Smooth Neighbors on Teacher Graphs for Semi-supervised Learning

Yucen Luo[1] Jun Zhu[1*] Mengxi Li[2] Yong Ren[1] Bo Zhang[1]
[1] Dept. of Comp. Sci. & Tech., State Key Lab for Intell. Tech. & Sys., BNRist Lab, Tsinghua University
[2] Department of Electronical Engineering, Tsinghua University
{luoyc15, limq14, renyong15}@mails.tsinghua.edu.cn; {dcszj, dcszb}@tsinghua.edu.cn

Abstract

The recently proposed self-ensembling methods have achieved promising results in deep semi-supervised learning, which penalize inconsistent predictions of unlabeled data under different perturbations. However, they only consider adding perturbations to each single data point, while ignoring the connections between data samples. In this paper, we propose a novel method, called Smooth Neighbors on Teacher Graphs (SNTG). In SNTG, a graph is constructed based on the predictions of the teacher model, i.e.*, the implicit self-ensemble of models. Then the graph serves as a similarity measure with respect to which the representations of "similar" neighboring points are learned to be smooth on the low-dimensional manifold. We achieve state-of-the-art results on semi-supervised learning benchmarks. The error rates are 9.89%, 3.99% for CIFAR-10 with 4000 labels, SVHN with 500 labels, respectively. In particular, the improvements are significant when the labels are fewer. For the non-augmented MNIST with only 20 labels, the error rate is reduced from previous 4.81% to 1.36%. Our method also shows robustness to noisy labels.*

1. Introduction

As collecting a fully labeled dataset is often expensive and time-consuming, semi-supervised learning (SSL) has been extensively studied in computer vision to improve generalization performance of the classifier by leveraging limited labeled data and a large amount of unlabeled data [9]. The success of SSL relies on the key *smoothness* assumption, *i.e.*, data points close to each other are likely to have the same label. It has a special case named *cluster* or *low density separation* assumption, which states that the decision boundary should lie in low density regions, not crossing high density regions [10]. Based on these assumptions, many traditional methods have been developed [22, 54, 51, 10, 4].

Recently due to the great advances of deep learning [25], remarkable results have been achieved on SSL [24, 35, 40, 27]. Among these works, perturbation-based methods [37, 2, 35, 39, 27] have demonstrated great promise. Adding noise to the deep model is important to reduce overfitting and learn more robust abstractions, *e.g.*, dropout [21] and randomized data augmentation [13]. In SSL, perturbation regularization aids by exploring the *smoothness* assumption. For example, the Manifold Tangent Classifier (MTC) [37] trains contrastive auto-encoders to learn the data manifold and regularizes the predictions to be insensitive to local perturbations along the low-dimensional manifold. Pseudo-Ensemble [2] and Γ model in Ladder Network [35] evaluate the classifiers with and without perturbations, which act as a *"teacher"* and a *"student"*, respectively. The student needs to predict consistently with the targets generated by the teacher on unlabeled data. Following the same principle, temporal ensembling, mean teacher and virtual adversarial training [27, 46, 33] improve the target quality in different ways to form better teachers. All these approaches aim to fuse the inputs into coherent clusters by adding noise and smoothing the mapping function locally [27].

However, these methods only consider the perturbations around *each single* data point, while ignoring the connections between data points, therefore not fully utilizing the information in the unlabeled data structure, such as clusters or manifolds. An extreme situation may happen where the function is smooth in the vicinity of each unlabeled point but not smooth in the vacancy among them. This artifact could be avoided if the unlabeled data structure is taken into consideration. It is known that data points similar to each other (*e.g.*, in the same class) tend to form clusters (*cluster* assumption). Therefore, the connections between similar data points help the fusing of clusters become tighter and more effective (see Fig. 5 for the visualization of real data).

Motivated by that, we propose *Smooth Neighbors on Teacher Graphs* (SNTG) that considers the connections between data points to induce smoothness on the data manifold. By learning a teacher graph based on the targets generated by the teacher, our model encourages invariance when some perturbations are added to the neighboring points on the graph. Since deep networks have a hierarchical property,

*Corresponding author.

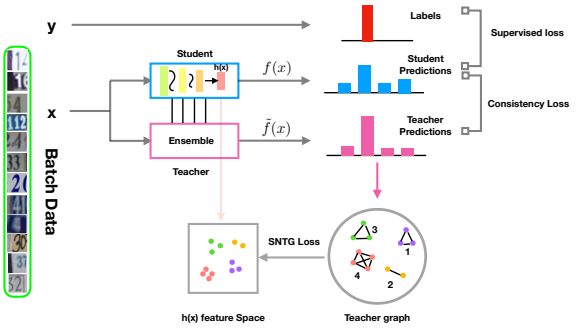


Figure 1: The structure of our model.

the top layer maps the inputs into a low-dimensional feature space [5, 42, 29]. Given the teacher graph, SNTG makes the learned features more discriminative by enforcing them to be similar for neighbors and dissimilar for those non-neighbors. The model structure is depicted in Fig. 1. We then propose a doubly stochastic sampling algorithm to reduce the computational cost with large mini-batch sizes. Our method can be applied with very little engineering effort to existing deep SSL works including both generative and discriminative approaches because SNTG does not introduce any extra network parameters. We demonstrate significant performance improvements over state-of-the-art results while the extra time cost is negligible.

2. Related work

Using unlabeled data to improve generalization has a long and rich history and the literature in SSL is vast [52, 9]. So in this section we focus on reviewing the closely related papers, especially the recent advances in SSL with deep learning.

Self-training methods iteratively use the current classifier to label those unlabeled ones with high confidence [38]. Co-training [6, 34] uses a pair of classifiers with disjoint views of data to iteratively learn and generate training labels. Transductive SVMs [22] implement the *cluster* assumption by keeping unlabeled data far away from the decision boundaries. Entropy minimization [19], a strong regularization term commonly used, minimizes the conditional entropy $H\left(p\left(y|x\right)\right)$ to ensure that one instance is assigned to one class with a high probability to avoid class overlap.

Graph-based Methods. Graph-based SSL methods [53, 54, 51] define the similarity of data points by a graph and make predictions smooth with respect to the graph structure. Many of them often optimize a supervised loss over labeled data with a graph Laplacian regularizer [4, 18]. Label propagation [53] pushes label information from a labeled instance to its neighbors using a predefined distance metric. We emphasize that our work differs from these traditional methods in the construction and utilization of the graph. Previous work usually constructs the graph in advance using prior knowledge or manual labeling and the graph remains fixed in the following training process [4, 50]. This can lead to several disadvantages as detailed in Sec. 4.2 and 5.3. Although some works [49] establish the graph dynamically during the classification, their performance is far from recent state-of-the-art deep learning based methods.

Generative Approaches. Besides aforementioned discriminative approaches, another line is generative models, which pay efforts to learn the input distribution $p(x)$ that is believed to share some information with the conditional distribution $p(y|x)$ [28]. Traditional models such as Gaussian mixtures [52] try to maximize the joint log-likelihood of both labeled and unlabeled data using EM. For modern deep generative models, variational auto-encoder (VAE) makes it scalable by employing variational methods combined with deep learning [24] while generative adversarial networks (GAN) generate samples by optimizing an adversarial game between the discriminator and the generator [43, 40, 11, 15]. The samples generated by GAN can be viewed as another kind of "data augmentation" to "tell" the decision boundary where to lie. For example, "fake" samples can be generated in low density regions where the training data is rare [40, 15] based on the *low density separation* assumption. Alternatively, more "pseudo" samples could be generated in high density regions to keep away from the decision boundary thus improve the robustness of the classifier [11]. Our work is complementary to these efforts and can be easily combined with them. We observe improvements over feature matching GAN [40] with SNTG (see Section 5.6).

3. Background

We consider the semi-supervised classification task, where the training set $\mathcal{D}$ consists of N examples, out of which L have labels and the others are unlabeled. Let $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{L}$ be the labeled set and $\mathcal{U} = \{x_i\}_{i=L+1}^{N}$ be the unlabeled set where the observation $x_i \in \mathcal{X}$ and the corresponding label $y_i \in \mathcal{Y} = \{1, 2, ..., K\}$. We aim to learn a function $f : \mathcal{X} \to [0, 1]^K$ parameterized by $\theta \in \Theta$ by solving a generic optimization problem:

$$\min_{\theta} \sum_{i=1}^{L} \ell(f(x_i; \theta), y_i) + \lambda R(\theta, \mathcal{L}, \mathcal{U}), \qquad (1)$$

where ℓ is a pre-defined loss function like cross-entropy loss and $f(x; \theta)$ represents the predicted distribution $p\left(y|x; \theta\right)$. Since only a small portion of training data is labeled ($L \ll N$), the regularization term R is important to leverage unlabeled data. Here, λ is a non-negative regularization parameter that controls how strongly the regularization is penalized.

3.1. Perturbation-based methods

As mentioned earlier, the models in perturbation-based methods assume a dual role, *i.e.*, a teacher and a student [30]. The training targets for the student are generated by the teacher. Recent progresses focus on improving the quality of targets by using self-ensembling and exploring different perturbations [27, 46, 33], as

summarized in [46]. Formally, self-ensembling methods [27, 46] fit in Eq. (1) by defining R as a consistency loss:

$$R_C(\theta, \mathcal{L}, \mathcal{U}) = \sum_{i=1}^{N} \mathbb{E}_{\xi', \xi}\, d\big(\tilde{f}(x_i; \theta', \xi'), f(x_i; \theta, \xi)\big), \quad (2)$$

where $\tilde{f}$ is a "noisy" teacher model with parameters θ' and random perturbations ξ', similarly, f is a student model with θ and ξ, and $d(\cdot, \cdot)$ denotes the divergence between the two distributions. For example, d can be l_2 distance or KL divergence. The perturbations include the input noise and the network dropout. The teacher is defined as an implicit ensemble of previous student models and is expected to give better predictions than the student. $\tilde{f}(x)$ can be seen as the training targets and the student is supposed to predict consistently with $\tilde{f}(x)$. Below are several ways to define the teacher $\tilde{f}$, which have proven effective in previous work [27, 46, 33].

Π **model [27].** In order to alleviate the bias in the targets, Π model adds noise ξ' to $\tilde{f}$, which shares the same parameters with f, *i.e.*, $\theta' = \theta$ in Eq. (2). Π model evaluates the network twice under different realizations of i.i.d. perturbations ξ' and ξ every iteration and minimizes their l_2 distance. We observe that, in this case, optimizing the objective in Eq. (2) is equivalent to minimizing the variance of the prediction. See details in Appendix B.

Temporal ensembling (TempEns) [27]. To reduce the variance of targets, TempEns maintains an exponentially moving average (EMA) of predictions over epochs as $\tilde{f}$. The ensemble output is defined as

$$\tilde{F}^{(t)}(x_i) = \alpha \tilde{F}^{(t-1)}(x_i) + (1 - \alpha) f^{(t)}(x_i; \theta, \xi), \quad (3)$$

where $f^{(t)} : \mathcal{X} \to [0, 1]^K$ is the prediction given by the current student model at training epoch t and α is the momentum. The target given by $\tilde{f}$ for x_i at epoch t is the debias correction of $\tilde{F}^{(t)}$, divided by factor $(1 - \alpha^t)$, *i.e.*, $\tilde{f}^{(t)}(x_i) = \tilde{F}^{(t)}(x_i)/(1 - \alpha^t)$. Since the target $\tilde{f}(x_i)$ obtained in TempEns is based on EMA, the network only needs to be evaluated once, leading to a speed-up for Π model.

Mean teacher (MT) [46]. Instead of averaging predictions every epoch, MT updates the targets more frequently to form a better teacher, *i.e.*, it averages parameters θ every iteration:

$$\theta' \leftarrow \alpha \theta' + (1 - \alpha)\theta. \quad (4)$$

MT provides more accurate targets and enables learning large datasets. It also evaluates the network twice, one by teacher $\tilde{f}(\cdot; \theta', \xi')$ and the other by student $f(\cdot; \theta, \xi)$.

Virtual adversarial training (VAT) [33]. Instead of l_2 distance, VAT defines R as the KL divergence between the model prediction and that of the input under adversarial perturbations ξ'_{adv}:

$$R_C(\theta, \mathcal{L}, \mathcal{U}) = \sum_{i=1}^{N} \mathrm{KL}(\tilde{f}(x_i; \theta) \| f(x_i; \theta, \xi'_{adv})). \quad (5)$$

It is assumed that a model trained under the worst-case (adversarial) perturbations will generalize well [33]. Generally, VAT is also in the framework of self-ensembling in the sense of enforcing consistent predictions. VAT resembles Π model but distinguishes itself in the distance metric and the type of perturbations. $\tilde{f}$ in Eq. (5) can be seen as the teacher model while f with ξ'_{adv} is treated as the student model.

As these methods generate targets themselves, the teacher model is likely to render incorrect targets. However, previous results [27, 46] as well as ours (see Sec. 5.2 and 5.5) suggest that the "teacher-student" models converge well and are robust to incorrect labels. They mitigate the hazard by using a better teacher and the balanced trade-off between ℓ and R_C. The success of these methods can be understood as indirectly exploiting the *low-density separation* assumption because the points near the decision boundaries are prone to alter predictions under perturbations thus have large consistency losses. The explicitly penalized R_C will keep unlabeled data far away from the decision boundaries in low density regions and concentrated in high density regions.

4. Our approach

One common shortcoming of the perturbation-based methods is that they regularize the output to be smooth near a data point locally, while ignoring the cluster structure. We address it by proposing a new SSL method, SNTG, that enforces neighbors to be smooth, which is a stronger regularization than only imposing smoothness at a single unlabeled point. We show that SNTG contributes to form a better teacher model, which is the focus of recent advances on perturbation-based methods. In the following, we formalize our approach by answering two key questions: (1) how to define the graph and neighbors? and (2) how to induce the smoothness of neighboring points using the graph?

4.1. Learning the graph with the teacher model

Most existing graph-based SSL methods [4, 50] depend on a distance metric in the input space $\mathcal{X}$, which is typically low-level (*e.g.*, pixel values of images). For natural images, pixel distance cannot reflect semantic similarity well. Instead, we use the distance in the label space $\mathcal{Y}$, and treat the data points from the same class as neighbors. However, an issue is that the true labels of unlabeled data are unknown. We address it by learning a teacher graph using the targets generated by the teacher model. Self-ensembling is a good choice for constructing the graph because the ensemble predictions are expected to be more accurate than the outputs of current classifier. Inspired by that, a teacher graph can guide the student model to move in correct directions. A comparison to other graphs could be found in Sec. 5.3.

Formally, for $x_i \in \mathcal{D}$, a target prediction $\tilde{f}(x_i)$ is given by the teacher defined in the previous section. Denote the

hard target as $\tilde{y}_i = \text{argmax}_k \left[\tilde{f}(x_i)\right]_k$ where $[\cdot]_k$ is the k-th component of the vector, indicating the probability that the example is of class k. We build the graph as follows:

$$W_{ij} = \begin{cases} 1 & \text{if } \tilde{y}_i = \tilde{y}_j \\ 0 & \text{if } \tilde{y}_i \neq \tilde{y}_j \end{cases}, \quad (6)$$

where W_{ij} measures the similarity between sample x_i and x_j and those pairs with nonzero entries are treated as "neighbors". Here we simply restrict $W_{ij} \in \{0, 1\}$ to construct a 0-1 sparse graph. Other choices include computing the KL divergence between the *soft* predictions $\tilde{f}(x_i)$ and $\tilde{f}(x_j)$.

4.2. Guiding the low-dimensional feature mapping

Given a graph, we clarify how to regularize neighbors with smoothness. Generally, a deep classifier (*i.e.*, the student) f can be decomposed as $f = g \circ h$, where $h : \mathcal{X} \to \mathbb{R}^p$ is the mapping from the input space to the penultimate layer and $g : \mathbb{R}^p \to [0, 1]^K$ is the output layer usually parameterized by a fully-connected layer with softmax. Due to the hierarchical nature of deep networks, $h(x)$ can be seen as a low-dimensional feature of the input. And the feature space is expected to be linearly separable, as shown in the common practice that a following linear classifier g suffices. In terms of approximating the semantic similarity of two instances, the Euclidean distance of $h(x_i)$ and $h(x_j)$ is more suitable than that of $f(x)$ which represents class probabilities. Hence we use the graph to guide $h(x)$ in the feature space, making them distinguishable among classes.

Given a $N \times N$ similarity matrix W of the sparse graph, we define the SNTG loss as

$$R_S(\theta, \mathcal{L}, \mathcal{U}) = \sum_{x_i, x_j \in \mathcal{D}} \ell_G(h(x_i; \theta), h(x_j; \theta), W_{ij}) \quad (7)$$

The choice of ℓ_G is quite flexible, which is related to unsupervised feature learning or clustering. Traditional choices include multidimensional scaling [14], ISOMAP [47] and Laplacian eigenmaps [3]. Here we utilize the contrastive Siamese networks [8] since they are able to learn an invariant mapping to a smooth and coherent feature space and perform well in metric learning and face verification [20, 12, 45]. Specifically, the loss is defined as follows:

$$\ell_G = \begin{cases} \|h(x_i) - h(x_j)\|^2 & \text{if } W_{ij} = 1 \\ \max\left(0, m - \|h(x_i) - h(x_j)\|\right)^2 & \text{if } W_{ij} = 0 \end{cases} \quad (8)$$

where $m > 0$ is a pre-defined margin and $\|\cdot\|$ is Euclidean distance. The margin loss is to constrain neighboring points to have consistent features. Consequently, the neighbors are encouraged to have consistent predictions while the non-neighbors (*i.e.*, the points of different classes) are pushed apart from each other with a minimum distance m. Visualizations can be found in Section 5.4.

One interpretation of why the proposed method works well is that SNTG explores more information in the teacher and improves the target quality. The teacher graph leads to better abstract representations in a smooth and coherent feature space and then aids the student f to give more accurate predictions. In turn, an improved student contributes to a better teacher model which can provide more accurate targets. Another perspective is that SNTG implements the *manifold* assumption for classification which underlies the loss ℓ_G, *i.e.*, the points of same class are encouraged to concentrate together on sub-manifolds. The perturbation-based methods only keep the decision boundaries far away from each unlabeled data point while our method encourages the unlabeled data points to form tighter clusters, leading the decision boundaries to locate between the clusters.

We discuss the difference between SNTG and two early works LPDGL [18] and *Embed*NN [50]. For LPDGL, the definition and the usage of local smoothness are both different from ours. LPDGL defines deformed Laplacian to smooth the predictions of k neighbors in a local region while our work enforces the features to be smooth by the contrastive loss in Eq. (8) w.r.t. the 0-1 teacher graph. For *Embed*NN, despite they also measure the embedding loss, there are several key differences. First, inspired by Π model, SNTG aims to induce more smoothness using neighbors under perturbations, while *Embed*NN is motivated by using the embedding as an auxiliary task to help supervised tasks and does not consider the robustness to perturbations. Second, *Embed*NN uses a fixed graph W defined by k-nearest-neighbor (k-NN) based on the distance in $\mathcal{X}$. Our method takes a different approach using the teacher-generated targets in $\mathcal{Y}$. As mentioned in Section 4.1, the pixel-level distance in $\mathcal{X}$ may not reflect the semantic similarity as well as that in $\mathcal{Y}$ for natural images. Third, once the graph is built in *Embed*NN, the fixed graph cannot leverage the knowledge distilled by the classifier thus cannot be improved any more, while SNTG jointly learns the classifier and the teacher graph as stated above. Furthermore, on the time cost and scalability, SNTG is faster than *Embed*NN and can handle large-scale datasets. k-NN in *Embed*NN is slow for large k and even more time-consuming for large-scale datasets. We compute W in the much lower dimensional $\mathcal{Y}$ and use the sub-sampling technique that is to be introduced next. Experimental comparisons are in Section 5.3.

4.3. Doubly stochastic sampling approximation

Our overall objective is the sum of two components. The first one is the standard cross-entropy loss on the labeled data, and the second is the regularization term, which encourages the smoothness for each single point (*i.e.*, R_C) as well as for the neighboring points (*i.e.*, R_S). Alg. 1 presents the pseudo-code. Following [27], we use a ramp-up $w(t)$ for both the learning rate and the regularization term in the beginning.

As our model uses deep networks, we train it using Stochastic Gradient Descent (SGD) [7] with mini-batches.

Algorithm 1 Mini-batch training of SNTG for SSL

Require: x_i = training inputs, y_i for labeled inputs in $\mathcal{L}$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = neural network with parameters θ
for t in [1, numepochs] **do**
  for each minibatch B **do**
    $f_i \leftarrow f_\theta(x_{i\in B})$ evaluate network outputs
    $\tilde{f}_i \leftarrow \tilde{f}(x_{i\in B})$ given by the teacher model
    for (x_i, x_j) in a minibatch pairs S from B **do**
      Compute W_{ij} according to Eq. (6)
    end for
    loss $\leftarrow -\frac{1}{|B|}\sum_{i\in(B\cap\mathcal{L})}\log[f_i]_{y_i}$
    $+w(t)\Big[\lambda_1\frac{1}{|B|}\sum_{i\in B} d(\tilde{f}_i, f_i)$
    $+\lambda_2\frac{1}{|S|}\sum_{i,j\in S}\ell_G(h(x_i), h(x_j), W_{ij})\Big]$
    update θ using optimizers, *e.g.*, Adam [23]
  end for
end for
return θ

We follow the common practice and construct the sub-graph in a random mini-batch to estimate R_S in Eq. (7). For a mini-batch B of size n, we need to compute W_{ij} for all the data pairs $(x_i, x_j) \in B$, which is of size n^2 in total. Although this step is fast, the computation of $\|h(x_i) - h(x_j)\|$ related to W_{ij} is $O(p)$ and then the overall computational cost is $O(n^2p)$, which is slow for large n. To reduce the computational cost, we instead use doubly stochastic sampled data pairs to construct W_{ij} and only use them to compute Eq. (8), which is still an unbiased estimation of R_S. Specifically, in each iteration, we sample a mini-batch B and then subsample $s \leq n^2$ data pairs S from B. Empirically, SNTG can be incorporated into other SSL methods with not much extra time cost. See Appendix A for details.

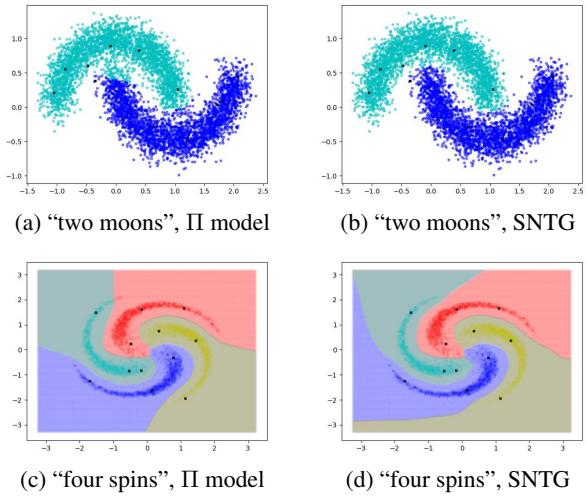
5. Experiments

This section presents both quantitative and qualitative results to demonstrate the effectiveness of SNTG. The purpose of experiments is to show the improvements that come from SNTG, using cutting-edge approaches as evidence. [1]

5.1. Synthetic datasets

We first test on the well-known "two moons" and "four spins" synthetic datasets where $x \in \mathbb{R}^2$ and $y \in \{1, 2\}$ and $y \in \{1, 2, 3, 4\}$, respectively. Each dataset includes 6000 data points and the label ratio is 0.002 (*i.e.*, only 12 data points are labeled). We use neural networks with three hidden layers, each of size 100 with leaky ReLU $\alpha = 0.1$ as suggested in CatGAN [43]. See Appendix A for details. The results are visualized in Fig. 2, where we compare with

[1] Source code is at `https://github.com/xinmei9322/SNTG`.



(a) "two moons", Π model (b) "two moons", SNTG

(c) "four spins", Π model (d) "four spins", SNTG

Figure 2: Comparison between Π model (a,c) and SNTG (b,d) on two synthetic datasets. The labeled data are marked with the black cross. Different colors denote different classes. The decision boundaries are shown for 2c and 2d.

Table 1: Error rates (%) on benchmark datasets without augmentation, averaged over 10 runs.

Models	MNIST (L=100)	SVHN (L=1000)	CIFAR-10 (L=4000)	CIFAR-100 (L=10000)
LadderNetwork [35]	0.89±0.50	–	20.40±0.47	–
CatGAN [43]	1.39±0.28	–	19.58±0.58	–
ImprovedGAN [40]	0.93±0.065	8.11±1.3	18.63±2.32	–
ALI [17]	–	7.42±0.65	17.99±1.62	–
TripleGAN [11]	0.91±0.58	5.77±0.17	16.99±0.36	–
GoodBadGAN [15]	0.795±0.098	4.25±0.03	14.41±0.03	–
Π model [27]	0.89±0.15*	5.43±0.25	16.55±0.29	39.15±0.36
Π+SNTG (**ours**)	**0.66±0.07**	4.22±0.16	13.62±0.17	**37.97±0.29**
VAT [33]	1.36	5.77	14.82	–
VAT+Ent [33]	–	4.28	13.15	–
VAT+Ent+SNTG (**ours**)	–	**4.02±0.20**	**12.49±0.36**	–

Π model, a strong baseline that performs well with only some failures. Specifically, in Fig. 2a, a small blob of data is misclassified to green and in Fig. 2c, the tail of the green spin is misclassified to red. The prediction of Π model is supposed to be smooth enough at these areas because the data points are in blobs. However, the Π model still fails to identify them. For our SNTG, the classifications are both correct in Fig. 2b and Fig. 2d due to effective utilization of neighboring points' structure. Compared to Fig. 2c, the decision boundaries in Fig. 2d also align better with the spins. These experiments demonstrate the effectiveness of SNTG.

5.2. Benchmark datasets

We then provide results on the widely adopted benchmarks, MNIST, SVHN, CIFAR-10 and CIFAR-100. Following common practice [35, 40], we randomly sample 100, 1000 4000 and 10000 labels for MNIST, SVHN, CIFAR-10

Table 2: Error rates (%) on SVHN with translation augmentation, averaged over 10 runs.

Model	250 labels	500 labels	1000 labels	All labels
Supervised-only [46]	42.65±2.68	22.08±0.73	14.46±0.71	2.81±0.07
Π model [27]	9.93±1.15*	6.65±0.53	4.82±0.17	2.54±0.04
Π+SNTG (**ours**)	5.07±0.25	4.52±0.30	**3.82±0.25**	**2.42±0.05**
TempEns [27]	12.62±2.91*	5.12±0.13	4.42±0.16	2.74±0.06
TempEns+SNTG (**ours**)	5.36±0.57	4.46±0.26	3.98±0.21	2.44±0.03
MT [46]	4.35±0.50	4.18±0.27	3.95±0.19	2.50±0.05
MT+SNTG (**ours**)	**4.29±0.23**	**3.99±0.24**	3.86±0.27	2.42±0.06
VAT [33]	–	–	5.42	–
VAT+Ent [33]	–	–	3.86	–
VAT+Ent+SNTG (**ours**)	–	–	3.83±0.22	–

Table 3: Error rates (%) on CIFAR-10 with standard augmentation, averaged over 10 runs.

Model	1000 labels	2000 labels	4000 labels	All labels
Supervised-only [27]	–	–	34.85±1.65	6.05±0.15
Π model [27]	31.65±1.20*	17.57±0.44*	12.36±0.31	5.56±0.10
Π+SNTG (**ours**)	21.23±1.27	14.65±0.31	11.00±0.13	**5.19±0.14**
TempEns [27]	23.31±1.01*	15.64±0.39*	12.16±0.24	5.60±0.10
TempEns+SNTG (**ours**)	**18.41±0.52**	**13.64±0.32**	10.93±0.14	5.20±0.14
VAT [33]	–	–	11.36	5.81
VAT+Ent [33]	–	–	10.55	–
VAT+Ent+SNTG (**ours**)	–	–	**9.89±0.34**	–

and CIFAR-100, respectively. We further explore fewer labels for the non-augmented MNIST as well as SVHN and CIFAR-10 with standard augmentation. The results are averaged over 10 runs with different seeds for data splits. Main results are presented in Tables 1, 2, 3 and 4. The accuracy of baselines are all taken from existing literature. In general, we can see that our method surpasses previous state-of-the-arts by a large margin.

All models are trained with the same network architecture and hyper-parameters to our baselines, *i.e.*, perturbation-based methods described in Sec. 3.1. The SNTG loss only needs three extra hyper-parameters: the regularization parameter λ_2, the margin m and the number of sub-sampled pairs s. We fix m and s, and only tune λ_2. More details on experimental setup can be found in Appendix A. For fair comparison, we also report our best implementation under the settings not covered in [27] (marked $*$).

Note that VAT is a much stronger baseline than Π model and TempEns since it explores adversarial perturbation with extra efforts and more time. VAT's best results are achieved with an additional entropy minimization (Ent) regularization [19]. We evaluate our method under the best setting VAT+Ent and observe a further improvement with SNTG, *e.g.*, from 13.15% to 12.49% and from 10.55% to 9.89% on CIFAR-10 without or with augmentation, respectively. In fact, we observed that Ent could also improve the performance of other self-ensembling methods if it was added along with SNTG. But to keep the results clear and focus on the efficacy of SNTG, we did not illustrate the results here.

Table 4: Error rates (%) on MNIST without augmentation.

Models	20 labels	50 labels	100 labels
ImprovedGAN [40]	16.77±4.52	2.21±1.36	0.93±0.065
Triple GAN [11]	4.81±4.95	1.56±0.72	0.91±0.58
Π model [27]	6.32±6.90*	1.02±0.37*	0.89±0.15*
Π+SNTG (**Ours**)	**1.36±0.78**	**0.94±0.42**	**0.66±0.07**

As shown in Tables 2 and 3, when SNTG is applied to the fully supervised setting (*i.e.*, all labels are observed), our method further reduces the error rates compared to self-ensembling methods, *e.g.*, from 5.56% to 5.19% on CIFAR-10 for Π model. It suggests that supervised learning also benefits from the additional smoothness and the learned invariant feature space in SNTG.

Fewer labels. Notably, as shown in Tables 4, 2 and 3, when labels are very scarce, *e.g.*, MNIST with 20 labels (only 2 labeled samples per class), SVHN with 250 labels and CIFAR-10 with 1000 labels, the benefits provided by SNTG are even more significant. The SNTG regularizer empirically reduces the overfitting on the small set of labeled data and thus yields better generalization.

Ablation study. Our reported results are based on adding SNTG loss R_S to baselines, and the overall objective has already included the consistency loss R_C (See Alg. 1, line

Table 5: Ablation study on CIFAR-10 with 4000 labels without augmentation. L_S denotes the supervised loss (the first term in Eq. 1), and R_C and R_S are defined in text. L_S+R_C equals to Π model and L_S+R_C+R_S equals to Π+SNTG.

Settings	L_S	L_S+R_C	L_S+R_S	L_S+R_C+R_S
Error (%)	35.56	16.55	15.36	13.62

9). To quantify the effectiveness of our method, Table 5 presents the evaluation of Π+SNTG compared to its ablated versions. The error rate of Π model, which only uses R_C, is 16.55%. However, using R_S alone yields a lower error rate of 15.36%. Thus, R_S considering the neighbors proves to be a strong regularization, comparable or even favorable to R_C, and they are also complementary.

Convergence. A potential concern of our method is the convergence, since the information in a teacher graph is likely to be inaccurate at the beginning of training. However, we did not observe any divergent cases in all experiments. Empirically, the teacher model is usually a little better than the student in training. Furthermore, the ramp-up $w(t)$ is used to balance the trade-off between the supervised loss and regularization, which is important for the convergence as described in previous works [27, 46]. Using the ramp-up weighting mechanism, the supervised loss dominates the learning in earlier training. As the training continues, the student model has more confidence in the information given by the teacher model, *i.e.*, the target predictions and the graph, which gradually contributes more to the learning process. Fig. 3 shows that our model converges well.

5.3. Comparison to *Embed*NN and other graphs

As our graph is learned based on the predictions in $\mathcal{Y}$ given by the teacher model, we further compare to other graphs. We test them on CIFAR-10 using 4000 labels without augmentation and share all the same hyper-parameter settings with Π model except the definition of W. The first baseline is a fixed graph defined by k-NN in $\mathcal{X}$—Following *Embed*NN [50], W is predefined so that 10 nearest neighbors of x_i have $W_{ij} = 1$, and $W_{ij} = 0$ otherwise. The second one is another fixed graph in $\mathcal{Y}$—Since only a small portion of labels are observed on training data in SSL, we construct the graph based on the predictions of a pre-trained Π model on training data. Fig. 3 shows that our model outperforms other graphs. The test error rate of the baseline Π model is 16.55%. Using k-NN in $\mathcal{X}$ gives a marginal improvement to 16.13%. Using the predictions in pre-trained Π model to construct a 0-1 fixed graph, the error rate is 15.71%. Using our method, learning a teacher graph from scratch, Π+SNTG achieves superior result with 13.62% error rate.

Note that Π model is a strong baseline surpassing most previous methods. For natural images like CIFAR-10, the pixel-level distance provides limited information for the similarity thus k-NN graph in $\mathcal{X}$ does not improve the strong baseline. The reason of the performance gap to the second one lies in that using a fixed graph in $\mathcal{Y}$ is more like "pre-training" while using teacher graph is like "joint-training". The teacher graph becomes better using the information extracted by the teacher and then benefits it in turn. However, the fixed graphs cannot receive feedbacks from the model in the training and all the information is from the pre-training or prior knowledge. Empirical results support our analysis.

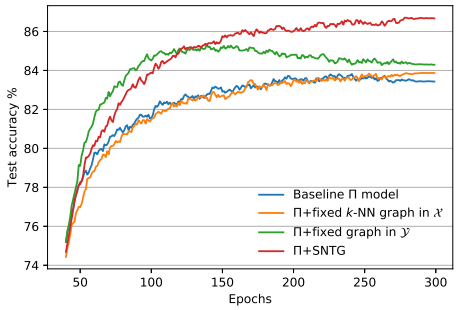


Figure 3: Comparison to the fixed graphs on CIFAR-10 with 4000 labels without augmentation.

5.4. Visualization of embeddings

We visualize the embeddings of our algorithm and Π model on test data under the same settings (CIFAR-10 with 4000 labels and MNIST with 100 labels, both without augmentation). We implemented it using TensorBoard in TensorFlow [1]. Fig. 5 shows the representations $h(x) \in \mathbb{R}^{128}$ projected to 2 dimension using PCA or tSNE [32]. The learned representations of our model are more concentrated within clusters and are potentially easier to separate for different classes. The visualization is also consistent with our assumption and analysis.

5.5. Robustness to noisy labels

We finally show that SNTG can not only benefit from unlabeled data, but also learn from noisy supervision. Following [27], we did extra experiments on supervised SVHN to show the tolerance to incorrect labels. Certain percentages of true labels on the training set are replaced by random labels. Fig. 4 shows that TempEns+SNTG retains over 93% accuracy even when 90% of the labels are noisy while TempEns alone only obtains 73% accuracy [27]. With standard supervised training, the model suffers a lot and overfits to the incorrect information in labels. Thus, our SNTG regularization improves the robustness and generalization performance of the model. Previous work [36] also shows that self-generated targets yield robustness to label noise.

5.6. Feature matching GAN benefits from SNTG

Recently, the feature matching (FM) GAN in Improved GAN [40] has performed well for SSL but usually generates

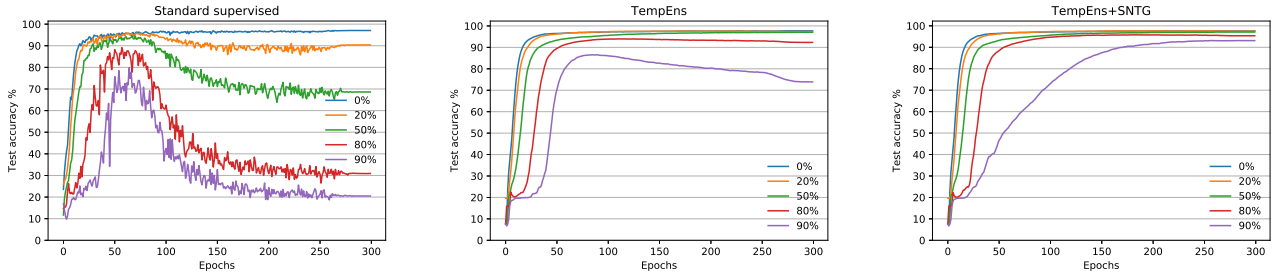
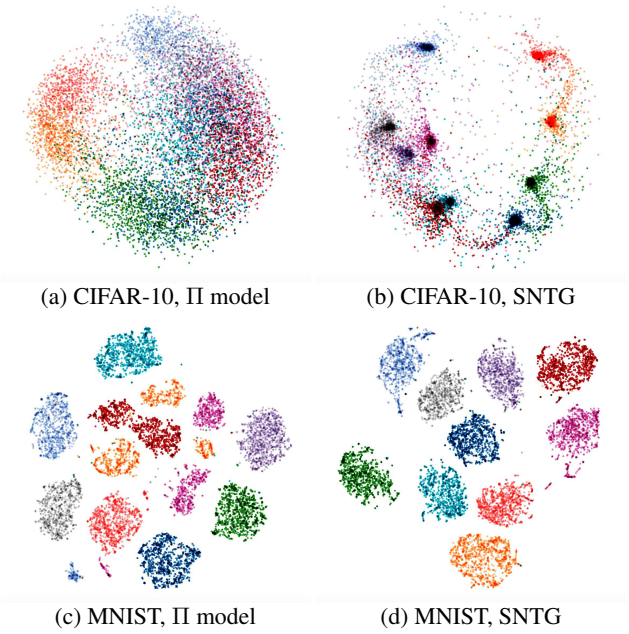


Figure 4: Test accuracy on supervised SVHN with noisy labels. Different colors denote the percentages of corrupted labels. With standard supervised training (left), the model suffers a lot and overfits to the incorrect information in labels. TempEns (middle) shows the resistance to the corruption but still has a drop in accuracy when the portion of randomized labels increases to 90%. Adding SNTG shows almost perfect robustness even when 90% labels are corrupted.



(a) CIFAR-10, Π model (b) CIFAR-10, SNTG

(c) MNIST, Π model (d) MNIST, SNTG

Figure 5: (a, b) are the embeddings of CIFAR-10 test data projected to 2-D using PCA. (c, d) are the 2-D embeddings of MNIST test data using t-SNE. Each color denotes a class. In (b, d) with SNTG, the embeddings of each class form a tight and concentrated cluster. In (c) without SNTG, the cluster of the same class are divided into several parts.

images with strange patterns. Some works have been done to analyze the reasons [15, 11, 26]. An interesting finding is that our method can also alleviate the problem. Fig. 6 presents the comparison between the samples generated in FM GAN [40] and FM GAN+SNTG. Apart from improving the generated sample quality of FM GAN, SNTG also reduces the error rate. FM GAN achieves 18.63% on CIFAR-10 with 4000 labels. We regularize the features of unlabeled data using SNTG and observe an improvement to 14.93%, which is comparable to the state-of-the-art 14.41% in deep generative models [15].

In FM GAN, the objective for the generator is defined as

$$\|\mathbb{E}_{x\sim p_{data}} h(x) - \mathbb{E}_{x\sim p_G} h(x)\|_2^2, \tag{9}$$

(a) FM GAN (b) FM GAN+SNTG

Figure 6: Comparison of generated images in SSL on CIFAR-10 with FM GAN [40], original in their paper (*left*) and with our SNTG loss (*right*). FM GAN has strange and repeated patterns in the samples. Adding SNTG, the quality and diversity of generated samples are improved.

which is similar to the neighboring case when $W_{ij} = 1$ in Eq. (8). In our opinion, SNTG helps shape the feature space better so that the generator could capture the data distribution by matching only the mean of features.

6. Conclusions and future work

We present a simple but effective SNTG, which regularizes the neighboring points on a learned teacher graph. Empirically, it outperforms all baselines and achieves new state-of-the-art results on several datasets. As a byproduct, we also learn an invariant mapping on a low-dimensional manifold. SNTG offers additional benefits such as handling extreme cases with fewer labels and noisy labels. In future work, it is promising to do more theoretical analysis of our method and to explore its combination with generative models as well as applications to large-scale datasets, *e.g.*, ImageNet with more classes.

Acknowledgements

The work is supported by the National NSF of China (Nos. 61620106010, 61621136008, 61332007), Beijing Natural Science Foundation (No. L172037), Tsinghua Tiangong Institute for Intelligent Computing, the NVIDIA NVAIL Program and a research fund from Siemens.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] P. Bachman, O. Alsharif, and D. Precup. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, pages 3365–3373, 2014.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[4] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(Nov):2399–2434, 2006.

[5] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. Also published as a book. Now Publishers, 2009.

[6] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[7] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[8] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

[9] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[10] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *AISTATS*, pages 57–64, 2005.

[11] L. Chongxuan, T. Xu, J. Zhu, and B. Zhang. Triple generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 4091–4101, 2017.

[12] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition*, volume 1, pages 539–546. IEEE, 2005.

[13] D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[14] T. F. Cox and M. A. Cox. *Multidimensional scaling*. CRC press, 2000.

[15] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. R. Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *Advances in Neural Information Processing Systems*, pages 6513–6523, 2017.

[16] S. Dieleman, J. Schlter, C. Raffel, E. Olson, S. K. Snderby, D. Nouri, et al. Lasagne: First release., Aug. 2015.

[17] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[18] C. Gong, T. Liu, D. Tao, K. Fu, E. Tu, and J. Yang. Deformed graph laplacian for semisupervised learning. *IEEE transactions on neural networks and learning systems*, 26(10):2261–2274, 2015.

[19] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, pages 529–536, 2005.

[20] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer Vision and Pattern Recognition*, volume 2, pages 1735–1742. IEEE, 2006.

[21] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[22] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 200–209, 1999.

[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[26] A. Kumar, P. Sattigeri, and P. T. Fletcher. Improved semi-supervised learning with gans using manifold invariances. *arXiv preprint arXiv:1705.08850*, 2017.

[27] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[28] J. A. Lasserre, C. M. Bishop, and T. P. Minka. Principled hybrids of generative and discriminative models. In *Computer Vision and Pattern Recognition*, volume 1, pages 87–94. IEEE, 2006.

[29] R. Liao, A. Schwing, R. Zemel, and R. Urtasun. Learning deep parsimonious representations. In *Advances in Neural Information Processing Systems 29*, pages 5076–5084. 2016.

[30] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

[31] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013.

[32] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[33] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

[34] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth inter-*

national conference on Information and knowledge management, pages 86–93. ACM, 2000.

[35] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015.

[36] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[37] S. Rifai, Y. N. Dauphin, P. Vincent, Y. Bengio, and X. Muller. The manifold tangent classifier. In *Advances in Neural Information Processing Systems*, pages 2294–2302, 2011.

[38] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *Application of Computer Vision, 2005. WACV/MOTIONS '05 Volume 1. Seventh IEEE Workshops on*, volume 1, pages 29–36, Jan 2005.

[39] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1163–1171, 2016.

[40] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[41] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

[42] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[43] J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[44] A. Subramanya, S. Petrov, and F. Pereira. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176. Association for Computational Linguistics, 2010.

[45] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[46] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[47] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[48] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[49] B. Wang, Z. Tu, and J. K. Tsotsos. Dynamic label propagation for semi-supervised multi-class multi-label classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 425–432, 2013.

[50] J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. In *Proceedings of the 25th International Conference on Machine Learning (ICML-08)*, pages 1168–1175, 2008.

[51] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 321–328, 2004.

[52] X. Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4, 2006.

[53] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. *Technical Report CMU-CALD-02-107, Carnegie Mellon University*, 2002.

[54] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.