ESA*: A Generic Framework for Semi-supervised Inductive Learning

Highlights

ESA*: A Generic Framework for Semi-supervised Inductive Learning

Shuyi Yang, Dino Ienco, Roberto Esposito, Ruggero G. Pensa

- A new inductive framework for graph-based semi-supervised classification is proposed.
- The proposed framework combines semi-supervised autoencoders and graph-based pseudo-labeling.
- Two variants based on confidence-aware label propagation and graph attention networks are proposed.
- The framework outperforms state-of-the-art competitors on data with very small amounts of labeled examples.
- The framework is generic as it is designed to work on data of any kind.

ESA*: A Generic Framework for Semi-supervised Inductive Learning

Shuyi Yang[a,b], Dino Ienco[c], Roberto Esposito[a], Ruggero G. Pensa[a,*]

[a] *University of Turin – Computer Science Department*
C.so Svizzera, 185 – I-10149 Torino, ITALY
[b] *Intesa Sanpaolo, Turin, Italy*
[c] *INRAE, UMR TETIS, Montpellier, France*

Abstract

Semi-supervised learning is crucial in many applications where accessing class labels is unaffordable or costly. The most promising approaches are graph-based but they are transductive and they do not provide a generalized model working on inductive scenarios. To address this problem, we propose a generic framework, ESA*, for inductive semi-supervised learning based on three components: an ensemble of semi-supervised autoencoders providing a new data representation that leverages the knowledge supplied by the reduced amount of available labels; a graph-based step that helps augmenting the training set with pseudo-labeled instances and, finally, a classifier trained with labeled and pseudo-labeled instances. Additionally, we also introduce two variants of our framework adopting different graph-based pseudo-labeling strategies: the first, ESA$^{\text{LP}}$, is based on a confidence-aware label propagation algorithm, while the second, ESA$^{\text{GAT}}$, on a graph convolutional attention network. The experimental results show that our framework outperforms state-of-the-art inductive semi-supervised methods.

Keywords: semi-supervised learning, graph-based algorithms, inductive methods

*Corresponding author

Email addresses: shuyi.yang@unito.it (Shuyi Yang), dino.ienco@inrae.fr (Dino Ienco), roberto.esposito@unito.it (Roberto Esposito), ruggero.pensa@unito.it (Ruggero G. Pensa)

1. Introduction

Prediction is one of the most important outcomes of any machine learning algorithm. It supports many time-consuming and tedious tasks once performed by humans and, although classification performances are not yet (always) comparable to humans', especially in noisy domains [1], it is one of the most important tasks deployed throughout the industry. Prediction accuracy strongly depends on the amounts and quality of labeled instances and, unfortunately, labeling is a cost-intensive manual activity requiring time, money, and expertise. Often, labeling (or annotation) tasks are outsourced to external companies (e.g., Amazon Mechanical Turk) that recruit users on the web for doing the job. However, in some sensitive areas, such as X-ray images interpretation [2], experts cannot be replaced by the wisdom of the crowd. Hence, labeling often turns out to be unaffordable for many organizations and, consequently, only small amounts of labeled instances are available for training. This is a major issue, especially for models requiring large amounts of training data such those implementing deep learning architectures.

Semi-supervised learning aims at mitigating the above-mentioned problem by leveraging the so-called *smoothness* and *cluster* assumptions: if two data instances are close to each other or belong to the same cluster in the input distribution, then they are likely to belong to the same class [3]. Instead of using the few labeled instances to train a classifier, the idea is to propagate the information to other "close" and unlabeled data instances. If labels are of good quality, and clusters are well separated, semi-supervised approaches usually outperform their supervised counterparts.

Among the different algorithmic solutions proposed in literature, graph-based models constitute one of the main families of semi-supervised techniques [4]. Graph-based methods leverage the manifold assumption: the graphs, typically nearest neighbor graphs built upon the local similarity between data points, provide a lower-dimensional representation of the high-dimensional input data. Graph-based semi-supervised learning algorithms typically involve two steps: in the first one, a nearest neighbor graph is constructed using all data points to capture the manifold of the data. Classification is then performed by propagating the information from labeled to unlabeled samples along the edges of the graph.

Unfortunately, graph-based methods are transductive [4], i.e., they do not construct any classification model and the prediction is limited to exactly those data instances that are already available during the training phase.

Therefore, graph-based methods are unable to classify new data examples, unless they are trained again on the augmented dataset. A second limitation concerns the construction of the graph: in general, this phase is completely unsupervised even though, for some instances, labels are available. When the cluster assumption is not completely satisfied, this could lead to poor prediction results. Although some solutions exist [5, 6], they only work in transductive settings. In this paper, we present a novel graph-based semi-supervised framework, ESA*, that improves in the areas mentioned above: it takes into account the information carried out by labeled instances during the graph construction and is designed to work properly in inductive settings.

Our approach – sketched graphically in Fig. 1 – first constructs a new representation using a semi-supervised autoencoder that takes all labeled and unlabeled training data as input. The representation learnt by the semi-supervised autoencoder, for both labeled and unlabeled training data, are then processed by a graph-based semi-supervised algorithm that propagates the label information from labeled to unlabeled data instances. This procedure provides pseudo-labels for the set of unlabeled instances. To this purpose, we will experiment with two different approaches, leading to two variants of our framework: the first, ESA$^{\text{LP}}$, is based on a graph-based label propagation algorithm that exploits homophily and heterophily, as well as the confidence of the inference results [7]; the second one, ESA$^{\text{GAT}}$, exploits a graph convolutional neural network with masked self-attention layers [8]. All training instances (labeled and unlabeled with pseudo-labels) are then used to train a classification model, which can perform prediction for new unseen examples as well. We show that our approach outperforms state-of-the-art approaches (including ladder networks [9] and ICT [10]), even with extremely small amounts of labeled instances.

The remainder of the paper is organized as follows: a brief related literature review is reported in Section 2; the general framework is introduced in Section 3; the two variants of the label propagation steps are described in Section 3.1 and 3.2 respectively; the results of our experimental validation are discussed in Section 4; finally, we draw conclusions in Section 5.

2. Related work

Semi-supervised learning algorithms can be characterized, depending on whether they build a general model or not for the underlying data, between transductive and inductive methods.

Transductive methods are mostly based on graphs, with the (dis)similarity between nodes coded as the weight of the graph edges. In these methods, once the graph has been constructed, an inference method is applied to make predictions on unlabeled nodes. For instance, in Confidence-Aware Modulated Label Propagation (CAMLP) [7], an iterative algorithm computes the probability distribution of each node over the classes by combining the prior belief and prediction confidence: a higher number of signals from the neighborhood implies higher confidence. In addition, signals from the neighbors are adjusted to manage both homophily and heterophily networks. Other semi-supervised transductive algorithms that are either confidence-aware or can handle homophily and heterophily networks exist: for instance, belief propagation (BP) [11, 12] can handle homophily and heterophily networks but it does not include the confidence component, SocNL [13], DGR [14], TACO [15], and ReLISH [16] are confidence-aware but can handle only homophily networks.

Very different approaches have been proposed for inductive semi-supervised methods. Early research focused on wrapper [17, 18, 19, 20, 21, 22] and unsupervised [23] preprocessing methods [4]. Unsupervised preprocessing is often used to provide a better initialization for the training parameters, effectively moving the weights of a neural network closer to the convergence region. For example, in a deep belief network, multiple restricted Boltzmann machines are stacked and trained with unlabeled data [24], then an output layer is added to the network structure and the entire network is trained on labeled data.

Autoencoders (AE) [25], instead, are trained to extract the latent representation of each instance with the goal of feeding the learning algorithm with representations that are conducive to learning better classifiers. Even though an autoencoder is designed to perform an unsupervised task, it can be extended to include a supervised component: in a semi-supervised autoencoder (SSAE) [26] a further prediction layer is attached to the bottleneck layer. SSAE are then trained with a loss that combines a reconstruction component and a classification component. A significant number of other frameworks also propose to combine unsupervised and supervised components and this approach is neither totally new: in fact, semi-supervised extensions for support vector machines [27, 28], probabilistic models based on Gaussian Processes [29] and density regularization methods [30] have already been developed in the past. What these models have in common is that they all try to maximize the margin by relying on a low-density assumption (i.e.,

in these models, the decision boundary is assumed to lay through regions of the instance space with low-density data).

In recent years, a particular formulation of the cluster assumption, called smoothness assumption (stating that close elements of an instance space should have similar target variables) has led to the development of a new set of inductive semi-supervised learning algorithms based on the idea of perturbation: predictions of instances that differ only for a small perturbation noise should be similar. A neural network based on this principle is the ladder network (LN) [9]. Although, its structure has some similarities with the one of a SSAE, it differs from SSAE in the training process and the cost function. In ladder networks, the network tries simultaneously to reconstruct the input and to denoise the representations built at every layer of the network. In this model, then, the additional denoising autoencoders that are added layer-wise build models for the latent representations, which in turn, help the learning process in both supervised and semi-supervised settings.

Other perturbation-based approaches, instead of injecting the noise in the data, perturb the model itself [31, 32]. Another class of semi-supervised learning methods perturbs data by combining feature vectors linearly [33, 34]. In [10], Verma *et et al.* presents Interpolation Consistency Training (ICT), which adopts the mean-teacher [35] method in addition to feature vector mixup: during the training, the student network is optimized to get correct predictions on labeled samples (by reducing the supervised loss) and to preserve consistency over unlabeled instances. Its prediction on a linear combination of unlabeled samples is compared to the linear combination of predictions made by the teacher network on the same samples (via the consistency loss).

In our work, differently from the reported literature, we propose a framework to tackle the inductive graph-based semi-supervised classification task by combining an ensemble of semi-supervised autoencoders with a graph-based pseudo-labeling process in order to feed a final classifier both with originally labeled instances and pseudo-labeled ones. The pseudo-labeling process provides labels for unlabeled instances and it can be implemented with two different strategies: the first one is based on the confidence-aware label propagation [7], while the second one adopts graph attention networks [8] to perform convolution operations on graph nodes representing both labeled and unlabeled instances.

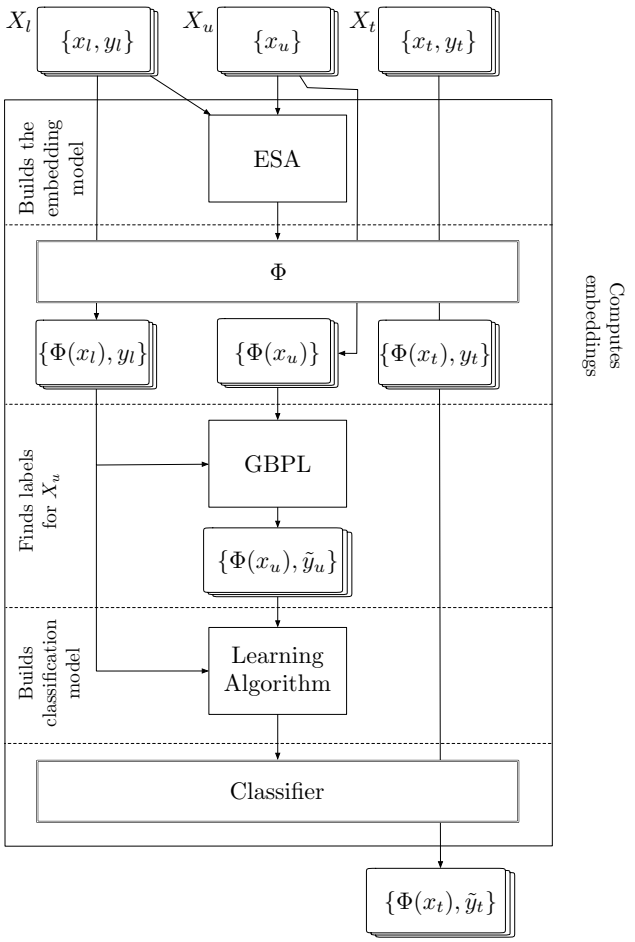


Figure 1: The proposed framework. The real labels of the test set instances are represented in the figure because they are used to assess the performance. The supervised classifier is trained only on labeled and pseudo-labeled instances.

3. Inductive graph-based semi-supervised learning

In a semi-supervised learning setting, in addition to labeled instances, unlabeled ones are introduced as part of available data during the training phase: let $X_l \in \mathbb{R}^{n_l \times f}$ be the matrix of n_l labeled samples each with f predictors and y_l be the corresponding labels, then a supplementary matrix $X_u \in \mathbb{R}^{n_u \times f}$ representing n_u unlabeled instances is also provided without the corresponding y_u labels. Generally, the number n_l of labeled instances is limited and much smaller than the number n_u of unlabeled instances.

Our framework aims to provide an inductive semi-supervised learning algorithm by leveraging graph-based semi-supervised learning in order to augment the amount of labeled instances to train a supervised classifier.

As shown in Figure 1, our framework consists of different parts: embedding computation, pseudo-labeling of unlabeled instances, and classification. In the embedding computation part, we train an ensemble of neural networks to extract a latent representation for each instance. These representations are used to build a graph over labeled and unlabeled instances so that a graph-based model can be employed to provide a pseudo-label for each unlabeled instance. Finally, labeled instances and pseudo-labeled ones are both used to train a supervised classification model.

In order to extract the data embeddings, an Ensemble of Semi-supervised Autoencoders (ESA) [26, 6] is trained on both labeled and unlabeled data.

An autoencoder (AE) is a neural network that employs a series of fully connected layers which constitute the encoder $E : \mathbb{R}^f \rightarrow \mathbb{R}^e$. An encoder transforms each instance in a e-dimensional latent representation. When the dimension of the representation layer is smaller than the dimension of the input layer, the representation layer is called bottleneck layer and the autoencoder is defined under-complete [36]. The bottleneck layer is attached to a second series of fully connected layers (decoder layers) in order to reconstruct the original instance $D : \mathbb{R}^e \rightarrow \mathbb{R}^f$. Given an instance x_r of X_l or X_u we expect that x_r is similar to its reconstructed version via the encoding/decoding process $D(E(x_r))$ and therefore the bottleneck layer provides a latent representation containing all relevant information needed to recover the input despite having much fewer dimensions. Besides the reconstruction task, if we add a classification layer $CL : \mathbb{R}^e \rightarrow \mathbb{R}^{|C|}$ to the bottleneck one, we can also train the network to learn a bottleneck representation tailored to the custom classification task we are undertaking. The output of the classification layer is a probability distribution over all possible labels in the label set C.

The loss function we use to learn the internal parameters of the SSAE is a combination of reconstruction and classification loss. More formally:

$$L_{\text{SSAE}} = L_{\text{AE}} + \lambda L_{\text{CL}} \tag{1}$$

where

$$L_{\text{AE}} = \frac{1}{n_l + n_u} \sum_{x_i \in X_l \cup X_u} ||x_i - D(E(x_i|\theta_E)|\theta_D)||^2, \tag{2}$$

Algorithm 1 SSAE - Training

Require: X_l: set of labeled instances, y_l: labels of X_l, X_u: set of unlabeled instances, f: number of input features, $size_{hidden}$: size of the hidden layers of SSAE, $size_{bottleneck}$: size of the bottleneck layer, $|C|$: number of classes, epochs: number of training epochs

Ensure: weights of the encoder, decoder and classification layers θ_E, θ_D, θ_{CL}

1: Initialize a SSAE with

- the encoder layers of sizes $[f, size_{hidden}, size_{bottleneck}]$
- decoder layers of sizes $[size_{hidden}, f]$
- the classification layer of size $|C|$

2: counter = 1
3: **while** counter $\leq$ epochs **do**
4: Update θ_E and θ_D by descending the gradient of the autoencoder loss $\nabla_{\theta_E,\theta_D} L_{\text{AE}}$
5: Update θ_E, θ_D and θ_{CL} by descending the gradient of the total loss $\nabla_{\theta_E,\theta_D,\theta_{CL}} \{L_{\text{AE}} + \lambda L_{\text{CL}}\}$
6: counter++
7: **end while**
8: **return** θ_E, θ_D, θ_{CL}

$$L_{\text{CL}} = -\frac{1}{n_l} \sum_{x_i \in X_l} \sum_{c=1}^{|C|} y_{lic} \cdot \log(CL(E(x_i|\theta_E)|\theta_{CL})_c), \tag{3}$$

and θ_E, θ_D and θ_{CL} are respectively the set of parameters of the encoder, decoder and classification layer, y_{lic} is the c-th element of the i-th row of y_l, $CL(\cdot)_c$ is the c-th element of the output vector of CL and λ is a parameter that controls the importance of the classification loss. See Algorithm 1 for the training details.

In our architecture (see Figure 2) the encoder has an input layer followed by other two hidden layers; the decoder has one hidden layer of the same size of the first hidden layer of the encoder and an output layer. The size of the input layer, the output layer and the classification layer are respectively fixed to f, f and $|C|$, while $size_{hidden}$ and $size_{bottleneck}$ (respectively, the size of the hidden layer and that of the bottleneck one) can be varied. In order to get diverse and multi-resolution representations, similarly as in [6], we train K independent SSAEs, each with the sizes of the layers extracted randomly from the intervals $\frac{f}{2} \leq size_{hidden} < f$ and $\frac{f}{4} \leq size_{hidden} < \frac{f}{2}$. Once the ensemble is trained we obtain the new representations $\Phi(X_l)$, $\Phi(X_u)$ of X_l and X_u by concatenating the embeddings of these K SSAEs:

$$\Phi(\cdot) = ||_{k=1}^{k=K} E_k(\cdot|\theta_{E_k}) \tag{4}$$

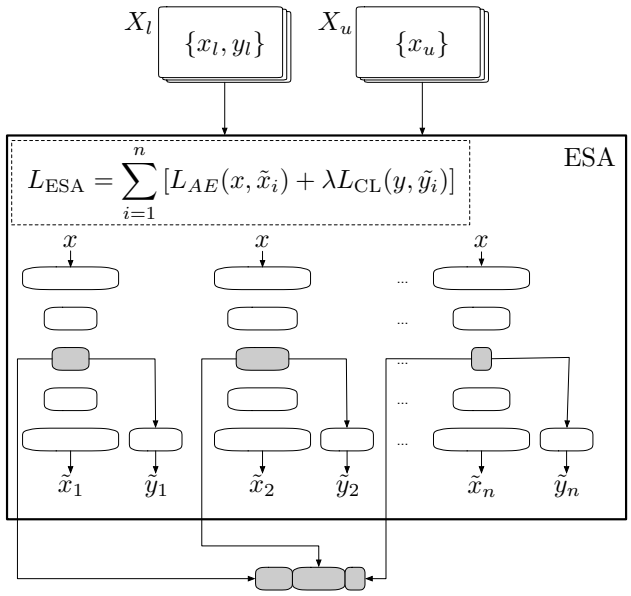


Figure 2: A graphical representation of ESA (ensemble of semi-supervised autoencoders) step of our framework.

Algorithm 2 ESA - Training

Require: X_l: set of labeled instances, y_l: labels of X_l, X_u: set of unlabeled instances, f: number of input features, $|C|$: number of classes, epochs: number of training epochs, K: number of SSAEs to be trained

Ensure: K SSAEs' encoder weights

1: $k = 1$
2: **while** $k \leq K$ **do**
3: sample an integer $size_{hidden}$ in the range $\left[\frac{f}{2}, f\right]$
4: sample an integer $size_{bottleneck}$ in the range $\left[\frac{f}{4}, \frac{f}{2}\right]$
5: initialize and train the k-th SSAE in order to learn its weights θ_{E_k}, θ_{D_k}, θ_{CL_k} (Algorithm 1)
6: $k++$
7: **end while**
8: **return** θ_{E_1}, θ_{E_2}, ..., θ_{E_K}

where || is the stack (concatenation) operator and E_k and θ_{E_k} are respectively the encoder of the k-th SSAE and its weights. See Algorithm 2 for the training details of ESA.

Given the latent representations, a kNN graph structure can be derived from the data points of $X_l \cup X_u$: embedding representations are nodes and

two of them can be considered connected if both of them belong to the top k nearest neighbors of each other, respectively.

At this point we perform graph-based pseudo-labeling (GBPL) to assign pseudo class labels to unlabeled instances. To this purpose, any graph-based semi-supervised learning algorithm (GBSSL) can be applied to infer the labels of the unlabeled portion of data $\tilde{y}_u$ by propagating the class information from the labeled data y_l over the graph constructed on the embeddings. Successively, a supervised classifier (SC) can be trained leveraging the union of the labeled data $(\Phi(X_l), y_l)$ with the pseudo-labeled one $(\Phi(X_u), \tilde{y}_u)$ as training set. In prediction, we first compute the latent representation of unseen data $\Phi(X_t)$ with the trained ESA, then we make predictions with the supervised classifier SC. It is worth pointing out that during the entire process, the transductive GBSSL process is used only during the training phase to provide pseudo-labels of the unlabeled data (as in wrapper methods) in order to help the supervised classifier to generalize better. Therefore, our approach, hereinafter referred as $\texttt{ESA}^\star$, is inductive[1]. See Algorithm 3 and Algorithm 4 for the training and prediction details of the overall framework. In the next two sections, we present two variants adopting different strategies to perform pseudo-labeling based on different graph-based semi-supervised learning approaches.

3.1. Pseudo-labeling based on confidence-aware label propagation

In this section we introduce the first variant of our framework for semi-supervised learning (see Figure 3). The adopted strategy consists in instantiating the graph-based pseudo-labeling (GBPL) part with a confidence-aware label propagation algorithm working on both homophily and heterophily networks [7]. In the following we provide the details of this strategy, which we name $\texttt{ESA}^{\text{LP}}$.

Given the adjacency matrix A, $\texttt{ESA}^{\text{LP}}$ computes the probability distribution over the classes as the solution of:

$$F_{i\varphi} = \frac{1}{Z_i}\left(y_{i\varphi} + \beta \sum_j A_{ij} s_{ji}(\varphi)\right) \tag{5}$$

[1]In the paper, we will use ESA to indicate the block consisting of the ensemble of semi-supervised autoencoders in our framework, while $\texttt{ESA}^\star$ stands for the overall semi-supervised learning approach.

Algorithm 3 `ESA*` - Training

Require: X_l: set of labeled instances, y_l: labels of X_l, X_u: set of unlabeled instances, f: number of input features, $|C|$: number of classes, epochs: number of training epochs, K: number of SSAEs to be trained, SC: a classifier to be trained

Ensure: K SSAEs' encoder weights, parameters of the classifier SC

1: Learn the weights θ_{E_1}, θ_{E_2}, ..., θ_{E_K} of ESA (Algorithm 2)
2: For each instance $x_i \in X_l \cup X_u$ compute the bottleneck representation of each SSAE $E_1(x_i|\theta_{E_1})$, $E_2(x_i|\theta_{E_2})$, ..., $E_K(x_i|\theta_{E_K})$
3: Stack the K embedding representations of each x_i by forming one unique array of real numbers $\Phi(x_i) = ||_{k=1}^{k=K} E_k(x_i|\theta_{E_k})$
4: For each $\Phi(x_i)$ compute its 20-nearest neighbors in the euclidean space
5: Construct an adjacency matrix A where the (i, j)-element is equal to 1 if and only if $\Phi(X_i)$ is in the top 20-nearest neighbors of $\Phi(X_j)$ and vice versa
6: Feed a GBSSL algorithm with A, $(\Phi(X_l), y_l)$ and $\Phi(X_u)$ in order to infer the labels of unlabeled instances $\tilde{y}_u$
7: Train a classifier $SC(\cdot|\theta_{SC})$ with the labeled embeddings $(\Phi(X_l), y_l)$ and the pseudo-labeled ones $(\Phi(X_u), \tilde{y}_u)$
8: **return** θ_{E_1}, θ_{E_2}, ..., θ_{E_K}, θ_{SC}

Algorithm 4 `ESA*` - Prediction

Require: θ_{E_1}, θ_{E_2}, ..., θ_{E_K}: learned weights of ESA, θ_{SC}: learned parameters of the classifier SC, x_{new}: a new instance

Ensure: class prediction of x_{new}

1: Given the weights of ESA (Algorithm 2), compute the embedding representation as $\Phi(x_{new}) = ||_{k=1}^{k=K} E_k(x_{new}|\theta_{E_k})$
2: **return** $SC(\Phi(X_{new})|\theta_{SC})$

where $F_{i\varphi}$ is the probability that i-th instance has label φ, Z_i is a normalization term, $y_{i\varphi}$ is the prior belief of i-th instance having label φ, $0 < \beta$ represents the importance of the neighborhood's influence, A_{ij} is the i, j entry of the adjacency matrix and $s_{ji}(\varphi)$ represents how intense the node j believes that the node i has class φ. More formally:

$$s_{ji}(\varphi) = \sum_{l} F_{jl} H_{l\varphi} \tag{6}$$

where H is the modulation matrix. If $H_{l\varphi}$ is low then class l has a low correlation with the class φ, on the contrary, if it is high these two classes have a strong correlation. On homophily networks, H is the identity matrix, while on heterophily networks it can be designed empirically. In our experiments we assume that the graph obtained by the embeddings of ESA is a homophily network.

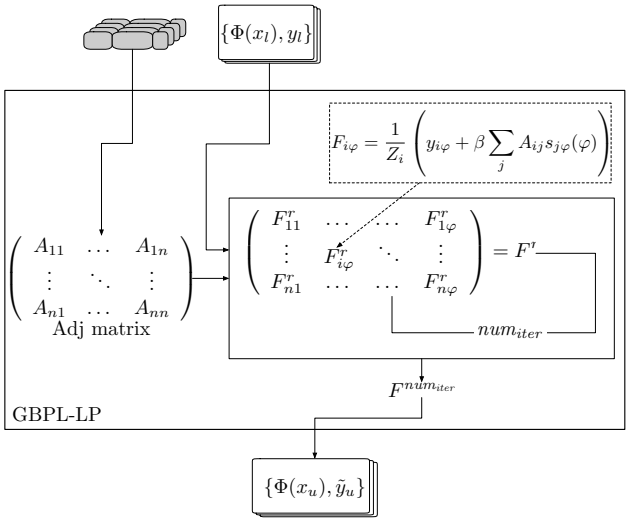


Figure 3: A graphical representation of the label propagation strategy of $\texttt{ESA}^{\text{LP}}$.

We can rewrite the Equation 5 in matrix form and in an iterative way:

$$F^{r+1} = Z^{-1}(Y + \beta A F^r H) \tag{7}$$

where $Z = I + \beta D$ and D is the node degrees diagonal matrix. Once obtained the adjacency matrix A of labeled and unlabeled instances, as described in the previous section, we initialize F^0 as a $(n_l + n_u) \times |C|$ matrix of zeros. Then we apply the Equation 7 num_{iter} times to obtain $F^{num_{iter}}$, which represents the probability distributions of the instances over the classes. From $F^{num_{iter}}$ we extract only the predictions of X_u and keep the original labels for X_l. They are then used to feed a classifier SC.

Since the algorithm of our generic framework is already detailed previously, in Algorithm 5 we only report the details of the label propagation phase (Step 6 of Algorithm 3).

3.2. Pseudo-labeling based on graph attention networks

For the second variant, we consider a completely different approach leveraging the convolution operation. Many recent works employ the spectral representation of the graph and perform convolution by computing the eigendecomposition of the Laplacian associated to the graph [37, 38, 39, 40]. However, since the Laplacian eigenbasis depends on the graph structure,

Algorithm 5 GBPL – LP

Require: X_l: the set of n_l labeled instances, y_l: labels of X_l, X_u: the set of n_u unlabeled instances, $|C|$: number of classes, H: modulation matrix, A: adjacency matrix, β: importance of the neighborhood's influence, D: the diagonal matrix of node degrees, num_{iter}: number of iterations

Ensure: probability distribution of instances of $X_l \cup X_u$ over the classes

1: Initialize F^0 as a $(n_l + n_u) \times |C|$ matrix with zeros as its entries
2: Initialize Y as a $(n_l + n_u) \times |C|$ matrix where $Y_{ij} = 1$ if the i-th instances of $X_l \cup X_u$ has j-th label, $Y_{ij} = 0$ if the i-th instances of $X_l \cup X_u$ is labeled but has not j-th label, $Y_{ij} = \frac{1}{|C|}$ if the i-th instances of $X_l \cup X_u$ is unlabeled
3: Initialize $Z = I + \beta D$
4: $r = 0$
5: **while** r $< num_{iter}$ **do**
6: compute $F^{r+1} = z^{-1}(Y + \beta A F^r H)$
7: $r++$
8: **end while**
9: **return** $F^{num_{iter}}$

these methods, once trained, can hardly be applied to graphs with different structures. On the other hand, non-spectral methods perform feature extraction from the neighborhood nodes while still maintaining shared weights [41, 42, 43, 44]. For instance, in [45] the weights are optimized to output similar representations for nearby nodes and once learned, they can be applied to graphs with different structures. In this models, each neighbor contributes equally to a node's embedded representation. Graph attention networks (GAT) [8], instead, overcome this limitation by adding a multi-head attention mechanism to each embedding layer so that nodes of the same neighborhood can assume different importance. Moreover, the weights of the feature transformation and the multi-head attention are shared and this method can be either transductive or inductive. Hence, GATs are good candidates for our pseudo-labeling process, as they are able to capture different levels of importance of features of neighborhood nodes in the kNN graph built upon the embeddings computed by ESA. We call this strategy $\texttt{ESA}^{\text{GAT}}$ and provide the details below (a graphical representation is given in Figure 4).

Given the set of nodes, each represented by a b-dimensional real numbers array obtained from the ESA embedding process or as a result of a previous convolutional layer, we can compute the self-attention on nodes as

$$e_{ij} = a(Wh_i, Wh_j|\aleph) \tag{8}$$

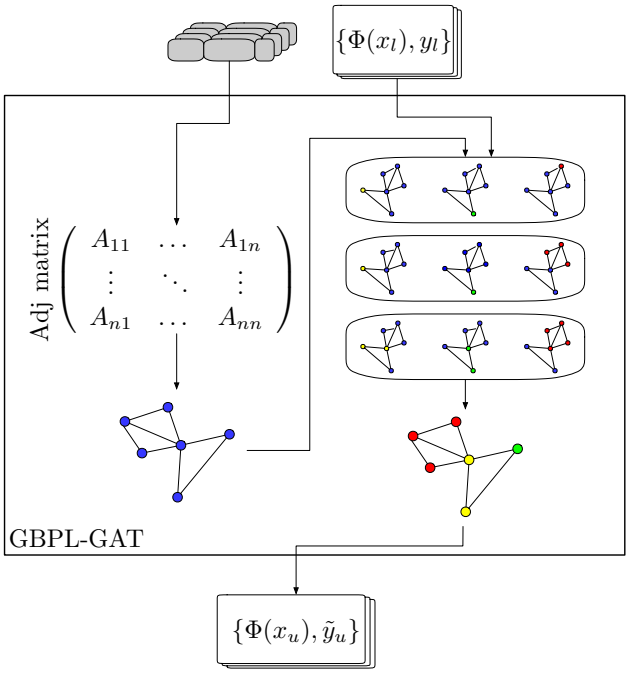


Figure 4: a graphical representation of the graph attention mechanism of $\texttt{ESA}^{\text{GAT}}$.

where $h_i, h_j \in \mathbb{R}^b$ are the embeddings of instance i and j, W is a $b' \times b$ shared linear transformation matrix, and $a : \mathbb{R}^{b'} \times \mathbb{R}^{b'} \to \mathbb{R}$ is the attentional mechanism consisting in a feedforward layer with weights $\aleph$ and LeakyReLU activation [46]. Each e_{ij} is computed only for connected nodes (masked attention) so that the graph structure is embedded into the coefficients. The attention coefficients are then normalized using the softmax function:

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{\iota \in N(i)} \exp e_{i\iota}} \tag{9}$$

with $N(i)$ representing the nodes connected to the node i. The new representation of the node i through the attention layer is then computed as

$$h_i^{'} = \sigma \left(\sum_{j \in N(i)} \alpha_{ij} W h_j \right) \tag{10}$$

with σ a non linear transformation. As in ESA, we can concatenate the outputs of different independent attention layers in order to employ a multi-head

Algorithm 6 GBPL – GAT

Require: X_l: the set of n_l labeled instances, y_l: labels of X_l, X_u: the set of n_u unlabeled instances, A: adjacency matrix, num_{heads}: number of heades, $h_1, ..., h_{n_l+n_u}$: ESA embeddings of labeled and unlabeled instances, *epochs*: number of epochs, a: attentional mechanism, σ, σ': non linear functions

Ensure: weights of the GAT

Initialize random matrices W^1, W^2, ..., $W^{num_{heads}}$, $\aleph^1$, $\aleph^2$, .., $\aleph^{num_{heads}}$, W', $\aleph'$
$counter = 1$
while $counter \leq epochs$ **do**
  $\hbar = 1$
  while $\hbar \leq num_{heads}$ **do**
    For each couple of instances (i, j) compute $e_{ij}^{\hbar} = a(W^{\hbar}h_i, W^{\hbar}h_j|\aleph^{\hbar})$
    For each couple of instances (i, j) compute the attention coefficients $\alpha ij^{\hbar} = \frac{\exp e_{ij}^{\hbar}}{\sum_{\iota \in N(i)} \exp e_{i\iota}^{\hbar}}$
    For each instance i compute its new representation $h_i'^{\hbar} = \sigma\left(\sum_{j \in N(i)} \alpha_{ij} W h_j\right)$
    $\hbar++$
  end while
  For each instance i concatenate its representations $h_i' = ||_{\hbar=1}^{k=num_{heads}} h_i'^{\hbar}$
  For each couple of instances (i, j) compute $e_{ij}' = a(W'h_i, W'h_j|\aleph')$
  For each couple of instances (i, j) compute the attention coefficients $\alpha_{ij}' = \frac{\exp e_{ij}'}{\sum_{\iota \in N(i)} \exp e_{i\iota}'}$
  For each instance i compute its new representation $h_i'' = \sigma'\left(\sum_{j \in N(i)} \alpha_{ij}' W' h_j'\right)$
  Compute the negative log-likehood loss for the labeled instances $L(h_i'', y_l)$
  Update W^1, W^2, ..., $W^{num_{heads}}$, $\aleph^1$, $\aleph^2$, ..., $\aleph^{numheads}$, W', $\aleph'$ by descending the gradient ∇L
end while
return W^1, W^2, ..., $W^{num_{heads}}$, $\aleph^1$, $\aleph^2$, ..., $\aleph^{num_{heads}}$, W', $\aleph'$

attention mechanism. To do that, once the ESA embeddings are obtained, we can apply the Equation 10 multiple times: from the point of view of the neural network structure, the mechanism is realized by adding additional convolutional layers, each with its own weights to be trained and the number of nodes of the last layer should be equal to the number of classes. See Algorithm 6 for the training details of a 2-layers GAT.

Even though there are some similarities between the attention mechanism in $\texttt{ESA}^{\text{GAT}}$ and the modulation matrix in $\texttt{ESA}^{\text{LP}}$, it is worth noting that while $\texttt{ESA}^{\text{GAT}}$ operates on the latent representation of the nodes, $\texttt{ESA}^{\text{LP}}$ regards the relationships among classes.

4. Experiments

To assess the behavior of our framework under different settings and configurations, we conduct four experiments.

In the first experiment, we compare the two versions of our framework: $\texttt{ESA}^{\text{LP}}$ and $\texttt{ESA}^{\text{GAT}}$. Our objective is twofold: on the one hand, we want to measure the impact of varying the different components of our framework, on the other hand, we aim at identifying the best combination(s). To this purpose, both $\texttt{ESA}^{\text{LP}}$ and $\texttt{ESA}^{\text{GAT}}$ variants of our framework are combined with four different classifiers: random forest (RF), multilayer perceptron (MLP), support vector machine (SVM) and discriminative ridge machine (DRM [47]). We name all possible combinations as follows: $\texttt{ESA}^{\text{LP}}$+RF, $\texttt{ESA}^{\text{LP}}$+MLP, $\texttt{ESA}^{\text{LP}}$+SVM, $\texttt{ESA}^{\text{LP}}$+DRM, $\texttt{ESA}^{\text{GAT}}$+RF, $\texttt{ESA}^{\text{GAT}}$+MLP, $\texttt{ESA}^{\text{GAT}}$+SVM and $\texttt{ESA}^{\text{GAT}}$+DRM.

In the second experiment, we compare the two best performing configurations of our framework to four well-known supervised methods (RF, MLP, SVM and DRM) and two recent state-of-the-art semi-supervised approaches: interpolation consistency training (ICT) [10] and ladder networks (LN) [9]. In our experiments, ICT is coupled with fully connected layers, instead of convolutional layers, to deal with tabular shaped dataset, as we do not specifically target image datasets. Our objective is to understand both the strengths and the limitations of our framework and to study under which conditions it outperforms (or is outperformed by) the other approaches.

In the third experiment, we conduct an ablation study for inspecting the contribution of the different components of our framework on the prediction accuracy. We compare the performances of three different inductive semi-supervised methods: i) a semi-supervised autoencoder (SSAE), ii) a semi-supervised autoencoder ensemble coupled with a classifier (ESA+RF), iii) our complete framework (including the graph-based semi-supervised component).

In the last experiment, we compare the two best performing configurations of our framework to two state-of-the-art transductive semi-supervised approaches: structured graph learning with multiple kernel (SGMK) [48] and robust graph construction (RGC) [49].

The remainder of the section is organized as follows: we first present the datasets used in our experiments (Section 4.1); then, in Section 4.2 we give the details of our experimental protocol; finally, we report and discuss the results of our four experiments in Section 4.3.

Dataset	# instances	# classes	# features
ANTIVIRUS [54]	373	2	513
COIL20 [53]	1440	20	576
FMNIST [52]	70000	10	784
LANDSAT [54]	6435	6	36
MADELON [54]	2600	2	500
MALWARE [54]	6248	2	1087
MNIST [51]	70000	10	784
PARKINSON [54]	756	2	753
SONAR [54]	208	2	60
SPAMBASE [54]	4601	2	57
USPS [50]	9298	10	256
WAVEFORM [54]	5000	3	40
WINE [54]	178	3	12

Table 1: List of datasets used during our experiments. This table contains the number of instances and the number of classes of each of them.

4.1. Datasets

As shown in Table 1, thirteen publicly available real-world classification datasets, encompassing a wide variety of application scenarios, have been considered in our experiments. They exhibits different sizes (from 178 to 70 000 instances) and dimensionality (from 12 to 1087 features). The datasets include 4 well-known image datasets (USPS [50], MNIST [51], FMNIST [52], COIL20 [53]). However, to be fair when comparing with competing approaches, we do not consider any convolutional filter and handle the images as numeric vectors.

4.2. Experiments settings

Each dataset is randomly split into three parts: labeled instances (p% of the dataset), unlabeled instances ((70$-p$)% of the dataset), and test instances (30% of the dataset). The random split is stratified so that each dataset maintains the same proportion of labels as in the original datasets. Every supervised model (RF, MLP, SVM, DRM) is trained on labeled instances only and evaluated on test instances, while all semi-supervised models are trained both on labeled instances and unlabeled ones and evaluated on the test instances.

During the experiments we vary the percentage p of labeled instances to study how the performances change when the portion of labeled instances increases in both supervised and semi-supervised models. More in details, p takes one of the following values: 0.1, 0.5, 1, 2, 3, 4 or 5. Therefore, in the most adverse situation, the percentage of labeled instances is only 0.1% of the entire dataset while the remaining 69.9% is unlabeled. In any case, we consider at least one labeled instance per class.

To obtain more robust performance indicators, for each combination of dataset, percentage p and model, we evaluate 25 different random splits as described above and then take the average performances. The model is re-trained for each of the 25 different splits and new predictions are made on every different test set. For any given percentage p and random split, all models are trained on the same labeled/unlabeled set, and compared on the same test set. As performance index, we consider the micro-averaged F1-score computed on the test set.

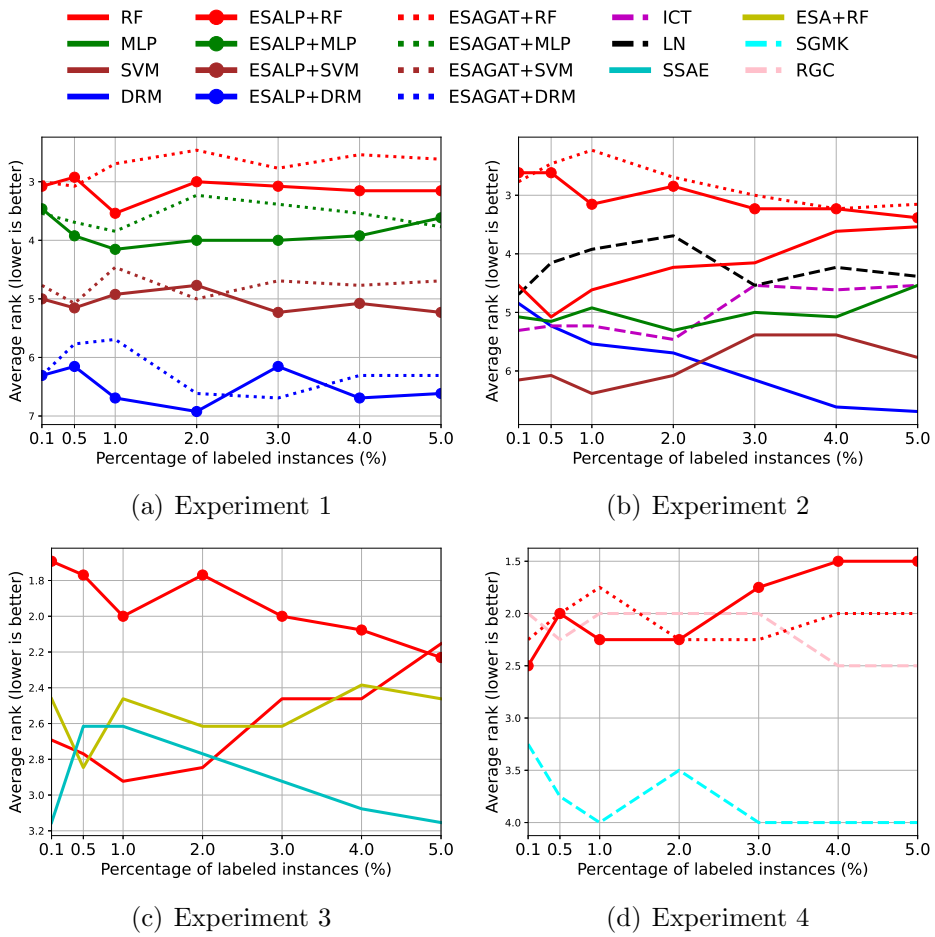
In Table 2 we report the structures and the hyper-parameters of the models used in our experiments. We have not performed a grid search to find the best combination of hyper-parameters for each model since this would be infeasible for the huge number of experiments required. Instead, when available, we used the default values reported in the original papers of models. When defaults values were unavailable, we made an educated guess based on the considerations found in the papers introducing the algorithms. It is worth pointing out that, in any case, the performances are always measured on an independent test set.

4.3. Results and discussion

In all evaluations of the experiments, we compute detailed performance results for each dataset (Figures 6, 7, 8, 9) and a summary of the results (Figure 5). The latter is obtained as follows: for a given percentage p of labeled instances, we compute the average ranks across all the dataset for each algorithm, according to the micro-averaged F1 score, and then we plot them for increasing values of $p\%$ (lower ranks are plotted higher in the pictures and mean better performances). This allows us to obtain an overall picture of the relative performances of all competitors considered in our study. In the following, we present and discuss the results for each experiment.

Model	**Structure / Hyperparameters**
RF	- default values.
MLP	- 2 hidden layers of sizes 64 and 32; - default values for the rest.
SVM	- **Kernel:** Nystroem approximation of an RBF kernel with gamma = 0.2, and number of components = 300 (as in the example given by the authors), - **Optimization:** Stochastic Gradient Descent; - default values for the rest.
DRM	- the DRM algorithm based on PPA (DRM-PPA [47]) is used in our experiments; - linear DRM-PPA is used for the datasets MNIST and FMNIST; - alpha = 0.1; - beta = 0.1
SSAE	- 2 hidden layers of sizes 64 and 32; - autoencoder batch size = 64; - semi-supervised autoencoder batch size = 8; - epochs = 100
ESA*	10 SSAEs each one having: - the first hidden layer of size in the interval $\left[\frac{f}{2}, f\right]$ - the second hidden layer of size in the interval $\left[\frac{f}{4}, \frac{f}{4}\right]$; these intervals are reduced to $\left[\frac{f}{5}, \frac{f}{2.5}\right]$ and $\left[\frac{f}{10}, \frac{f}{5}\right]$ for the datasets MNIST and FMNIST.
CAMLP	- default values.
GAT	- 2 layers of size 8; - number of heads = 3; - epochs = 100; - default values for the rest.
LN	- 4 hidden layers of sizes 128, 64, 32 and 32; - noise standard deviation = 0.3 - denoising cost = [1000, 10, 0.1, 0.1, 0.1, 0.1] - epochs = 100 - default values for the rest.
ICT	- mixup consistency = 10; - consistency rampup starts = 0; - consistency rampup ends = 100; - mixup sup alpha = 0.2; mixup usup alpha=0.2; - mixup hidden = False; - num mix layer = 2 - in the base model convolutional layers are substituted by full dense layers of sizes 64 and 32; - default values for the rest.
SGMK	- gamma = 1.0; - default values for the rest.
RGC	- beta = 0.1; - mu = 16; - k = 5; - r = 0.5; - default values for the rest.

Table 2: The structures and the hyperparameters of the models used in the experiments. The hyper-parameters are maintained unchanged for the models used in different parts of our experiments: for instance, RF, which is a standalone classifier, is also used as part of ESA$^{\mathrm{LP}}$+RF and ESA$^{\mathrm{GAT}}$+RF.



(a) Experiment 1

(b) Experiment 2

(c) Experiment 3

(d) Experiment 4

Figure 5: Average rank according to micro-averaged F1-measure in the experiments.

4.3.1. Experiment 1: analysis of the variants

This experiment aims to study different variants of our framework and select two of them as reference for further analysis (Experiments 2 and 3). To this purpose, we compare both $\texttt{ESA}^{\text{LP}}$ and $\texttt{ESA}^{\text{GAT}}$ variants with a different choice of final classifier (RF, MLP, SVM and DRM). The detailed results are provided in Figure 6 while a summary is presented in Figure 5(a). It can be noticed that, in general, when both variants are coupled with RF, on average, they outperform all other combinations, for all percentages of labeled samples. From the plots it is also evident that $\texttt{ESA}^{\text{GAT}}$+RF performs slightly better than $\texttt{ESA}^{\text{LP}}$+RF when $p \geq 1$. Apart from the variants with DRM

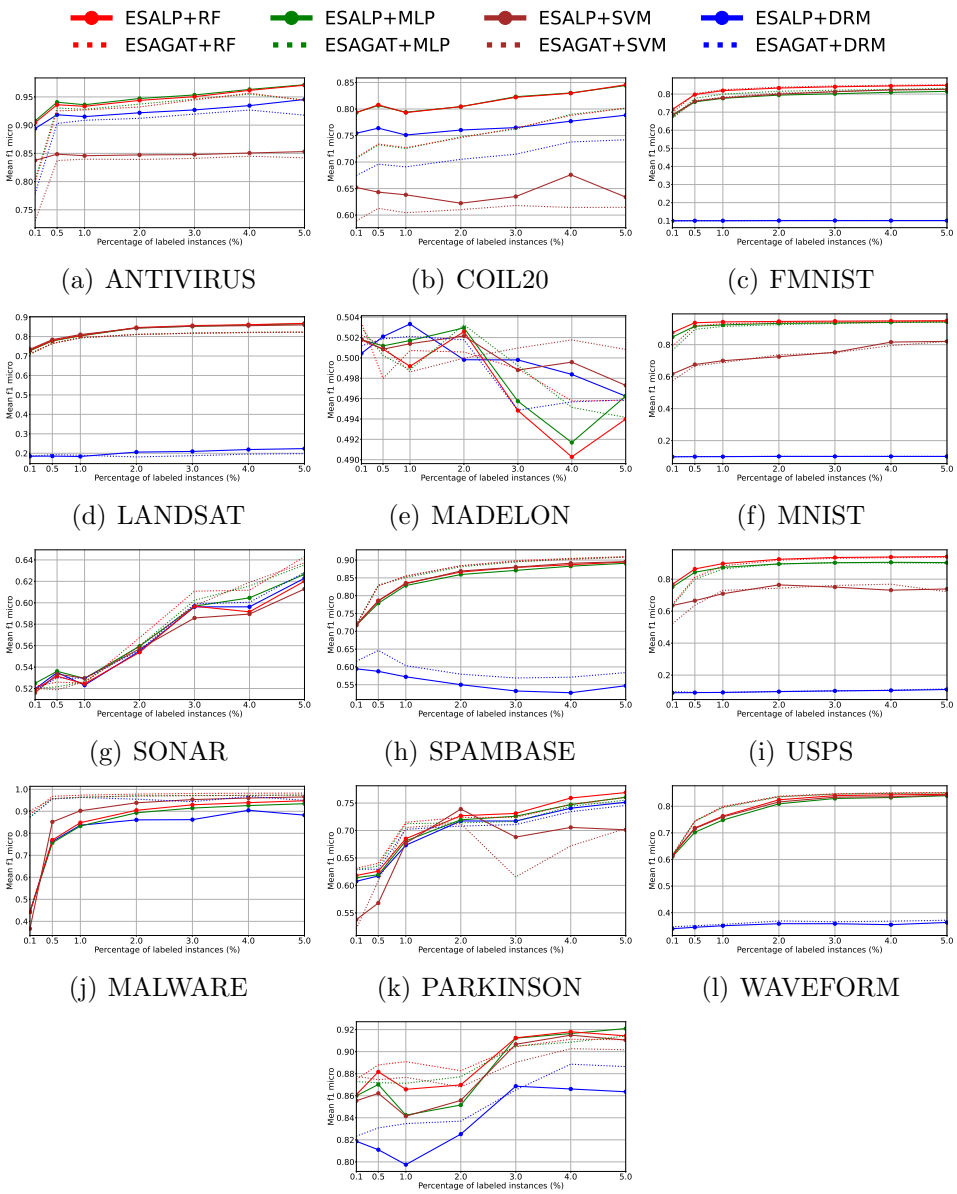
(which perform poorly on some datasets, probably due to their particularities), it is worth noting that, the overall behavior is similar for the remaining algorithms and all datasets (see Figure 6) except for small datasets (SONAR and WINE) and for COIL20 (the dataset with the largest number of classes). This first batch of experiments suggests that $\texttt{ESA}^{\text{LP}}$+RF and $\texttt{ESA}^{\text{GAT}}$+RF are the best combination for our technique on the chosen datasets, and we shall use these two models for the comparative analysis in the next experiments.

4.3.2. Experiment 2: comparative analysis

In this analysis the objective is to compare our framework to other supervised and semi-supervised methods. From the obtained results (Figure 5(b)), it emerges that, for every percentage of labeled examples, on average, the two variants of our framework outperform all other methods, including the four fully supervised classifiers considered in this study (RF, MLP, SVM and DRM). It is worth noting that, in contrast, the two competing semi-supervised methods (*ICT* and *LN*) are not able to outperform the supervised competitors with the same consistency. The micro-averaged F1 score of *ICT* is below the one of RF, for any given value of p. Ladder networks (*LN*) are ranked third with less than 2% of labeled samples, but RF is still competitive w.r.t. *LN* despite the fact that it does not take advantage of unlabeled instances. From the Figure (7), we can point out that *LN* performs well on some datasets (ANTIVIRUS, LANDSAT, PARKINSON, MNIST, FMNIST, USPS), but achieves relatively low accuracy in all other cases. Furthermore, both *ICT* and *LN* are ineffective on COIL20: the reason for such deceiving performances could be in the high number of classes of such a dataset. In this case, our methods are the only two capable to exceed the 0.7 threshold of micro-averaged F1 score. Finally, it is worth pointing out that, not surprisingly, when the number of labeled instances increases, the differences between semi-supervised methods and fully supervised ones decrease.

4.3.3. Experiment 3: ablation study

With this evaluation we measure the impact of every individual component of our framework in an inductive setting. In this case, we focus on $\texttt{ESA}^{\text{LP}}$+RF (the variant that considers label propagation and random forest) and analyze RF (the supervised counterpart), ESA+RF (an ensemble of semi-supervised autoencoders coupled with RF), and a simple semi-supervised autoencoder. Figure 5(c) summarizes the results by presenting the average rank of each competitor. From this plot, it turns out that using



(a) ANTIVIRUS (b) COIL20 (c) FMNIST

(d) LANDSAT (e) MADELON (f) MNIST

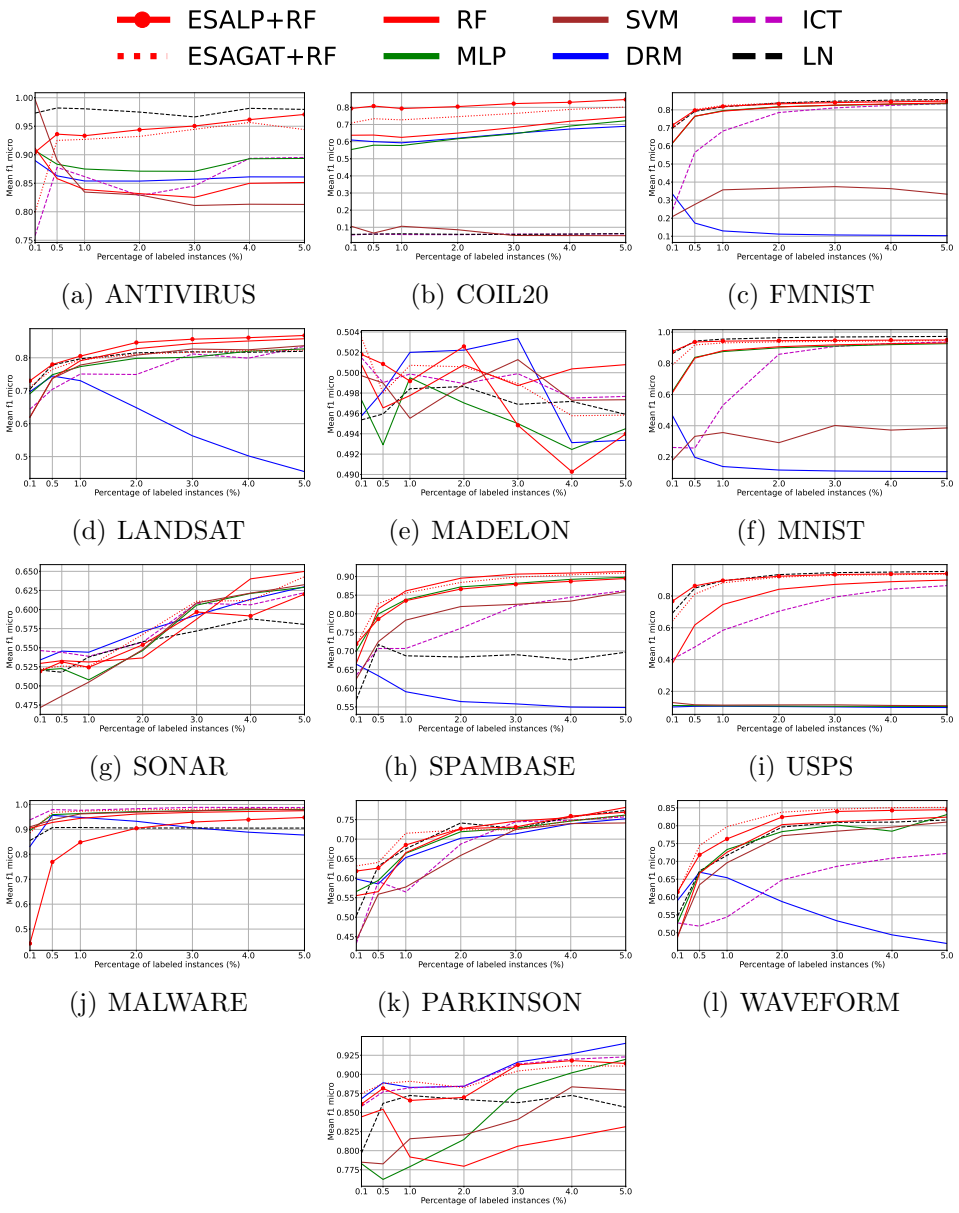
(g) SONAR (h) SPAMBASE (i) USPS

(j) MALWARE (k) PARKINSON (l) WAVEFORM

(m) WINE

Figure 6: Micro-averaged F1-measure for increasing percentage of labeled instances and for different datasets and variants of our algorithm (Experiment 1).

an ensemble of autoencoders helps improve the performances of a single semi-supervised autoencoder, but not to a great extent. Indeed, ESA+RF and RF



(a) ANTIVIRUS (b) COIL20 (c) FMNIST

(d) LANDSAT (e) MADELON (f) MNIST

(g) SONAR (h) SPAMBASE (i) USPS

(j) MALWARE (k) PARKINSON (l) WAVEFORM

(m) WINE

Figure 7: Micro-averaged F1-measure for increasing percentage of labeled instances and for different datasets and competitors (Experiment 2).

have still comparable overall performances and only the overall framework (in this case, $\texttt{ESA}^{\text{LP}}$+RF) enable us to substantially outperform RF. Hence, providing pseudo-labels for unlabeled instances via the graph-based semi-supervised learning step is crucial. Notice that, in this study, we can not consider the graph-based semi-supervised learning step alone, since it is a transductive approach and it does not provide a model that can be deployed on the test data.

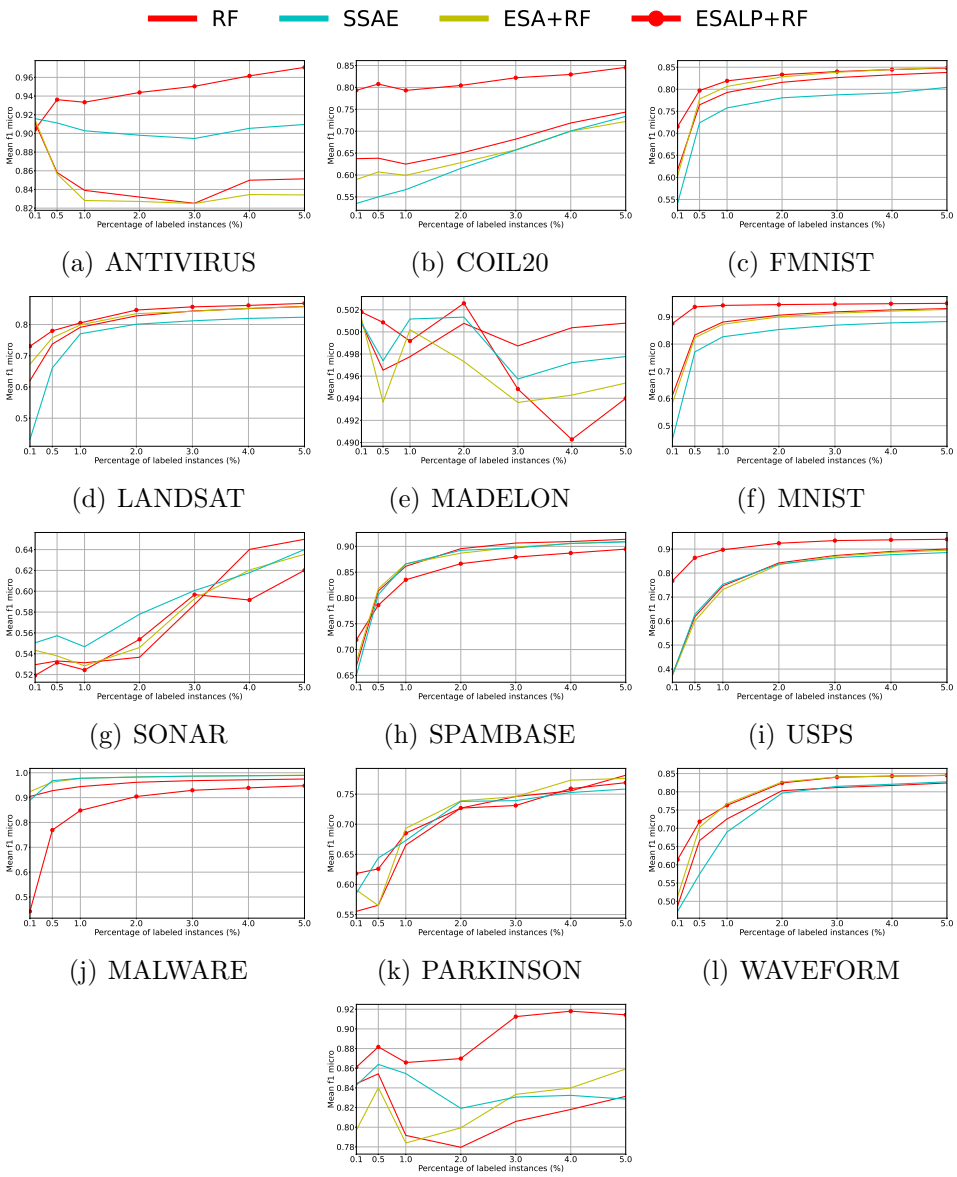
4.3.4. Experiment 4: comparative analysis w.r.t. transductive methods

In the last experiment the objective is to compare our inductive framework to recent semi-supervised transductive methods. It is worth pointing out that, while the transductive setting has potential liabilities in terms of applicability, it has advantages in the possibility of leveraging more information than its inductive counterparts (since it can leverage the test set distribution when propagating the labels). As we shall see, while our method is at a disadvantage here, it works quite well nonetheless and we believe that this comparison further clarifies the strength and the weaknesses of the proposed approach.

We compare the variants $\texttt{ESA}^{\text{LP}}$+RF and $\texttt{ESA}^{\text{GAT}}$+RF to the structured graph learning with multiple kernel (SGMK) [48] and to the robust graph construction (RGC) [49]. Due to computational limitations, the experiments are performed only on a subset of the available datasets, namely: ANTIVIRUS, SONAR, PARKINSON and WINE. From the results shown in Figure 9, our methods outperform the competing ones on ANTIVIRUS and PARKINSON. However, in SONAR and in WINE, RGC has better performances. Overall, with the Figure 5(d), we can conclude that, within our experimental settings, $\texttt{ESA}^{\text{LP}}$+RF and $\texttt{ESA}^{\text{GAT}}$+RF are able to reach competitive performances compared to SOTA transductive methods (i.e. RGC), and, in some cases, even outperform them (i.e. SGMK).

5. Conclusion

In this paper, we have presented a new inductive semi-supervised learning framework that take the most of two successful approaches: semi-supervised autoencoders and graph-based semi-supervised learning. While the former supports the generation of new data representations improved by labeled instances, the latter spread the label information to unlabeled instances in the new representation space. Thanks to an extensive experimental study, we



(a) ANTIVIRUS (b) COIL20 (c) FMNIST

(d) LANDSAT (e) MADELON (f) MNIST

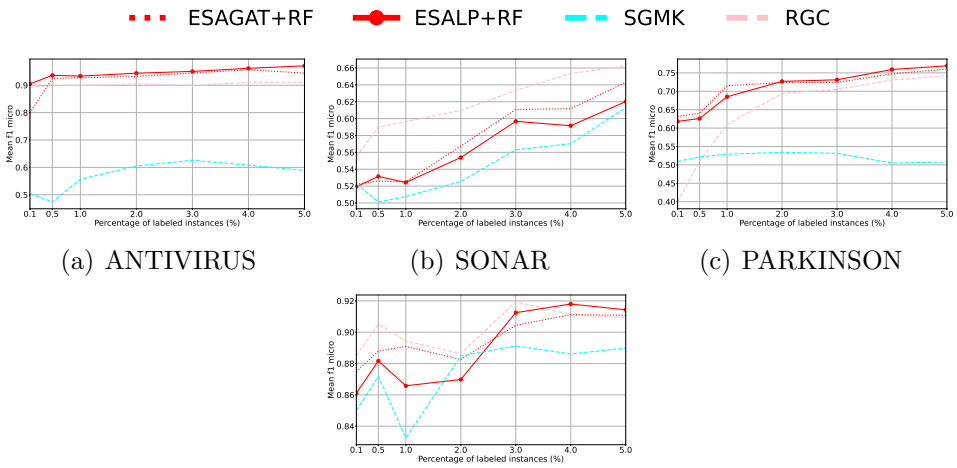
(g) SONAR (h) SPAMBASE (i) USPS

(j) MALWARE (k) PARKINSON (l) WAVEFORM

(m) WINE

Figure 8: Micro-averaged F1-measure for increasing percentage of labeled instances and for different datasets and different subcomponents of our algorithm (Experiment 3).

have shown that a classifier trained with both labeled instances and pseudo-labeled instances achieves better prediction accuracy than its supervised



(a) ANTIVIRUS (b) SONAR (c) PARKINSON

(d) WINE

Figure 9: Micro-averaged F1-measure for increasing percentage of labeled instances and for different datasets and different subcomponents of our algorithm (Experiment 4).

counterpart trained only on labeled instances, and also outperforms state-of-the-art semi-supervised competitors. Additionally, we have proposed two variants of our framework: the first one, based on a graph-based confidence-aware label propagation approach, is more effective with very few labeled instances; the second one, leveraging graph attention networks, outperforms the first one when more labeled instances are available.

As possible future work, we will introduce a variant of our framework especially tailored for image data by including convolutional layers in the first stages of the pipeline. Moreover, we will design an end-to-end inductive semi-supervised deep neural network process to strengthen the interaction among the different components of the proposed framework.

Author statement and acknowledgement

Shuyi Yang: Conceptualization, Methodology, Software, Investigation, Validation, Writing - Original draft **Dino Ienco:** Conceptualization, Methodology, Writing – Review & Editing. **Roberto Esposito:** Conceptualization, Methodology, Writing – Original draft, Writing – Review & Editing. **Ruggero G. Pensa:** Supervision, Conceptualization, Methodology, Writing – Original draft, Writing – Review & Editing.

The authors wish to thank Mattia Cerrato for stimulating discussions and its critical review of the paper.

References

[1] R. Geirhos, C. R. M. Temme, J. Rauber, H. H. Schütt, M. Bethge, F. A. Wichmann, Generalisation in humans and deep neural networks, in: S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Proceedings of the Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada, 2018, pp. 7549–7561.

[2] S. Rajaraman, J. Siegelman, P. O. Alderson, L. S. Folio, L. R. Folio, S. K. Antani, Iteratively pruned deep learning ensembles for COVID-19 detection in chest x-rays, IEEE Access 8 (2020) 115041–115050.

[3] O. Chapelle, B. Schölkopf, A. Zien (Eds.), Semi-Supervised Learning, The MIT Press, 2006.

[4] J. E. van Engelen, H. H. Hoos, A survey on semi-supervised learning, Mach. Learn. 109 (2020) 373–440.

[5] M. G. Quiles, L. Zhao, F. A. Breve, A. Rocha, Label propagation through neuronal synchrony, in: Proceedings of the International Joint Conference on Neural Networks, IJCNN 2010, Barcelona, Spain, 18-23 July, 2010, IEEE, 2010, pp. 1–8.

[6] D. Ienco, R. G. Pensa, Enhancing graph-based semisupervised learning via knowledge-aware data embedding, IEEE Transactions on Neural Networks and Learning Systems (2019) 1–7.

[7] Y. Yamaguchi, C. Faloutsos, H. Kitagawa, CAMLP: confidence-aware modulated label propagation, in: S. C. Venkatasubramanian, W. M. Jr. (Eds.), Proceedings of the International Conference on Data Mining, SIAM 2016, Miami, Florida, USA, May 5-7, 2016, SIAM, 2016, pp. 513–521.

[8] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, OpenReview.net, 2018.

[9] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, T. Raiko, Semi-supervised learning with ladder networks, in: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (Eds.), Proceedings of the Annual Conference on Neural Information Processing Systems 2015, NIPS 2015, Montreal, Quebec, Canada, December 7-12, 2015, 2015, pp. 3546–3554.

[10] V. Verma, A. Lamb, J. Kannala, Y. Bengio, D. Lopez-Paz, Interpolation consistency training for semi-supervised learning, in: S. Kraus (Ed.), Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, ijcai.org, 2019, pp. 3635–3641.

[11] W. G. S. Günnemann, D. Koutra, C. Faloutsos, Linearized and single-pass belief propagation, Proceedings of the VLDB Endowment 8 (2015).

[12] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI magazine 29 (2008) 93–93.

[13] Y. Yamaguchi, C. Faloutsos, H. Kitagawa, Socnl: Bayesian label propagation with confidence, in: T. Cao, E. Lim, Z. Zhou, T. B. Ho, D. W. Cheung, H. Motoda (Eds.), Proceedings of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2015, Ho Chi Minh City, Vietnam, May 19-22, 2015, volume 9077 of *Lecture Notes in Computer Science*, Springer, 2015, pp. 633–645.

[14] Y. Fang, B. P. Hsu, K. C. Chang, Confidence-aware graph regularization with heterogeneous pairwise features, in: W. R. Hersh, J. Callan, Y. Maarek, M. Sanderson (Eds.), Proceedings of the 35th International Conference on Research and Development in Information Retrieval, SIGIR 2012, Portland, OR, USA, August 12-16, 2012, ACM, 2012, pp. 951–960.

[15] M. Orbach, K. Crammer, Graph-based transduction with confidence, in: P. A. Flach, T. D. Bie, N. Cristianini (Eds.), Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, ECML PKDD 2012, Bristol, UK, September 24-28, 2012, volume 7524 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 323–338.

[16] C. Gong, D. Tao, K. Fu, J. Yang, Relish: Reliable label inference via smoothness hypothesis, in: C. E. Brodley, P. Stone (Eds.), Proceedings of the 28th Conference on Artificial Intelligence, AAAI 2014, Québec City, Québec, Canada, July 27 -31, 2014, AAAI Press, 2014, pp. 1840–1846.

[17] D. Yarowsky, Unsupervised word sense disambiguation rivaling supervised methods, in: H. Uszkoreit (Ed.), Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, ACL 1995, MIT, Cambridge, Massachusetts, USA, 26-30 June 1995, Morgan Kaufmann Publishers / ACL, 1995, pp. 189–196.

[18] F. d'Alché-Buc, Y. Grandvalet, C. Ambroise, Semi-supervised marginboost, in: T. G. Dietterich, S. Becker, Z. Ghahramani (Eds.), Proceedings the International Conference on Neural Information Processing Systems, NIPS 2001, Vancouver, British Columbia, Canada, December 3-8, 2001, MIT Press, 2001, pp. 553–560.

[19] K. P. Bennett, A. Demiriz, R. Maclin, Exploiting unlabeled data in ensemble methods, in: Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining, SIGKDD 2002, Edmonton, Alberta, Canada, July 23-26, 2002, ACM, 2002, pp. 289–296.

[20] P. K. Mallapragada, R. Jin, A. K. Jain, Y. Liu, Semiboost: Boosting for semi-supervised learning, IEEE transactions on pattern analysis and machine intelligence 31 (2008) 2000–2014.

[21] W. Wang, Z. Zhou, A new analysis of co-training, in: J. Fürnkranz, T. Joachims (Eds.), Proceedings of the 27th International Conference on Machine Learning ICML 2010, Haifa, Israel, June 21-24, 2010, Omnipress, 2010, pp. 1135–1142.

[22] Z.-H. Zhou, M. Li, Semi-supervised learning by disagreement, Knowledge and Information Systems 24 (2010) 415–439.

[23] A. B. Goldberg, X. Zhu, A. Singh, Z. Xu, R. D. Nowak, Multi-manifold semi-supervised learning, in: D. A. V. Dyk, M. Welling (Eds.), Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009, volume 5 of *JMLR Proceedings*, JMLR.org, 2009, pp. 169–176.

[24] G. E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural computation 18 (2006) 1527–1554.

[25] G. E. Hinton, R. S. Zemel, Autoencoders, minimum description length and helmholtz free energy, in: J. D. Cowan, G. Tesauro, J. Alspector (Eds.), Proceedings of the 7th Neural Information Processing Systems, NIPS 19993, Denver, Colorado, USA, 1993], Morgan Kaufmann, 1993, pp. 3–10.

[26] A. Gogna, A. Majumdar, Semi supervised autoencoder, in: A. Hirose, S. Ozawa, K. Doya, K. Ikeda, M. Lee, D. Liu (Eds.), Proceedings of the 23rd International Conference on Neural Information Processing, ICONIP 2016, Kyoto, Japan, October 16-21, 2016, volume 9948 of *Lecture Notes in Computer Science*, 2016, pp. 82–89.

[27] V. Vapnik, V. Vapnik, Statistical learning theory wiley, New York 1 (1998) 624.

[28] O. Chapelle, V. Sindhwani, S. S. Keerthi, Optimization techniques for semi-supervised support vector machines, Journal of Machine Learning Research 9 (2008) 203–233.

[29] N. D. Lawrence, M. I. Jordan, Semi-supervised learning via gaussian processes, in: Proceedings the International Conference on Neural Information Processing Systems, NIPS 2004, Vancouver, British Columbia, Canada, December 13-18, 2004], 2004, pp. 753–760.

[30] A. Corduneanu, T. S. Jaakkola, On information regularization, in: C. Meek, U. Kjærulff (Eds.), Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, UAI 2003, Acapulco, Mexico, August 7-10 2003, Morgan Kaufmann, 2003, pp. 151–158.

[31] P. Bachman, O. Alsharif, D. Precup, Learning with pseudo-ensembles, in: Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, K. Q. Weinberger (Eds.), Proceedings of the Annual Conference on Neural Information Processing Systems, NIPS 2014, Montreal, Quebec, Canada, December 8-13, 2014, 2014, pp. 3365–3373.

[32] S. Laine, T. Aila, Temporal ensembling for semi-supervised learning, arXiv preprint arXiv:1610.02242 (2016).

[33] H. Zhang, M. Cissé, Y. N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, in: Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, OpenReview.net, 2018.

[34] D. Berthelot, N. Carlini, I. J. Goodfellow, N. Papernot, A. Oliver, C. Raffel, Mixmatch: A holistic approach to semi-supervised learning, in: H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, R. Garnett (Eds.), Proceedings of the Annual Conference on Neural Information Processing Systems, NeurIPS 2019, Vancouver, BC, Canada, 8-14 December 2019, 2019, pp. 5050–5060.

[35] A. Tarvainen, H. Valpola, Weight-averaged consistency targets improve semi-supervised deep learning results, CoRR abs/1703.01780 (2017).

[36] I. Goodfellow, Y. Bengio, A. Courville, Deep learning book, MIT Press 521 (2016) 800.

[37] J. B. Estrach, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and deep locally connected networks on graphs, in: Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014, 2014.

[38] M. Henaff, J. Bruna, Y. LeCun, Deep convolutional networks on graph-structured data, arXiv preprint arXiv:1506.05163 (2015).

[39] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, R. Garnett (Eds.), Proceedings of the Annual Conference on Neural Information Processing Systems, NIPS 2016, Barcelona, Spain, December 5-10, 2016, 2016, pp. 3837–3845.

[40] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, OpenReview.net, 2017.

[41] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, R. P. Adams, Convolutional

networks on graphs for learning molecular fingerprints, in: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (Eds.), Proceedings of the Annual Conference on Neural Information Processing Systems, NIPS 2015, Montreal, Quebec, Canada, December 7-12, 2015, 2015, pp. 2224–2232.

[42] J. Atwood, D. Towsley, Diffusion-convolutional neural networks, in: D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, R. Garnett (Eds.), Proceedings of the Annual Conference on Neural Information Processing Systems, NIPS 2016, Barcelona, Spain, December 5-10, 2016, 2016, pp. 1993–2001.

[43] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, in: M. Balcan, K. Q. Weinberger (Eds.), Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, volume 48 of *JMLR Workshop and Conference Proceedings*, JMLR.org, 2016, pp. 2014–2023.

[44] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, M. M. Bronstein, Geometric deep learning on graphs and manifolds using mixture model cnns, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, IEEE Computer Society, 2017, pp. 5425–5434.

[45] W. L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, R. Garnett (Eds.), Proceedings of the Annual Conference on Neural Information Processing Systems, NIPS 2017, Long Beach, CA, USA, 4-9 December 2017, 2017, pp. 1024–1034.

[46] A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proceedings of the International Conference on Machine Learning, ICML 2013, Atlanta, USA, June 16-21, 2013, 2013.

[47] C. Peng, Q. Cheng, Discriminative ridge machine: A classifier for high-dimensional data or imbalanced data, IEEE Transactions on Neural Networks and Learning Systems (2020). Available online.

[48] Z. Kang, C. Peng, Q. Cheng, X. Liu, X. Peng, Z. Xu, L. Tian, Structured graph learning for clustering and semi-supervised classification, Pattern Recognition 110 (2021) 107627.

[49] Z. Kang, H. Pan, S. C. Hoi, Z. Xu, Robust graph learning from noisy data, IEEE transactions on cybernetics 50 (2019) 1833–1843.

[50] J. J. Hull, A database for handwritten text recognition research, IEEE Transactions on pattern analysis and machine intelligence 16 (1994) 550–554.

[51] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (1998) 2278–2324.

[52] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. arXiv:cs.LG/1708.07747.

[53] S. A. Nene, S. K. Nayar, H. Murase, et al., Columbia object image library (coil-100) (1996).

[54] D. Dua, C. Graff, UCI machine learning repository, 2017. URL: http://archive.ics.uci.edu/ml.