Label Propagation for Deep Semi-supervised Learning

Ahmet Iscen[1] Giorgos Tolias[1] Yannis Avrithis[2] Ondřej Chum[1]
[1]VRG, FEE, CTU in Prague [2]Univ Rennes, Inria, CNRS, IRISA

Abstract

Semi-supervised learning is becoming increasingly important because it can combine data carefully labeled by humans with abundant unlabeled data to train deep neural networks. Classic methods on semi-supervised learning that have focused on transductive learning have not been fully exploited in the inductive framework followed by modern deep learning. The same holds for the manifold assumption—that similar examples should get the same prediction. In this work, we employ a transductive label propagation method that is based on the manifold assumption to make predictions on the entire dataset and use these predictions to generate pseudo-labels for the unlabeled data and train a deep neural network. At the core of the transductive method lies a nearest neighbor graph of the dataset that we create based on the embeddings of the same network. Therefore our learning process iterates between these two steps. We improve performance on several datasets especially in the few labels regime and show that our work is complementary to current state of the art.

1. Introduction

Modern approaches to many computer vision problems exploit deep neural networks. These are popular for being very efficient and providing great performance at test time. The downside is a requirement of large amounts of training examples, which are labeled either by humans or automatically on proxy tasks.

Visual data are available in large quantities, however, data reliably annotated by humans are still very scarce. Obtaining large amounts of annotated training data for every single task is not only impractical, potentially costly, but it also turns out to be error prone. The low quality of crowdsourced annotation is a common motivation to minimize the need of annotation.

In the domain of *metric learning*, promising results have been recently achieved by *unsupervised* methods for either learning from scratch or fine-tuning a supervised network for domain adaptation, which devise proxy tasks for learning. These tasks exploit the distribution of data in the original space, for instance pairwise relations of training examples [42], relations between examples and cluster centroids [1], or considering the manifold structure of data [19]. Alternatively, in *self-supervised* learning, one can take advantage of additional information like spatial layout in images [5, 12] or temporal relation in videos [40, 28]; or mine for such information in unstructured data by *algorithmic supervision* using conventional methods [13, 30]. However, most such proxy tasks are inferior when directly compared to laboriously annotated data by humans.

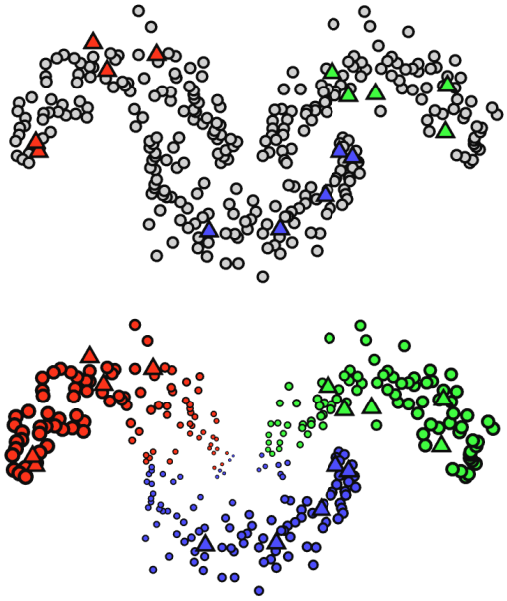

Figure 1. Label propagation on manifolds toy example. Triangles denote labeled, and circles un-labeled training data, respectively. Top: color-coded ground truth for labeled points, and gray color for unlabeled points. Bottom: color-coded pseudo-labels inferred by diffusion that are used to train the CNN. The size reflects the certainty of the pseudo-label prediction.

In classification, *semi-supervised* methods attempt to reduce the number of labeled examples, whereby the fully supervised performance on all data acts as an upper bound. In *transductive learning* [43, 45], label inference restricted to a given set of unlabeled examples is of interest. In *inductive learning*, the goal is generalization to new unseen data, while the original training data are discarded. This is achieved *e.g.* by combining classification loss on labeled

data with unsupervised objectives on all data, where the latter act as regularization [41, 38]. Or, an existing classifier can be used to assign pseudo-labels [24, 35], which is another form of algorithmic supervision. Using a powerful classifier trained on carefully annotated data can provide high-quality pseudo-labels, opening the door to learning from real unlabeled, large scale data. In such *omni-supervised* learning [31], the fully supervised performance on the labeled part is actually the lower bound. This only refreshes the interest in inductive semi-supervised methods.

In this paper, we use efficient transductive label propagation [43] to infer pseudo-labels for unlabeled data, which are used to train the classifier. Label propagation is a graph-based method, and in this work the graph is constructed exploiting the embeddings obtained by the classification network itself. Thus, the proposed method alternates between two steps. First, the network is trained from labeled and pseudo-labeled data. The second step uses the embeddings of the network trained in the previous step to construct a nearest neighbor graph. Label propagation is then used to infer pseudo-labels for unlabeled images, as well as a certainty score per image and per class. Training is performed on all data, using certainty-based weights.

We experimentally show on standard datasets that the proposed method outperforms other semi-supervised approaches. The less labeled data is available, the more pronounced the advantage of the proposed approach is.

2. Related work

The literature is rich in the problem of *semi-supervised learning* (SSL). The reader is advised to see [3] for an extensive overview. The same holds for SSL in image classification [10, 16, 4, 37]. In this section, we mostly restrict the discussion to approaches that use deep learning for SSL and perform the training on a large image collection with mini-batch optimization.

Prior work on semi-supervised deep learning for image classification is divided into two main categories. The first consists of methods, *e.g*. [15, 23, 34, 38], that add an *unsupervised loss* term (often called a regularizer) into the loss function. This term is applied to either all images or only the unlabeled ones. Methods in the second category, *e.g*. [24, 36], assign *pseudo-labels* to the unlabeled examples. The pseudo-labeled data are then used in training with a supervised loss, such as cross entropy. Both categories use a standard loss term that is trained with supervision from labeled images. A thorough evaluation of SSL deep image classification can be found in Miyato *et al*. [27].

Our contribution belongs to the second category, and is conceptually and implementation-wise orthogonal to the first. It is therefore straightforward to combine the proposed method with any method from the first category. We do combine it with [38] as shown in Section 5.

Unsupervised loss in deep SSL. Assuming that every training image, labeled or not, belongs to a single category, a natural requirement on the classifier is to make a confident prediction on the training set. This idea was formalized by Sajjadi *et al*. [35], where the regularizer is designed to minimize the entropy of the network output. Such a loss term is easily combined with other terms. A similar combination is performed for denoising auto-encoders that are applied on all images in an unsupervised manner [32].

A direction attracting a lot of attention is that of *consistency loss*, where two related cases, *e.g*. coming from two similar images, or made by two networks with related parameters, are encouraged to have similar network outputs. Sajjadi *et al*. [34] is the first, to our knowledge, to use a consistency loss between the outputs of a network on random perturbations of the same image. Laine and Aila [23] rather apply consistency between the output of the current network and the temporal average of outputs during training. The state-of-the-art *mean teacher* (MT) method [38] replaces output averaging by averaging of network parameters. Consistency loss is commonly measured by squared Euclidean distance. The Jensen-Shannon divergence is used instead by Qiao *et al*. [29], while complementarity of the two networks is enforced via adversarial examples. A similar idea is proposed by Miyato *et al*. [26].

Pseudo-labeling in deep SSL. Lee [24] uses the current network to infer pseudo-labels of unlabeled examples, by choosing the most confident class. These pseudo-labels are treated like human-provided labels in the cross entropy loss. Its impact is similar to that of entropy minimization [35]; in both cases the network is forced to have more confident predictions. The same principle is adopted by Shi *et al*. [36], where the authors further add contrastive loss to the consistency loss. Our method is different from all such prior work in that pseudo-labels are inferred by label propagation rather than network predictions.

Label propagation has been extensively used in a transductive setup (see chapter 11 [3]). Recently, Douze *et al*. [7] perform label propagation on a large image dataset with CNN descriptors for few shot learning. Unseen images are classified via online label propagation, which requires storing the entire dataset, while the network is trained in advance and descriptors are fixed. Our work is different in that we perform label propagation on the training set offline while training the network, such that inference is possible without accessing the original training set. *Learning by association* [17] can been seen as two steps of propagation on a constrained bi-partite graph between labeled and unlabeled examples. *Graph transduction game* (GTG) [9], a form of label propagation, has been used for pseudo-labels [8] as in our work, but in this case the network is pre-trained, the graph remains fixed and there is no weighting mechanism. We compare to this approach in Section 5.

3. Preliminaries

In this section we formulate the *semi-supervised learning* problem and then we discuss the classifier, different loss functions that are commonly used in prior work, and finally a transductive learning approach that our method is based on. In our experiments we use a *convolutional neural network* (CNN) to perform image classification, but this formulation applies to any network architecture in any domain.

Problem formulation. We assume a collection of n examples $X := (x_1, \ldots, x_l, x_{l+1}, \ldots, x_n)$ with $x_i \in \mathcal{X}$. The first l examples x_i for $i \in L := \{1, \ldots, l\}$, denoted by X_L, are labeled according to $Y_L := (y_1, \ldots, y_l)$ with $y_i \in C$, where $C := \{1, \ldots, c\}$ is a discrete label set for c classes. The remaining $u := n - l$ examples x_i for $i \in U := \{l+1, \ldots, n\}$, denoted by X_U, are unlabeled. The goal in SSL is to use all examples X and labels Y_L to train a classifier that maps previously unseen samples to class labels.

Classifier. The network takes an input example from $\mathcal{X}$ and produces a vector of class confidence scores. We denote it by $f_\theta : \mathcal{X} \to \mathbb{R}^c$, where θ are the network parameters. It is conceptually divided in two parts. The first is a feature extraction network $\phi_\theta : \mathcal{X} \to \mathbb{R}^d$ mapping the input to a feature vector, or descriptor. We denote the descriptor of the i-th example by $\mathbf{v}_i := \phi_\theta(x_i)$. The second typically consists of a *fully connected* (FC) layer applied on top of ϕ_θ and followed by softmax, producing a vector of *confidence scores*. Function f_θ is the mapping from input space directly to confidence scores. The output of the network for the i-th example is $f_\theta(x_i)$ and the *prediction* is the one of maximum confidence score

$$\hat{y}_i := \arg\max_j f_\theta(x_i)_j, \tag{1}$$

where subscript j denotes the j-th dimension of the vector.

Supervised loss. In supervised learning, the network is trained by minimizing a *supervised* loss term of the form

$$L_s(X_L, Y_L; \theta) := \sum_{i=1}^{l} \ell_s\left(f_\theta(x_i), y_i\right), \tag{2}$$

which applies only to labeled examples in X_L. Such term is part of the total loss when training a network in a semi-supervised setup [36, 38, 29]. A standard choice for the loss function ℓ_s in classification is *cross-entropy*, given by $\ell_s(\mathbf{s}, y) := -\log \mathbf{s}_y$ for $\mathbf{s} \in \mathbb{R}^c$ and $y \in C$.

Pseudo-labeling is the process of assigning a pseudo-label $\hat{y}_i$ to each example x_i for $i \in U$. Denoting by $\hat{Y}_U := (\hat{y}_{l+1}, \ldots, \hat{y}_n)$ the collection of pseudo-labels for X_U, the following additional *pseudo-label* loss term applies

$$L_p(X_U, \hat{Y}_U; \theta) := \sum_{i=l+1}^{n} \ell_s\left(f_\theta(x_i), \hat{y}_i\right), \tag{3}$$

where again ℓ_s is any supervised loss function like cross-entropy. An example is the approach proposed by Lee [24], who first train network f_θ with (2) and then assign pseudo-labels according to (1) for $i \in U$.

Unsupervised loss is another common alternative where the loss function applies to both labeled and unlabeled examples and encourages consistency under different transformations of the data or the network. The so-called *consistency loss* [36, 38, 36] is defined as

$$L_u(X; \theta) := \sum_{i=1}^{n} \ell_u(f_\theta(x_i), f_{\tilde{\theta}}(\tilde{x}_i)), \tag{4}$$

where $\tilde{x}_i$ refers to a different transformation of example x_i. Note that according to the standard practice of data augmentation, every forward pass of x_i during training is performed under some random transformation. Parameter set $\tilde{\theta}$ is either equal to θ or any other transformation of it, such as a moving average over the sequence of network updates [38]. A simple choice of ℓ_u is the squared Euclidean distance, *i.e.* $\ell_u(\mathbf{s}, \tilde{\mathbf{s}}) := ||\mathbf{s} - \tilde{\mathbf{s}})||^2$ for $\mathbf{s}, \tilde{\mathbf{s}} \in \mathbb{R}^c$, forcing the two outputs to be as close as possible.

Transductive learning solves a more specific problem. Instead of training a generic classifier able to classify new, yet unseen, examples, the goal is to use X and Y_L to infer labels for examples in X_U. In this work, we adopt the graph-based approach of Zhou *et al*. [43] for transductive learning by diffusion[1].

Diffusion for transductive learning [43]. Let $V = (\mathbf{v}_1, \ldots, \mathbf{v}_l, \mathbf{v}_{l+1}, \ldots, \mathbf{v}_n)$ be the descriptor set, where $\mathbf{v}_i$ corresponds to x_i as defined earlier. A symmetric *adjacency matrix* $W \in \mathbb{R}^{n \times n}$ with zero diagonal is constructed, whose elements w_{ij} are non-negative pairwise similarities between $\mathbf{v}_i$ and $\mathbf{v}_j$. Its symmetrically normalized counterpart is given by $\mathcal{W} = D^{-1/2} W D^{-1/2}$, where $D := \mathrm{diag}(W\mathbf{1}_n)$ is the *degree matrix* and $\mathbf{1}_n$ is the all-ones n-vector. A $n \times c$ *label matrix* Y is defined with elements

$$Y_{ij} := \begin{cases} 1, & \text{if } i \in L \wedge y_i = j \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

That is, the rows of Y corresponding to labeled examples are one-hot encoded labels and the rest are zero. Diffusion amounts to computing the $n \times c$ matrix

$$Z := (I - \alpha\mathcal{W})^{-1} Y, \tag{6}$$

where $\alpha \in [0, 1)$ is a parameter. Finally, the class prediction for an unlabeled example x_i is

$$\hat{y}_i := \arg\max_j z_{ij}, \tag{7}$$

where z_{ij} is the (i, j) element of matrix Z.

[1] We first present the original approach and discuss our design choices in the following section.

It is interesting to observe that matrix Z as defined by (6) is the minimizer of the following quadratic cost function

$$J(Z) := \frac{\alpha}{2} \sum_{i,j=1}^{n} w_{ij} \left\| \frac{\mathbf{z}_i}{\sqrt{d_{ii}}} - \frac{\mathbf{z}_j}{\sqrt{d_{jj}}} \right\|^2 + (1-\alpha) \left\| Y - Z \right\|_F^2 , \tag{8}$$

where $\mathbf{z}_i$ is the i-th row of matrix Z, d_{ii} is the i-th diagonal diagonal element of D and $\|\cdot\|_F$ is the Frobenius norm. The first term encourages *smoothness* such that nearby examples get the same predictions, while the second attempts to maintain predictions for the labeled examples [43].

4. Method

In the following, we begin by providing an overview of our approach. We then develop the main elements of our solution, put everything together in a concrete algorithm, and discuss how our approach is complementary to approaches using unsupervised loss for SSL [38, 36, 36]. Finally, we discuss the relation to prior work that encourages smoothness in deep networks.

Overview. We introduce a new iterative process for semi-supervised learning that can be summarized as follows. First, we construct a nearest neighbor graph and perform label propagation by transductive learning on the training set. Then, we estimate of a weight reflecting the uncertainty of label propagation for each unlabeled example. Finally, we inject the obtained labels into the network training process. These ideas are developed below, while a graphical overview of the proposed approach is shown in Figure 2.

Nearest neighbor graph. Given a network with parameters θ, we construct the descriptor set $V = (\mathbf{v}_1, \ldots, \mathbf{v}_l, \mathbf{v}_{l+1}, \ldots, \mathbf{v}_n)$, where $\mathbf{v}_i := \phi_\theta(x_i)$. A sparse *affinity matrix* $A \in \mathbb{R}^{n \times n}$ with elements

$$a_{ij} := \begin{cases} [\mathbf{v}_i^\top \mathbf{v}_j]_+^\gamma, & \text{if } i \neq j \wedge \mathbf{v}_i \in \text{NN}_k(\mathbf{v}_j) \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

is constructed, where NN_k denotes the set of k nearest neighbors in X, and γ is a parameter following recent work on manifold-based search [20]. Note that constructing the affinity matrix of the nearest neighbor graph is efficient even for large n [20], while constructing the full affinity matrix as in Zhou *et al*. is not tractable. Then, let $W := A + A^\top$, which is indeed a symmetric nonnegative adjacency matrix with zero diagonal.

Label propagation. Estimating matrix Z by (6) is impractical for large n because the inverse matrix $(I - \alpha\mathcal{W})^{-1}$ is not sparse. We rather use the the conjugate gradient (CG) method to solve linear system

$$(I - \alpha\mathcal{W})Z = Y, \tag{10}$$

which applies because matrix $(I - \alpha\mathcal{W})$ is positive-definite. This solution is known to be faster than the iterative solution of Zhou *et al*. [43], and has been used in semi-supervised learning [44], interactive image segmentation [14], image retrieval [20] and semantic image segmentation [2]. Finally, we infer the pseudo-labels $\hat{Y}_U = (\hat{y}_{l+1}, \ldots, \hat{y}_n)$, where $\hat{y}_i$ is given by (7).

Pseudo-label certainty and class balancing. Inferring pseudo-labels from matrix Z by hard assignment has two undesired effects: first, we define pseudo-labels on all unlabeled examples while clearly we do not have the same certainty for each example. Second, pseudo-labels may not be balanced over classes, which will impede learning.

To deal with the former issue we associate with each pseudo-label a weight reflecting the certainty of the prediction. We use *entropy*, as a measure of uncertainty, to assign weight ω_i to example x_i, defined by

$$\omega_i := 1 - \frac{H(\hat{\mathbf{z}}_i)}{\log(c)}, \tag{11}$$

where $\hat{Z}$ is the row-wise normalized counterpart of Z, *i.e.* $\hat{z}_{ij} = z_{ij} / \sum_k z_{ik}$, and function $H : \mathbb{R}^c \to \mathbb{R}$ is the entropy function. Weight ω_i is normalized in $[0, 1]$ because $\log(c)$ is the maximum possible entropy in $\mathbb{R}^c$.

To deal with the latter issue of class imbalance, we assign weight ζ_j to class j that is inversely proportional to class population, defined as $\zeta_j := (|L_j| + |U_j|)^{-1}$, where L_j (resp. U_j) are the examples labeled (resp. pseudo-labeled) as class j.

Given the above definitions of per-example and per-class weights, we associate the following *weighted loss* to the labeled and pseudo-labeled examples

$$\begin{aligned} L_w(X, Y_L, \hat{Y}_U; \theta) := & \sum_{i=1}^{l} \zeta_{y_i} \ell_s \left(f_\theta(x_i), y_i \right) \\ & + \sum_{i=l+1}^{n} \omega_i \zeta_{\hat{y}_i} \ell_s \left(f_\theta(x_i), \hat{y}_i \right) , \end{aligned} \tag{12}$$

which is the sum of weighted versions of L_s (2) and L_p (3). In contrast to (3), pseudo-labels originate in diffusion rather than network predictions.

A toy example showing the result of label propagation and the estimated weights is shown in Figure 3.

Iterative training. Given the above definitions of nearest neighbor graph definition, label propagation, example/class weighting and pseudo-label loss, we plug those components into an iterative learning process. We begin by randomly initializing the network parameters θ and we train the network for T epochs in a fully supervised manner on the l labeled examples X_L using the supervised loss term (2). The trained network then provides the starting point for the

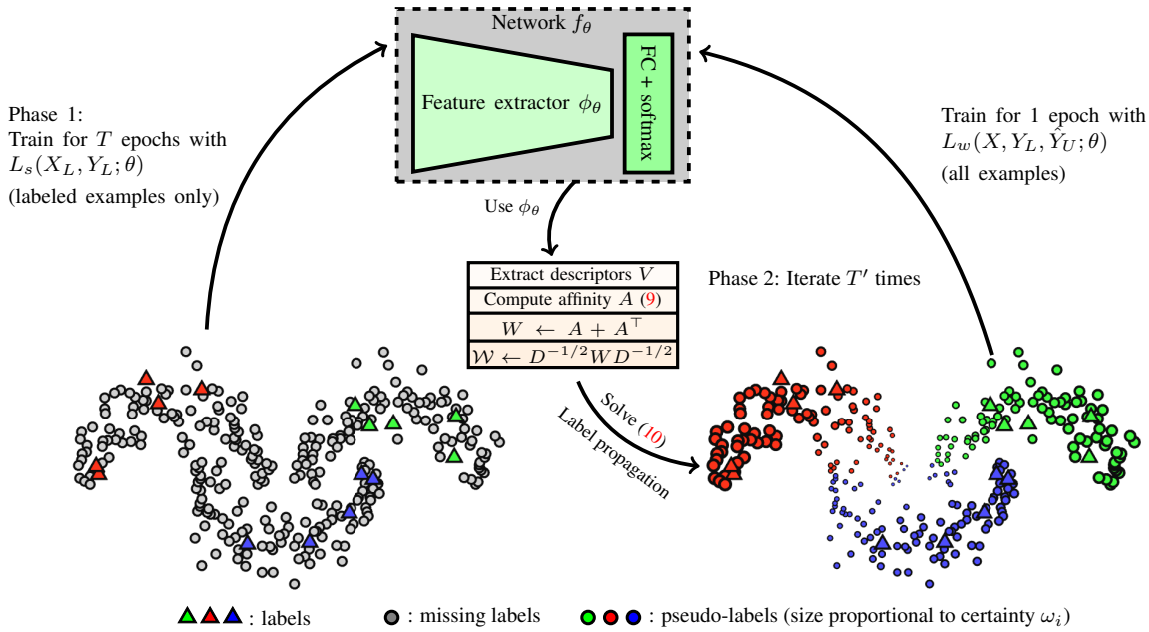

Figure 2. Overview of the proposed approach. Starting from a randomly initialized network, we first train it in a supervised fashion on the labeled examples. Then we initiate an iterative process where at each iteration we compute a nearest neighbor graph of the entire training set in the feature space of the current network, we propagate labels by transductive learning, and then we train the network on the entire training set, with true labels or pseudo-labels on the labeled or unlabeled examples respectively. The pseudo-labels are weighted per example and per class according to prediction certainty and inverse class population, respectively.

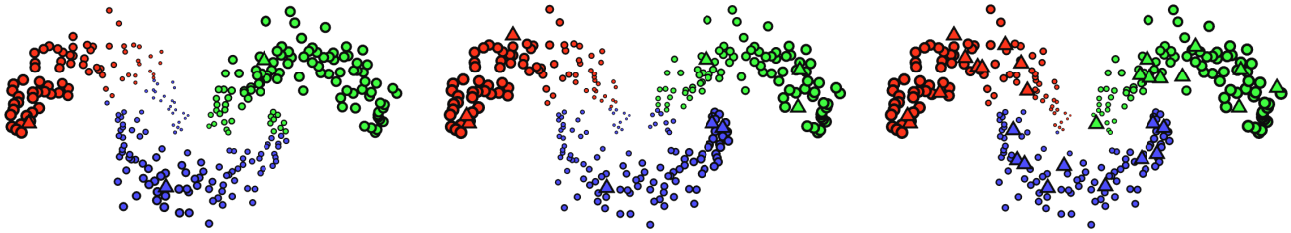

Figure 3. Toy example with 300 examples demonstrating label propagation for different number of labeled examples. Triangle markers correspond the labeled examples and circles to the unlabeled ones which are finally pseudo-labeled by label propagation. The class is color-coded and the size of the circles corresponds to weight ω_i. The true labels are the same as the example of Figure 1 (top).

following iterative process. First, we extract descriptors V on the entire training set X and compute nearest neighbors to construct the adjacency matrix W. Second, we perform label propagation by solving linear system (10) and assign pseudo-labels to unlabeled examples X_U by (7). Finally, we train the network for one epoch on the entire training set X using the weighted loss L_w (12). We repeat this iterative process for T' epochs. The above is summarized in Algorithm 1.

Procedure OPTIMIZE() refers to the mini-batch optimization of the corresponding loss term for one epoch, *i.e.* all examples are fed to the network once. More details about batch construction are given in the implementation details.

Combination with other approaches. Our contribution falls in the case of pseudo-label loss in the form of (3). It is orthogonal to approaches that use unsupervised loss, for instance (4), applied to both labeled and unlabeled examples. Combination of the two comes in a straightforward way by adding term (4) to the total loss optimized in lines 4 and 16 of Algorithm 1. This is exactly the way we combine the proposed approach with the state-of-the-art Mean-Teacher approach [38] in our experiments.

Discussion. In an inductive framework, if $\mathbf{z}_i/\sqrt{d_{ii}}$ is replaced by the network output $f_\theta(x_i)$ in the smoothness term of (8), then this becomes an unsupervised loss term, *e.g.* like (4), only now it encourages consistency between nearby example predictions. And indeed such solution is adopted *e.g.* by Weston *et al.* [41]. This is not very efficient because the adjacency matrix is typically sparse with non-zero-elements only on nearest neighbors, and then the gradient of the smoothness term will propagate from each example to its neighbors only at each iteration.

Algorithm 1 Label propagation for deep SSL

procedure LPDSSL(Training examples X, labels Y_L)
  $\theta \leftarrow$ initialize randomly
  for epoch $\in [1, \dots, T]$ **do**
    $\theta \leftarrow \text{OPTIMIZE}(L_s(X_L, Y_L; \theta))$ ▷ mini-batch optimization
  end for
  for epoch $\in [1, \dots, T']$ **do**
    for $i \in \{1, \dots, n\}$ **do** $\mathbf{v}_i \leftarrow \phi_\theta(x_i)$ ▷ extract descriptors
    for $(i, j) \in \{1, \dots, n\}^2$ **do** $a_{ij} \leftarrow$ affinity values (9)
    $W \leftarrow A + A^\top$ ▷ symmetric affinity
    $\mathcal{W} \leftarrow D^{-1/2} W D^{-1/2}$ ▷ symmetrically normalized affinity
    $Z \leftarrow$ solve (10) with CG ▷ diffusion
    for $(i, j) \in U \times C$ **do** $\hat{z}_{ij} \leftarrow z_{ij} / \sum_k z_{ik}$ ▷ normalize Z
    for $i \in U$ **do** $\hat{y}_i \leftarrow \arg\max_j \hat{z}_{ij}$ ▷ pseudo-label
    for $i \in U$ **do** $\omega_i \leftarrow$ certainty of $\hat{y}_i$ (11) ▷ pseudo-label weight
    for $j \in C$ **do** $\zeta_j \leftarrow (|L_j| + |U_j|)^{-1}$ ▷ class weight/balancing
    $\theta \leftarrow \text{OPTIMIZE}(L_w(X, Y_L, \hat{Y}_U; \theta))$ ▷ mini-batch optimization
  end for
end procedure

Our main idea therefore is that *instead of just encouraging nearby examples to get the same predictions, we encourage all examples to get predictions same as the ones we would get by transductive learning* according to the quadratic cost (8) and its solution Z (6). Computing Z is efficient because it is performed outside our main optimization process, *i.e.* it does not need iterating on mini-batches of data and backpropagating through the network. Then, given Z, the main optimization process drives all examples directly to that solution, as if they were all labeled.

5. Experiments

We present the datasets used in our experiments and the SSL setup that is followed. Then, we discuss the training details of our method and the methods reproduced for fair comparison. Finally, we perform experiments to show the impact of different components involved in the proposed method and to compare with the state of the art. All error rates reported are produced by our own implementation unless otherwise stated.

5.1. Datasets

We use three image classification datasets, namely CIFAR-10 [22], CIFAR-100 [22] and Mini-ImageNet [39]. Each dataset is used in an SSL setup where part of the training images are labeled and the rest are unlabeled. We evaluate the performance on an independent test set. Unless otherwise specified, error rate is reported in our experiments.

CIFAR-10. The training set consists of 50k images coming from 10 classes, while the test set consists of 10k images from the same 10 classes. All images have resolution 32×32. Evaluation is performed with 50, 100, 200, and 400 labeled images per classes, corresponding to $l = 500$, 1k, 2k, and 4k label images in total. We use the same random selection of labeled images that is used in Mean Teacher [38] when available (1k, 2k and 4k labels). The selection process is repeated 10 times, resulting in 10 different dataset splits for SSL on CIFAR 10. We follow the common practice which is to use each of them and report mean error and standard deviation.

CIFAR-100. Similarly to CIFAR-10, CIFAR-100 has 50k training and 10k test images of resolution 32×32, coming from 100 classes. We follow a protocol equivalent to the one of CIFAR-10. We evaluate with 40 and 100 labeled images per class, corresponding to 4k and 10k labeled images in total. There are 3 such dataset splits, mean error and standard deviation are reported.

Mini-ImageNet. We introduce an SSL evaluation setup for Mini-ImageNet [39] which is a subset of the well-known ImageNet [6] dataset and has been previously used for few-shot learning [11]. We use the train/test splits created in the work of Ravi and Larochelle [33]. It consists of 100 classes with 600 images per class, of resolution 84×84. We randomly assign 500 images from each class to the training set, and 100 images to the test set. The result is a train and test set of 50k and 10k images, respectively. We create three dataset splits for the case of 40 and 100 labeled images per class that correspond to 4k and 10k labeled images in total. Mean error and standard deviation over the three dataset splits are reported.

5.2. Training

We list the reproduced baselines, and provide training details per algorithm and dataset.

Implementation. We build our implementation on top of the publicly available Pytorch code for the Mean Teacher (MT) approach [38][2]. The fully supervised baseline and MT are reproduced identically as the original implementation. In all our experiments SGD optimization is used.

Networks. Experiments on CIFAR-10 and CIFAR-100 are performed with the "13-layer" network that is used in prior work [23, 38], while on Mini-ImageNet, Resnet-18 [18] is engaged. Both networks consist of a feature extractor ϕ_θ followed by an FC layer and softmax. We add an ℓ_2-normalization layer right after ϕ_θ (before the FC layer) providing unit-norm descriptors for the graph construction. The same choice is also adopted in the fully supervised baseline. One exception is all variants of MT as we observed that the ℓ_2-normalization layer slightly harms performance. We normalize images to have channel-wise zero mean and unit variance over the entire training set. Unlike prior work [38], we do not normalize the input images with ZCA, nor add Gaussian noise to the input layer, which result in worse performance according to our experiments.

Hyper-parameters and training choices are adapted from the MT method and implementation. These are fixed

[2] https://github.com/CuriousAI/mean-teacher/tree/master/pytorch

Pseudo-labeling	ω_i	ζ_j	CIFAR-10
Diffusion (7)			36.53 ± 1.42
		✓	36.17 ± 1.98
	✓		33.32 ± 1.53
	✓	✓	$\mathbf{32.40 \pm 1.80}$
GTG [8]	✓	✓	35.20 ± 2.23
Network (1)	✓	✓	35.17 ± 2.46

Table 1. Impact of weights ω_i, class weights ζ_j, and pseudo-labeling by diffusion prediction (7) or network prediction (1). Error rate is reported on CIFAR-10 with 500 labels.

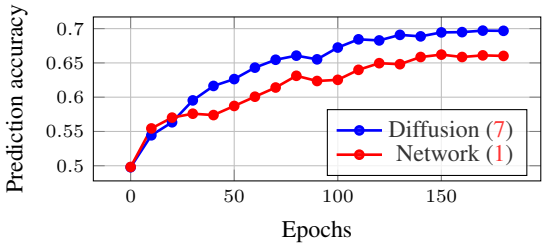

Figure 4. Accuracy of predicted pseudo-labels according to ground-truth on CIFAR-10 with 500 labeled images. Diffusion predictions (7) are compared against network predictions (1).

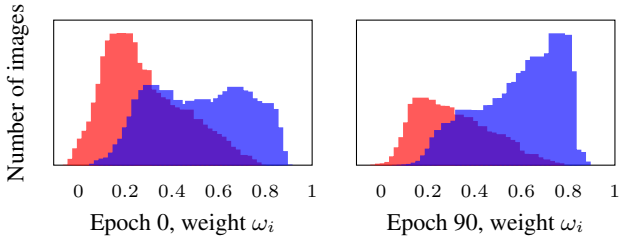

Figure 5. Distribution of weights ω_i for unlabeled images at epoch 0 (left) and epoch 90 (right) during the training of CIFAR-10 with 500 labels. Correct pseudo-labels according to ground-truth are shown in blue and incorrect in red.

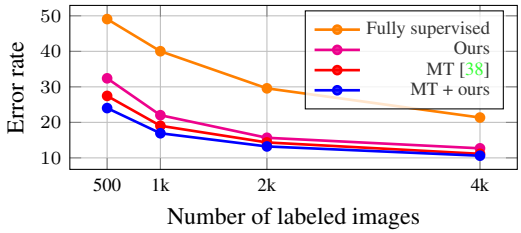

Figure 6. Error rate versus number of labeled images on CIFAR-10 using different methods.

for all approaches (re)produced by this work. The training is performed for 180 epochs in total. Initial learning rate l_0 is decayed with cosine annealing [25] so that it would have reached zero after 210 epochs, while $l_0 = 0.05$ on CIFAR-10, and $l_0 = 0.2$ on CIFAR-100 and Mini-ImageNet. Random data augmentation is performed by 4×4 random translations [38] followed by horizontal flip in CIFAR-10 and CIFAR-100. On Mini-ImageNet, each image is randomly rotated by 10 degrees before random horizontal flip. Batch

Figure 7. Examples of incorrectly pseudo-labeled images with highest ω_i in CIFAR-10. Predicted class and ω_i are shown below each image.

size is 100 for CIFAR-10 and 128 for CIFAR-100 and Mini-ImageNet. All other learning parameters remain unchanged from MT implementation.

The fully supervised approach corresponds to training with (2) and labeled images only. MT uses the additional dual output trick with coefficient 0.01. Both these approaches are reproduced.

Our approach is performed with mini-batch size $B = B_U + B_L$, where B_L images are labeled and B_U images are originally unlabeled. We set $B_L = 50$ for CIFAR-10 and $B_L = 31$ for CIFAR100 and Mini-ImageNet. Same is also applied for MT. One epoch is defined as one pass through all originally unlabeled examples in the training set, meaning that images in I_L appear multiple times per epoch. We follow the same diffusion parameters as Iscen *et al*. [20]. We set $k = 50$ for graph construction, $\gamma = 3$ in (9), and $\alpha = 0.99$ in (10). We solve (10) with at most 20 iterations of CG. Pairwise similarities for the graph are computed with the publicly available FAISS library [21]. Confidence weights ω_i are normalized over all examples s.t. $\max_i \omega_i = 1$. Class weights ζ_j are normalized over c classes such that the average class weight is 1. Pseudo-label predictions, ω_i, and ζ_j are updated after each epoch.

To assess the benefit of diffusion, we finally evaluate a variant of our approach where the pseudo-labels are not provided by diffusion but derived from the network with (1) or from GTG propagation [8] instead. Training is performed with (12), as with our method. This is in the spirit of pseudo-labeling in prior work [36, 24].

5.3. Ablation Study

We investigate the impact of different components of our method. First, we study the effectiveness of weights introduced in the loss function (12). Table 1 shows the classification performance on CIFAR-10 test set, when using only 500 labeled examples for training and the rest of the training set is considered unlabeled. Different weighting schemes are evaluated by setting all ω_i to one, all ζ_i to one, or both to one. It is shown that both weights have positive contributions. We also show the benefit of predicting with diffusion over predicting by the trained network or GTG propagation. Pseudo-labeling by the network predictions uses examples

Dataset	CIFAR-10			
Nb. labeled images	500	1000	2000	4000
Fully supervised	49.08 ± 0.83	40.03 ± 1.11	29.58 ± 0.93	21.63 ± 0.38
TDCNN [36]†	-	32.67 ± 1.93	22.99 ± 0.79	16.17 ± 0.37
Network prediction (1) + weights	35.17 ± 2.46	23.79 ± 1.31	16.64 ± 0.48	13.21 ± 0.61
Ours: Diffusion prediction (7) + weights	32.40 ± 1.80	22.02 ± 0.88	15.66 ± 0.35	12.69 ± 0.29
VAT [26]†	-	-	-	11.36
Π model [23]†	-	-	-	12.36 ± 0.31
Temporal Ensemble [23]†	-	-	-	12.16 ± 0.24
MT [38]†	-	27.36 ± 1.30	15.73 ± 0.31	12.31 ± 0.28
MT [38]	27.45 ± 2.64	19.04 ± 0.51	14.35 ± 0.31	11.41 ± 0.25
MT + Ours	$\mathbf{24.02 \pm 2.44}$	$\mathbf{16.93 \pm 0.70}$	$\mathbf{13.22 \pm 0.29}$	$\mathbf{10.61 \pm 0.28}$

Table 2. Comparison with the state of the art on CIFAR-10. Error rate is reported. "13-layer" network is used. The top part of the table corresponds to training with pseudo-labels, while the bottom part of the table includes methods that are complementary to ours, as shown by the combination of our method with MT. † denotes scores reported in prior work.

Dataset	CIFAR-100		Mini-ImageNet-*top1*		Mini-ImageNet-*top5*	
Nb. labeled images	4000	10000	4000	10000	4000	10000
Fully supervised	55.43 ± 0.11	40.67 ± 0.49	74.78 ± 0.33	60.25 ± 0.29	53.07 ± 0.68	38.28 ± 0.38
Ours	46.20 ± 0.76	38.43 ± 1.88	$\mathbf{70.29 \pm 0.81}$	57.58 ± 1.47	$\mathbf{47.58 \pm 0.94}$	36.14 ± 2.19
MT [38]	45.36 ± 0.49	36.08 ± 0.51	72.51 ± 0.22	57.55 ± 1.11	49.35 ± 0.22	32.51 ± 1.31
MT + Ours	$\mathbf{43.73 \pm 0.20}$	$\mathbf{35.92 \pm 0.47}$	72.78 ± 0.15	$\mathbf{57.35 \pm 1.66}$	50.52 ± 0.39	$\mathbf{31.99 \pm 0.55}$

Table 3. Performance comparison on CIFAR-100 and Mini-ImageNet with 4k and 10k labeled images. Error rate is reported. "13-layer" network is used for CIFAR-100 and Resnet-18 is used for Mini-ImageNet. All methods are reproduced by us.

that the network can already classify, while diffusion allows for accurate predictions beyond those examples. In Figure 4, we report the progress of the pseudo-label accuracy on unlabeled images X_U throughout the training. Diffusion predictions are consistently better than network predictions.

Figure 5 demonstrates how ω_i accurately estimates the certainty of the prediction. From the plots we observe that predictions become more accurate as the training evolves, while at the beginning most examples are misclassified. The proposed weighting mechanism is robust to incorrect pseudo-labels and prevents model collapse. Figure 7 shows some of the incorrectly pseudo-labeled images with high certainty ω_i. Most of the incorrect labels come from trucks labeled as automobiles or birds labeled as frogs.

5.4. Comparison with the state-of-the-art

We present a comparison with state-of-the-art on all 3 datasets in Tables 2 and 3. The comparison includes performance reported in prior work and our reproduced results. In the case of the work by Shi *et al*. [36], we only compare with their TDCNN variant which refers to pseudo-labeling for network training. The other loss terms in their work are complementary to ours, similarly to MT. We additionally compare with our implementation of pseudo-labeling with network predictions combined with the proposed weights.

The proposed approach performs the best out of the pseudo-label based approaches on CIFAR-10. Results in Figure 6 show that our benefit is larger when the number of labels is reduced. The results on CIFAR-10 show that our approach is complementary to unsupervised loss, such as the one used by MT. This combination achieves the best performance on this dataset. The same holds for CIFAR-100 and Mini-ImageNet for 10k available labels. Our method also achieves a lower error rate than temporal ensemble (38.65 ± 0.51) and Π-model (39.19 ± 0.36) on CIFAR-100 [23] with 10k labels. On Mini-ImageNet with 4k available labels, the best performance is achieved when using our method without combining with Mean Teacher.

6. Conclusions

Most recent approaches for deep SSL rely on training with unsupervised loss on both labeled and unlabeled images. We have proposed an approach that relies on graph-based label propagation to infer pseudo-labels for the unlabeled images. An additional training set is formed with these pseudo-labels, which are shown to be more valuable than the pseudo-labels inferred by the network itself. Our method is in principle complementary to unsupervised loss terms, which is experimentally shown in this work.

Acknowledgments This work is supported by the GAČR grant 19-23165S and the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

References

[1] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. *ECCV*, 2018. 1

[2] Siddhartha Chandra and Iasonas Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep Gaussian CRFs. In *ECCV*, 2016. 4

[3] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006. 2

[4] Dengxin Dai and Luc Van Gool. Ensemble projection for semi-supervised image classification. In *ICCV*, 2013. 2

[5] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 1

[6] Wei Dong, Richard Socher, Li Li-Jia, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, June 2009. 6

[7] Matthijs Douze, Arthur Szlam, Bharath Hariharan, and Hervé Jégou. Low-shot learning with large-scale diffusion. In *CVPR*, 2018. 2

[8] Ismail Elezi, Alessandro Torcinovich, Sebastiano Vascon, and Marcello Pelillo. Transductive label augmentation for improved deep network learning. *arXiv preprint arXiv:1805.10546*, 2018. 2, 7

[9] Aykut Erdem and Marcello Pelillo. Graph transduction as a noncooperative game. *Neural Computation*, 24, 2012. 2

[10] Rob Fergus, Yair Weiss, and Antonio Torralba. Semi-supervised learning in gigantic image collections. In *NIPS*, 2009. 2

[11] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, 2018. 6

[12] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 1

[13] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *IJCV*, 124(2), 2017. 1

[14] Leo Grady. Random walks for image segmentation. *IEEE Trans. PAMI*, 28(11):1768–1783, 2006. 4

[15] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NIPS*, 2005. 2

[16] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Multimodal semi-supervised learning for image classification. In *CVPR*, 2010. 2

[17] Philip Haeusser, Alexander Mordvintsev, and Daniel Cremers. Learning by association – a versatile semi-supervised training method for neural networks. In *CVPR*, 2017. 2

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[19] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Mining on manifolds: Metric learning without labels. In *CVPR*, 2018. 1

[20] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, Teddy Furon, and Ondrej Chum. Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. In *CVPR*, 2017. 4, 7

[21] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017. 7

[22] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 6

[23] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, 2017. 2, 6, 8

[24] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICMLW*, 2013. 2, 3, 7

[25] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *ICLR*, 2017. 7

[26] Takeru Miyato, Shin-ichi Maeda, Shin Ishii, and Masanori Koyama. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Trans. PAMI*, 2018. 2, 8

[27] Avital Oliver, Augustus Odena, Colin Raffel, Ekin D Cubuk, and Ian J Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *ICLRW*, 2018. 2

[28] Deepak Pathak, Ross B Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *CVPR*, 2017. 1

[29] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. Deep co-training for semi-supervised image recognition. In *ECCV*, 2018. 2, 3

[30] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning CNN image retrieval with no human annotation. *IEEE Trans. PAMI*, 2018. 1

[31] Ilija Radosavovic, Piotr Dollar, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. In *CVPR*, 2018. 2

[32] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *NIPS*, 2015. 2

[33] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2016. 6

[34] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Mutual exclusivity loss for semi-supervised deep learning. In *ICIP*, 2016. 2

[35] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *NIPS*, 2016. 2

[36] Weiwei Shi, Yihong Gong, Chris Ding, Zhiheng Ma, Xiaoyu Tao, and Nanning Zheng. Transductive semi-supervised deep learning using min-max features. In *ECCV*, 2018. 2, 3, 4, 7, 8

[37] Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Constrained semi-supervised learning using attributes and comparative attributes. In *ECCV*, 2012. 2

[38] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS*, 2017. 2, 3, 4, 5, 6, 7, 8

[39] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NIPS*, 2016. 6

[40] Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for selfsupervised visual representation learning. In *ICCV*, 2017. 1

[41] Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In *ICML*, 2008. 2, 5

[42] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. *CVPR*, 2018. 1

[43] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *NIPS*, 2003. 1, 2, 3, 4

[44] Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld. *Semi-Supervised Learning with Graphs*. PhD thesis, Carnegie Mellon University, Language Technologies Institute, School of Computer Science Pittsburgh, PA, 2005. 4

[45] Xiaojin Zhu, John D Lafferty, and Zoubin Ghahramani. Semi-supervised learning: From Gaussian fields to Gaussian processes. Technical report, 2003. 1