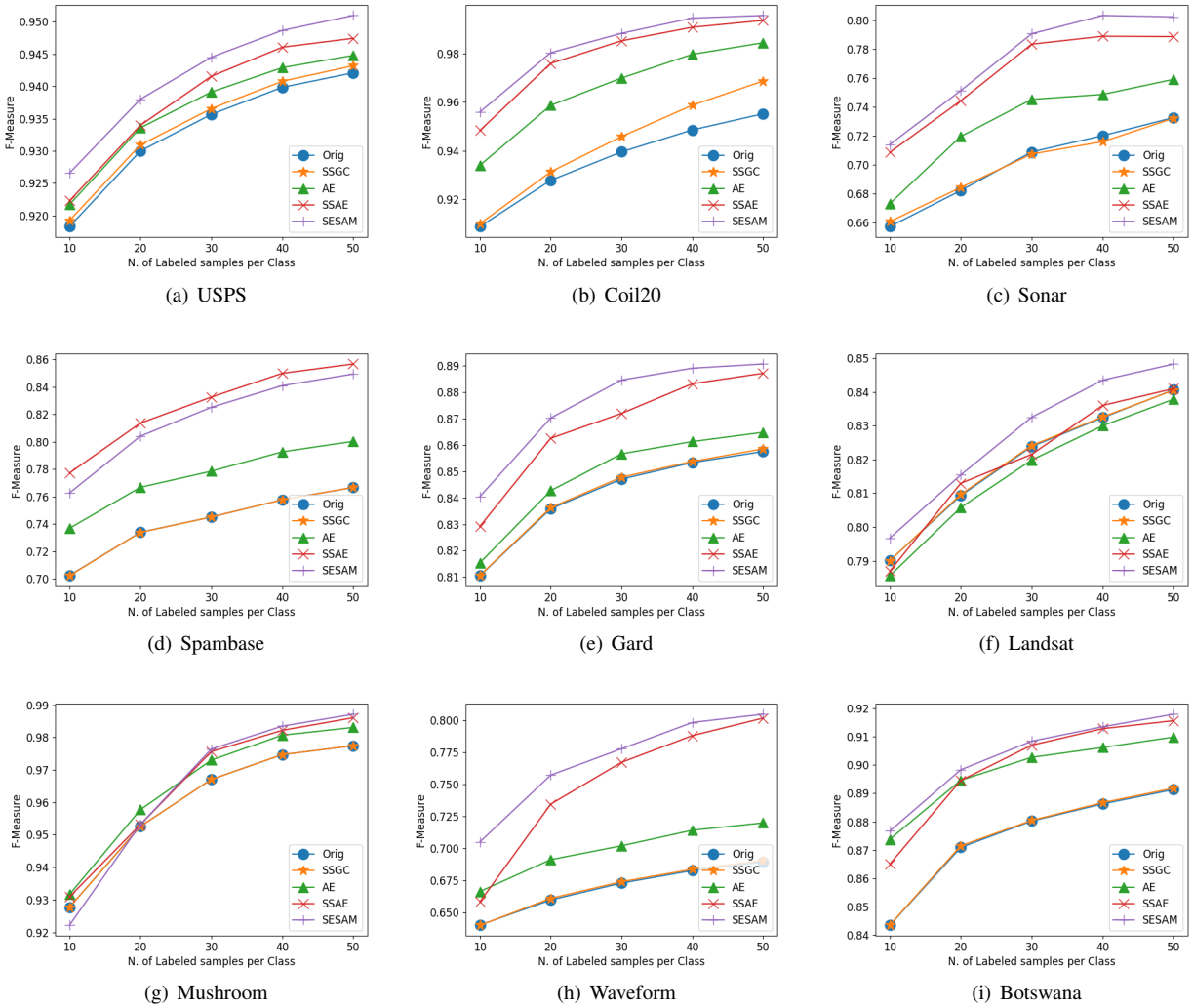
Regarding the competitors, we compare the results achieved using the original data representation (*ORIG*), with those supplied by our framework (*SESAM*). As additional baseline, following the work proposed in [5], we consider a modified kNN graph that involve must-link and cannot-link constraints derived by the available knowledge: an extra edge is added between two nodes if they belong to the same class while an edge connecting two nodes belong to two different classes is deleted. We name such baseline *SSGC* (Semi-Supervised Graph Construction). We introduce an unsupervised version of *SESAM* that stacks together the low-dimensional representations induced by an ensemble of fully unsupervised autoencoders. We name this competitor *AE*. We also consider the representation supplied by only one semi-supervised autoencoder to better understand the added value supplied by the ensemble solution. This approach is named *SSAE*.

TABLE I
DATASETS CHARACTERISTICS

Dataset	# Examples	# Features	# Classes	Size of *SESAM* repr.
USPS [15]	9 298	256	10	2 865.22 ± 95.49
Coil20 [16]	1 440	1 024	20	11 536.14 ± 415.34
Sonar [16]	208	60	2	664.18 ± 23.61
Spambase [16]	4 601	57	2	645.57 ± 21.02
Gard [17]	1 673	592	7	6 559.82 ± 205.90
Landsat [16]	4 435	36	6	393.00 ± 12.63
Mushroom [16]	8 124	22	2	1 368.64 ± 48.17
Waveform [16]	5 000	40	3	435.10 ± 15.10
Botswana [18]	3 248	145	14	1 631.92 ± 58.88

We evaluate the performances of the different approaches on nine multi-class classification tasks. The following datasets characteristics are reported in Table I: dataset name and reference, number of examples, number of features, number of classes and the average size (with standard deviation) of the representations generated by *SESAM*.

For each competing representation, we first construct the mutual k-nearest neighbors graph [1] (K=20) then, to obtain



(a) USPS (b) Coil20 (c) Sonar

(d) Spambase (e) Gard (f) Landsat

(g) Mushroom (h) Waveform (i) Botswana

Fig. 3. Results (in terms of F-Measure) of the graph-based SSL algorithm considering *ORIG*, *AE*, *SSAE* and *SESAM* data representations and varying the number of labeled samples per class.

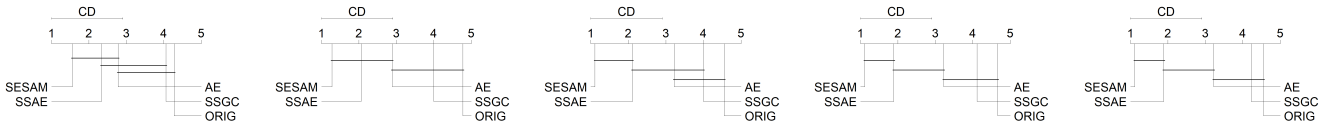
the final classification, we use the Confidence-Aware Modulated Label Propagation (CAMLP) algorithm [2], a recently proposed label propagation strategy, one of the best performing state-of-the-art methods. We assess the performances of the different competing methods varying the level of supervision in terms of labeled samples per class in the interval $[10, 50]$ with a step of 10. The sample selection process is repeated 30 times for each number of per-class labels and then average results are reported. For *SESAM*, the kNN graph leverages the embedding whose average size is reported in Table I.

As evaluation metric we choose the F-Measure [19] since it is well suited to evaluate predictive performances in class unbalanced scenarios. In our context, considering multi-class classification task, we calculate the average F-Measure weighted by the class support (the number of true examples for each class). The per-class F-Measure is defined as the harmonic mean between the Precision and Recall measures. We can define the F-Measure for the class j as $\textit{F-Measure}_j = 2 \cdot \frac{Precision_j \cdot Recall_j}{Precision_j + Recall_j}$, where $Precision_j$ and $Recall_j$ are the precision and recall computed for class j, respectively. The weighted multi-class F-Measure is defined as $\textit{F-Measure} = \sum_{j \in C} \frac{|X_j^u|}{|X^u|} \cdot \textit{F-Measure}_j$, where C is the set of classes, $|X_j^u|$ is the cardinality of the unlabeled examples set belonging to class j and $|X^u|$ is the number of unlabeled examples.

We fix the number of models in the ensemble to 30. Parameters optimization is achieved via the *RMSProp*[1] optimizer [20] and we fix the learning rate to 5×10^{-4} with a decay factor of 5×10^{-5}. Considering the whole evaluation, the deep learning methods are trained for 200 epochs. The *AE* baseline is trained with a batch size of 16. The same batch size is adopted for the reconstruction task of each SSAE on the whole set of data (Algorithm 1, line 5-6) while a batch size of 8 is used for the multi-task optimization of the SSAE (Algorithm 1, line 7-8). The value of parameter λ is fixed to 1. Experiments are carried out on a workstation with Intel(R) Xeon(R) E5-2667 v4@3.20Ghz CPU, 256Gb of RAM and a TITAN X GPU. *SESAM* is implemented using *Keras* python library[2].

[1]http://climin.readthedocs.io/en/latest/

[2]https://github.com/fchollet/keras



(a) 10 labeled samples (b) 20 labeled samples (c) 30 labeled samples (d) 40 labeled samples (e) 50 labeled samples

Fig. 4. Critical difference plots of the Nemenyi test according to different numbers of labeled samples per class. In all plots, $CD = 2.1288$.

A. Results of the comparative analysis

Figure 3 summarizes the results of the comparison over the different benchmarks. Although the benchmarks we used are heterogeneous in terms of dimensionality and classification task, we observe that the different representations exhibit a coherent behavior, i.e., they influence in the same way the classification performances. More in detail, the representation supplied by *SESAM* provides most of the time the best classification results regardless of the values of labeled samples per class. The only exception is constituted by the *Spambase* dataset, in which *SSAE* and *SESAM* obtain comparable performances, with the former slightly outperforming the latter. This is probably due to the fact that the ensemble variation, in this scenario, induces some instability in the graph construction process. Despite this fact, we can observe that the other baselines are still less effective, in terms of *F-Measure*, compared to our proposal. All these results support the fact that introducing the available knowledge as soon as possible in the GBSSL pipeline positively influences the whole process. We can also note that *SESAM* systematically outperforms its unsupervised counterpart (*AE*) as well as the original data representation and the *SSGC* strategy. This means that, even if the knowledge exploited to learn the new representation is limited, it is still valuable to steer towards knowledge-aware embeddings that incorporate class discriminating information.

To asses the statistical quality of our approach we use the Friedman statistics and the Nemenyi test [21]. For any number of labeled samples per class, the null hypothesis of the Friedman test is comfortably rejected with p-value < 0.005, so we can proceed with the post-hoc Nemenyi test. According to this test (see the critical difference plots in Fig. 4), *SESAM* is always significantly better than *ORIG* and *SSGC* (p-value < 0.1), although its improvement is not significant w.r.t. *SSAE*. With 30 or more labeled samples, the improvement of our approach is significant w.r.t. its unsupervised counterpart (*AE*) as well. Instead, *SSAE* is significantly better than *ORIG* for any level of supervision, while its improvement is significant w.r.t. *SSGC* only with 40 labeled samples or more. Differently from *SESAM*, *SSAE* never outperforms *AE* to a significant extent.

B. Sensitivity analysis w.r.t. the ensemble size

With the aim of assessing the sensitivity of *SESAM* w.r.t. the number of SSAEs in the ensemble set, we vary the ensemble size (number of models) between 10 and 50 with a step of 10, and we fix the number of labeled samples per class to 10. Figure 5 reports the results of this experiment. We note that *SESAM* exhibits a stable behavior w.r.t. this hyper-parameter over all the considered benchmarks. This

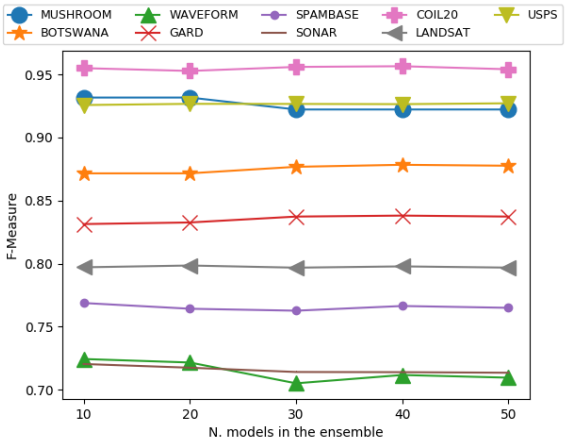


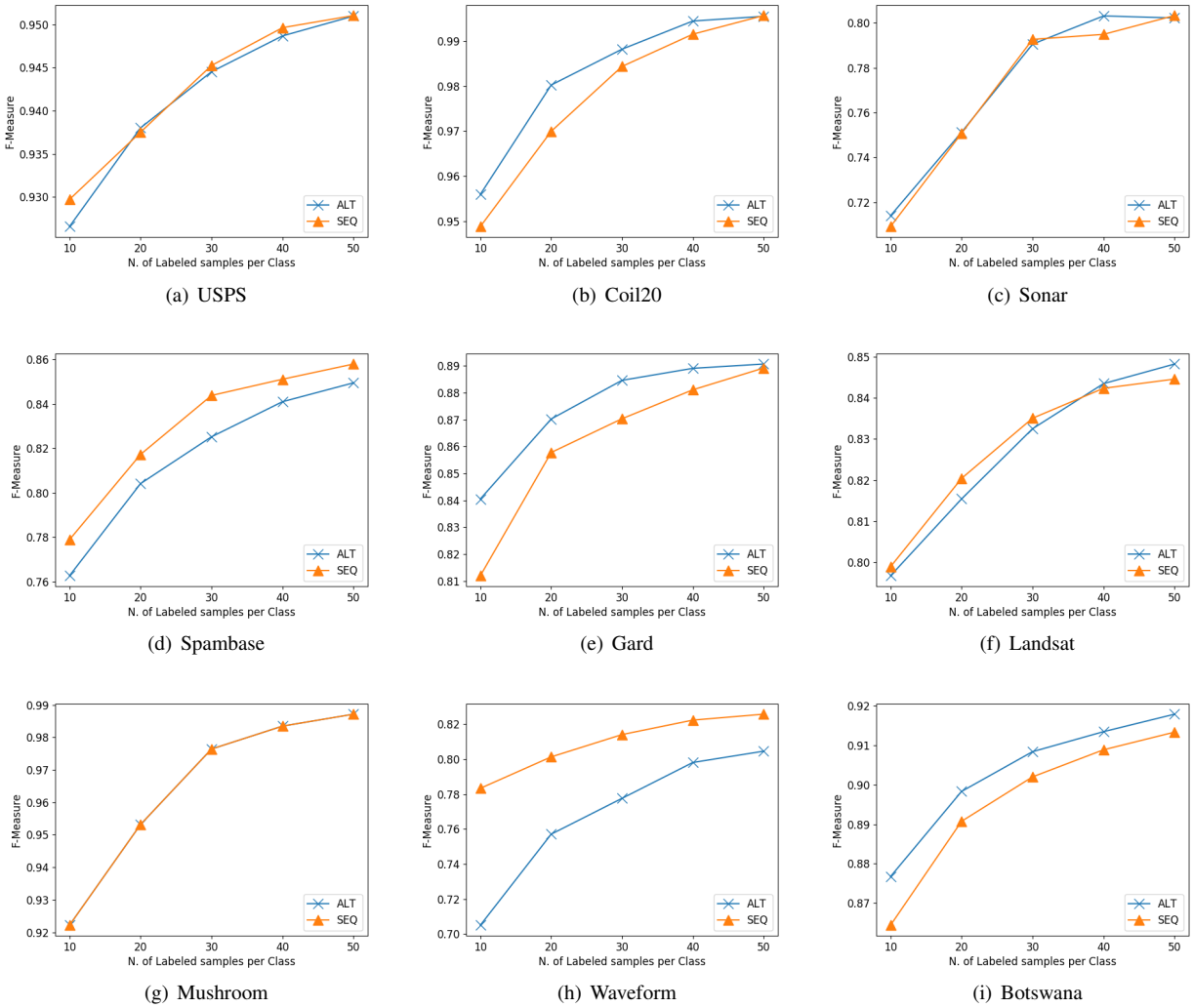
Fig. 5. Sensitivity analysis of the behavior of *SESAM* (in terms of F-Measure) varying the number of models in the ensemble set (between 10 and 50 with a step of 10) with the number of labeled samples per class fixed to 10.

indicates that, generally, 10 models are enough to obtain good performances, and considering a bigger number of models in the ensemble set does not bring any significant advantage to the graph-based semi-supervised classification task.

C. Alternating vs sequential optimization of SESAM

Here, we evaluate the behaviour of *SESAM* considering two different optimization schemes. The first one, that we name *ALT*, corresponds to the optimization procedure depicted in Algorithm 1. As we have explained in Section III, in this optimization procedure, the learning of the parameters involves the alternate optimization of the fully unsupervised task and the semi-supervised one. The second optimization scheme consists in a sequential optimization of the fully unsupervised task followed by the semi-supervised one. Here, we first optimize the unsupervised autoencoder over all the data and, then, we fine-tune the parameters of the model w.r.t. the semi-supervised task considering only the small set of labeled samples. We name such variant *SEQ*. Figure 6 reports the results of such comparison in terms of F-Measure.

We observe that the behavior of *SESAM* is not systematically the same according to the different ways of optimizing its internal parameters. We can observe differences regarding the following datasets: *Coil20*, *GARD*, *Botswana*, *Spambase* and *Waveform*. More in detail, on the first three datasets (*Coil20*, *GARD*, *Botswana*), *ALT* optimization seems more effective than *SEQ*. The opposite phenomenon can be observed on the two last datasets (*Spambase* and *Waveform*). On the other datasets, the two optimization methods behave similarly.



(a) USPS (b) Coil20 (c) Sonar

(d) Spambase (e) Gard (f) Landsat

(g) Mushroom (h) Waveform (i) Botswana

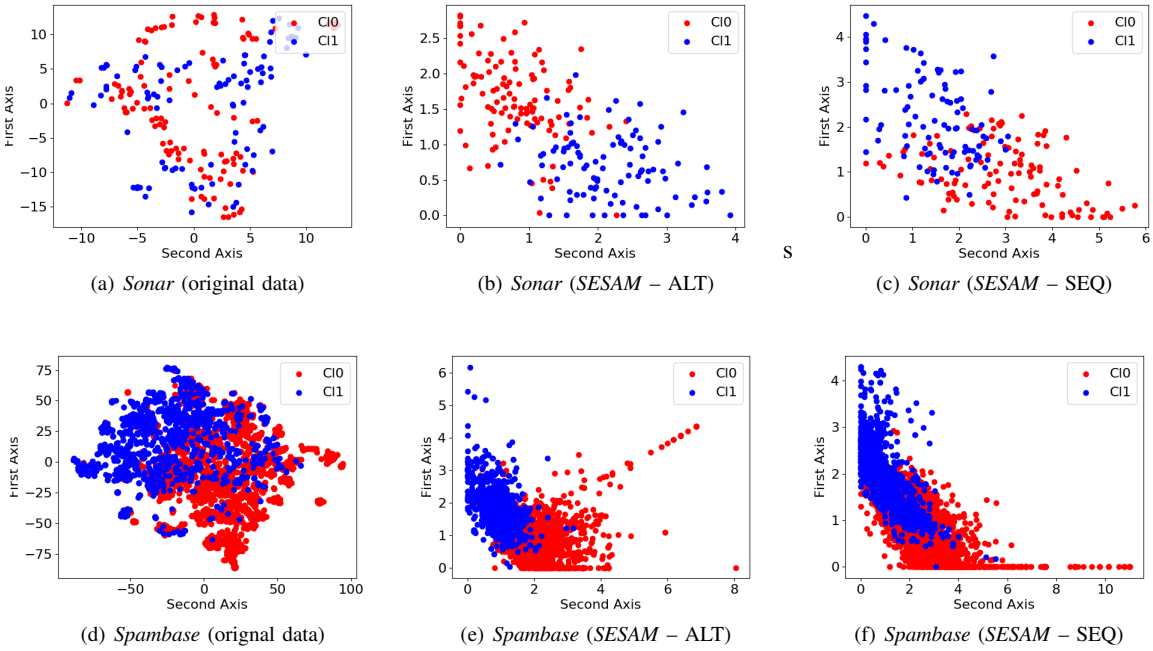
Fig. 6. Results (in terms of F-Measure) of the graph-based SSL algorithm, considering *SESAM* trained according to *ALT* and *SEQ* optimization procedures.

Inspecting the characteristics of the different datasets, we observe that *ALT* optimization performs better on datasets with high-dimensional embeddings while, conversely, *SEQ* is more effective when low-dimensional embeddings are generated. A possible explanation is that the alternate optimization struggles to resume both the original data as well as the discriminating information when the embedding dimension is too small (as in *Spambase* and *Waveform*). In such cases, reconstructing the data via unsupervised learning and then fine-tuning the obtained representation via semi-supervised autoencoders, allows to produce higher quality representation for the transductive GBSSL task. On the other hand, when the embedding dimension is bigger (as in *Coil20*, *GARD*, and *Botswana*), the expressiveness of the induced representation is such that the embedding contains, at the same time, the information to reconstruct the data as well as the information to discriminate among the different classes.

D. Visual Inspection of the new learnt representation

To better figure out why our method performs well, we visually inspect the new data representation. Figure 7 shows the two-dimensional projections of the original data versus the new data representation learnt by *SESAM* considering a number of labeled examples per class equal to 50 and both alternating (ALT) and sequential (SEQ) optimization schemes on *Sonar* (Figure 7(a), Figure 7(b) and Figure 7(c)) and *Spambase* (Figure 7(d), Figure 7(e) and Figure 7(f)) datasets. The two dimensional representation is obtained via the t-distributed stochastic neighbor embedding (*TSNE*) approach [22]. In both datasets, we can visually observe that the representation provided by *SESAM* induces a better discrimination (among classes) compared to the one associated to the original data. The native data representation does not allow to separate points belonging to different classes and this fact has a negative impact on distance based classifiers such as GBSSL approaches. Conversely, due to the fact that our approach exploits some (very limited) knowledge about labeled examples, it is able to reduce class overlapping and stretch the manifold on which examples are represented, thus increasing class separability.

The visual inspection is coherent with the behavior in terms of F-Measure we reported in previous discussions: the visual separability induced by a method is directly proportional to the



(a) *Sonar* (original data) (b) *Sonar* (*SESAM* – ALT) (c) *Sonar* (*SESAM* – SEQ)

(d) *Spambase* (orignal data) (e) *Spambase* (*SESAM* – ALT) (f) *Spambase* (*SESAM* – SEQ)

Fig. 7. Visual projection of the original data and the embeddings learnt by *SESAM* with alternating optimization (ALT) and sequential optimization (SEQ) on *Sonar* and *Spambase* with 50 Labeled examples per class.

quality of classification (in terms of F-Measure) obtained by the GBSSL approach. Moreover, the embeddings computed with the two alternative optimization strategies for *SESAM* exhibit similar separability between the two classes.

V. Conclusions

In this work, we have introduced *SESAM*, a strategy that integrates knowledge about labeled samples in the GBSSL pipeline in its early stages. Our approach leverages an ensemble of diverse semi-supervised autoencoders to build the new data representation. Diversity is induced by varying the structure (layer size) of each neural model. The experiments carried out on different benchmarks have demonstrated the capacity of our knowledge-aware embedding strategy to provide an effective data manifold for the label propagation process. We remind that, in our work, we have used *SESAM* in conjunction with a classifier working in a transductive setting. As future work, we plan to evaluate the ability of our framework to work in an inductive setting where the classification model is trained to classify samples not seen in the learning phase.

References

[1] I. Suzuki and K. Hara, "Centered knn graph for semi-supervised learning," in *SIGIR*. ACM, 2017, pp. 857–860.

[2] Y. Yamaguchi, C. Faloutsos, and H. Kitagawa, "CAMLP: confidence-aware modulated label propagation," in *SDM*, 2016, pp. 513–521.

[3] ——, "SocNL: Bayesian label propagation with confidence," in *PAKDD*, ser. Lecture Notes in Computer Science, vol. 9077. Springer, 2015, pp. 633–645.

[4] X. Wu, Z. Li, A. M. So, J. Wright, and S. Chang, "Learning with partially absorbing random walks," in *NIPS*, 2012, pp. 3086–3094.

[5] M. G. Quiles, L. Zhao, F. A. Breve, and A. Rocha, "Label propagation through neuronal synchrony," in *IJCNN*, 2010, pp. 1–8.

[6] W. Haiyan, Y. Haomin, L. Xueming, and R. Haijun, "Semi-Supervised Autoencoder: A Joint Approach of Representation and Classification," in *CICN*. IEEE, 2015, pp. 1424–1430.

[7] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *NIPS*, 2015, pp. 3546–3554.

[8] A. Gogna and A. Majumdar, "Semi supervised autoencoder," in *ICONIP*, 2016, pp. 82–89.

[9] J. Chen, S. Sathe, C. C. Aggarwal, and D. S. Turaga, "Outlier detection with autoencoder ensembles," in *SDM*, 2017, pp. 90–98.

[10] D. Ienco and R. G. Pensa, "Semi-supervised clustering with multiresolution autoencoders," in *IJCNN*, 2018, pp. 1–8.

[11] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *NIPS*, 2003, pp. 321–328.

[12] P. P. Talukdar and K. Crammer, "New regularized algorithms for transductive learning," in *ECML-PKDD*, 2009, pp. 442–457.

[13] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010, pp. 807–814.

[14] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[15] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *[NIPS]*, 1989, pp. 396–404.

[16] D. Dua and C. Graff, "UCI machine learning repository," 2019. [Online]. Available: http://archive.ics.uci.edu/ml

[17] R. Interdonato, D. Ienco, R. Gaetano, and K. Ose, "Duplo: A dual view point deep learning architecture for time series classification," *ISPRS J. Photogramm. Remote. Sens*, vol. 149, pp. 91–104, 2019.

[18] J. Ham, Y. Chen, M. M. Crawford, and J. Ghosh, "Investigation of the random forest framework for classification of hyperspectral data," *IEEE Trans. Geoscience and Remote Sensing*, vol. 43, no. 3, pp. 492–501, 2005.

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

[20] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[21] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[22] L. van der Maaten and G. Hinton, "Visualizing Data Using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.