arXiv:1806.02679v1 [cs.LG] 7 Jun 2018

Semi-Supervised Learning via Compact Latent Space Clustering

Konstantinos Kamnitsas [1 2] **Daniel C. Castro** [1 2] **Loic Le Folgoc** [2] **Ian Walker** [2] **Ryutaro Tanno** [1 3]
Daniel Rueckert [2] **Ben Glocker** [2] **Antonio Criminisi** [1] **Aditya Nori** [1]

Abstract

We present a novel cost function for semi-supervised learning of neural networks that encourages compact clustering of the latent space to facilitate separation. The key idea is to dynamically create a graph over embeddings of labeled and unlabeled samples of a training batch to capture underlying structure in feature space, and use label propagation to estimate its high and low density regions. We then devise a cost function based on Markov chains on the graph that regularizes the latent space to form a single compact cluster per class, while avoiding to disturb existing clusters during optimization. We evaluate our approach on three benchmarks and compare to state-of-the art with promising results. Our approach combines the benefits of graph-based regularization with efficient, inductive inference, does not require modifications to a network architecture, and can thus be easily applied to existing networks to enable an effective use of unlabeled data.

1. Introduction

Semi-supervised learning (SSL) addresses the problem of learning a model by effectively leveraging both labeled and unlabeled data (Chapelle et al., 2006). SSL is effective when it results in a model that generalizes better than a model learned from labeled data only. More formally, let $\mathcal{X}$ be a sample space with data points and $\mathcal{Y}$ the set of labels (e.g., referring to different classes). Let $\mathcal{D}_L \subseteq \mathcal{X} \times \mathcal{Y}$ be a set of labeled data points, and let $\mathcal{D}_U \subseteq \mathcal{X}$ be a set of unlabeled data. In this work we focus on classification tasks, where for each $(\mathbf{x}, y) \in \mathcal{D}_L$, y is the ground truth label for sample $\mathbf{x}$. Our objective is to learn a predictive model $f(\mathbf{x};\theta) = p(y|\mathbf{x},\theta)$, parametrized by θ, which approximates the true conditional distribution $q(y|\mathbf{x})$ generating the target labels. SSL methods learn this by utilizing both $\mathcal{D}_L$ and $\mathcal{D}_U$, often assuming that $|\mathcal{D}_U| \gg |\mathcal{D}_L|$. Thus leveraging the ample unlabeled data allows capturing more faithfully the structure of data.

[1]Microsoft Research Cambridge, United Kingdom [2]Imperial College London, United Kingdom [3]University College London, United Kingdom. Correspondence to: Konstantinos Kamnitsas <konstantinos.kamnitsas12@imperial.ac.uk>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

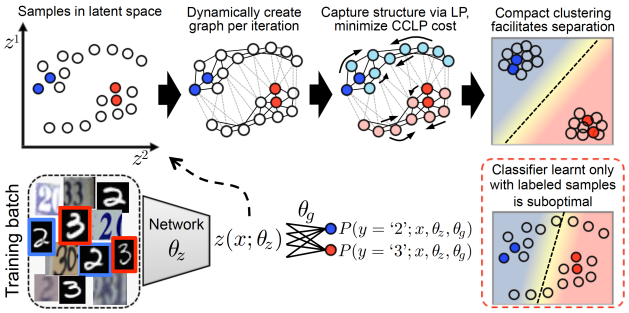

Figure 1. Overview of our method. We dynamically construct a graph in the latent space of a network at each training iteration, propagate labels to capture the manifold's structure, and regularize it to form a single, compact cluster per class to facilitate separation.

Various approaches to SSL have been proposed (see Section 2 for an overview). Underlying most of them is the notion of *consistency* (Zhou et al., 2004): samples that are close in feature space should be close in output space (*local consistency*) and samples forming an underlying structure should also map to similar labels (*global consistency*). This is the essence of the *smoothness* and *cluster assumptions* in SSL (Chapelle et al., 2006) that underpin our work. They respectively state that the label function should be smooth in high density areas of feature space, and points that belong to the same cluster should be of the same class. Hence decision boundaries should lie in low density areas.

We present a simple and effective SSL method for regularizing inductive neural networks (Fig. 1). The main idea is to dynamically create a graph in the network's latent space over samples in each training batch (containing both labeled and unlabeled data) to model the data manifold as it evolves during training. We then regularize the manifold's structure globally towards a more favorable state for class separation. We argue that the optimal feature space for classification should cluster all examples of a class to a

single, compact component, proposing a further constraint for SSL to those previously discussed: samples that map to the same class should belong to a single cluster. To learn such a latent space, we first use label propagation (LP) (Zhu & Ghahramani, 2002) as a proxy mechanism to estimate the arrangement of high/low density regions in latent space. This is in contrast to using LP as a transductive inference mechanism as done previously. We then propose a novel cost function, formulated via Markov chains on the graph, which not only brings together parts of the manifold with similar estimated LP posterior to form compact clusters, but also defines an optimization process that avoids disturbing existing high density areas, which are manifestations of information important for SSL.

We evaluate our approach on three visual recognition benchmarks: MNIST, SVHN and CIFAR10. Despite its simplicity, our method compares favorably to current state-of-the-art SSL approaches when labeled data is limited. Moreover, our regularization offers consistent improvements over standard supervision even when the whole labeled set is used. Our technique is computationally efficient and does not require additional network components. Thus it can be easily applied to existing models to leverage unlabeled data or regularize fully supervised systems.

2. Related Work

The great potential and practical implications of utilizing unlabeled data has resulted in a large body of research on SSL. The techniques can be broadly categorized as follows.

2.1. Graph-Based Methods

These methods operate over an input graph with adjacency matrix $\mathbf{A}$, where element A_{ij} is the similarity between samples $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}_L \cup \mathcal{D}_U$. Similarity can be based on Euclidean distance (Zhu & Ghahramani, 2002) or other, sometimes task-specific metrics (Weston et al., 2012). Transductive inference for the graph's unlabeled nodes is done based on the smoothness assumption, that nearby samples should have similar class posteriors. Label propagation (LP) (Zhu & Ghahramani, 2002) iteratively propagates the class posterior of each node to neighbors, faster through high density regions, until a global equilibrium is reached. Zhu et al. (2003) showed that for binary classification one arrives at the same solution by minimizing the energy:

$$E(\mathbf{f}) = \frac{1}{2} \sum_{i,j} A_{ij} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 = \mathbf{f}^\top \Delta \mathbf{f} \,. \quad (1)$$

Here, $\mathbf{f}$ is the vector with responses from predictor $f(\mathbf{x}) = p(y = 0|\mathbf{x})$ applied to all samples, Δ is the graph Laplacian. The solution being a harmonic function implies that the resulting posteriors for unlabeled nodes are the average of their neighbors (Zhu, 2005), showing that LP agrees with the smoothness assumption. Zhou et al. (2004) proposed a similar propagation rule and argued that predictions by propagation agree with the notion of global consistency. Many variations followed, such as the diffusion and graph convolutional networks (Atwood & Towsley, 2016; Kipf & Welling, 2017). These approaches are *transductive* and require a *pre-constructed* graph as a given, while their performance largely relies on the suitability of this given graph for the task. In contrast, we use LP not for transductive inference but as a sub-routine to estimate the structure of the clusters in a network's latent space. We then regularize the network's feature extractor, which *learns* an appropriate graph consistent with the smoothness property of LP, while preserving the network's efficient *inductive* classifier.

Equation (1) has also been used to define the graph Laplacian regularizer, which has been used for SSL of inductive models (Belkin et al., 2006; Weston et al., 2012). However, these methods still require a pre-constructed graph. A recent method inspiring our work relaxes this requirement and seeks associations between labeled and unlabeled data (Haeusser et al., 2017). This is modeled through the probability that a two-step random walk in the hidden embedding of a neural network would start and end at labeled samples of the same class, via one intermediate unlabeled point. But this formulation does not capture the global structure of the clusters and can collapse to the trivial solution of associating an unlabeled point to its closest cluster in Euclidean space (Haeusser et al., 2017). Hence, a second regularizer is required to keep all samples relatively close.

2.2. Self-Supervision and Entropy Minimization

One of the earliest ideas for leveraging unlabeled data is self-supervision or self-learning. It is a wrapper framework in which a classifier trained with supervision periodically classifies the unlabeled data, and confidently classified samples are added to the training set. The idea dates back to Scudder (1965) and saw multiple extensions. The method is heavily dependent on classifier's performance. It gained popularity recently for training neural networks (Lee, 2013), enabled by their overall good performance. Relevant is co-training (Blum & Mitchell, 1998), which uses confident predictions of two classifiers trained on distinct views of the data.

Closely related is regularization via conditional entropy minimization (Grandvalet & Bengio, 2005). Model parameters θ are learned by minimizing entropy in the prediction $H(y|\mathbf{x}, \theta) = \mathbb{E}\left[-\log p(y|\mathbf{x}, \theta)\right]$ for each unlabeled sample $\mathbf{x}$, additionally to the supervised loss. It can be seen as an efficient information-theoretic form of self-supervision, encouraging the model to make confident predictions. This pushes samples away from decision boundaries and vice-versa, favoring low-density separation. It may however in-

duce confirmation bias, hurting optimization if clusters are not yet well formed. Such is the case of a neural network's embedding in early training stages, where gradient descent can push samples away from the decision boundary towards the random side where they started (Fig. 2). Because of this, the regularizer's effect is commonly controlled with ad-hoc ramp-up schedules of a weight meta-parameter (Springenberg, 2015; Chongxuan et al., 2017; Dai et al., 2017; Miyato et al., 2017). Similar is the case of self-supervision. In contrast, our regularizer does not use the suboptimal classifier being trained. It only reasons about the latent manifold's geometry. As a result, gradients it applies are indifferent to the decision boundary's position and do not generally oppose gradients of the classification loss. Finally, since our cost depends on the confidence of labels propagated on the graph, its effect adapts throughout training, according to whether clusters are well formed or not.

2.3. Perturbation-Based Approaches

Regularizing the input-output mapping to be consistent when noise is applied to the input can improve generalization (Bishop, 1995). This goal of "consistency under perturbation" has been shown applicable for SSL (Bachman et al., 2014). In its generic form, a function f minimizes a regularizer of the form $R(f) = \mathbb{E}_\xi \left[d(f(\mathbf{x};\xi), f(\mathbf{x}))\right]$ for each sample $\mathbf{x}$, assuming ξ is a noise process such that $\mathbb{E}_\xi \left[f(\mathbf{x};\xi)\right] = f(\mathbf{x})$. f can be the classification output or hidden activations of a neural network, d is a distance metric such as the L_2 norm. This cost encourages *local* consistency of the classifier's output around each unlabeled sample, pushing decision boundaries away from high density areas. The approach has given promising results, with ξ taking various forms such as different dropout masks (Bachman et al., 2014), Gaussian noise applied to network activations (Rasmus et al., 2015), sampling input augmentations, predictions from models at different stages of training (Laine & Aila, 2017; Tarvainen & Valpola, 2017) or adversarial perturbation (Miyato et al., 2017). Like self-supervision, these methods can induce confirmation bias. Orthogonally to encouraging local smoothness around each individual sample, our method regularizes geometry of the manifold globally by treating all samples and their connections jointly.

2.4. Generative Models

Generative models have also been used within SSL frameworks. In particular, probabilistic models such as Gaussian mixtures (McLachlan, 2004) are representative examples. These approaches model how samples $\mathbf{x}$ are generated, estimating $p(\mathbf{x}|y)$ or the joint distribution $p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$. In this framework, SSL can be modeled as a missing data (y) problem. This is however a substantially more general problem than estimating $p(y|\mathbf{x})$ with a discriminative model. One might argue that estimating the joint distribution is not the best objective for SSL, as it requires models of unnecessarily large representational power and complexity. Examples of popular neural models are auto-encoders (AE) (Ranzato & Szummer, 2008; Rasmus et al., 2015) and variational auto-encoders (VAE) (Kingma et al., 2014; Maaløe et al., 2016). Unfortunately, spending the encoder's capacity on preserving variation of the input that is potentially unrelated to label y, as well as the requirement for a similarly powerful decoder, make these approaches difficult to scale to large and complex databases.

Generative adversarial networks (GAN) (Goodfellow et al., 2014) have been recently applied to SSL with promising results. Conditional GANs were used to generate synthetic samples $(\mathbf{x}, y)$, which can serve as additional training data (Chongxuan et al., 2017). Salimans et al. (2016) encouraged the discriminator to identify the class of real samples, aside from distinguishing real from fake inputs. Similarly, in CatGAN (Springenberg, 2015) the discriminator minimizes the conditional entropy of $p(y|\mathbf{x})$ for real but maximizes it for fake samples. The reason why the classification objective gains from the real-versus-fake discrimination was analyzed in Dai et al. (2017). Interestingly, rather than directly benefiting from modeling the generative process, it was shown that bad examples from the generator that lie in low-density areas of the data manifold guide the classifier to better position its decision boundary, thus connecting the improvements with the cluster assumption. Promising results were achieved, yet the requirement for a generator and the challenges of adversarial optimization leave space for future work. Note that these methods are orthogonal to ours, which regularizes the latent manifold's structure.

3. Method

Our work builds on the *cluster assumption*, whereby samples forming a structure are likely of the same class (Chapelle et al., 2006), by enforcing a further constraint: *All samples of a class should belong to the same structure*.

In this work we take the labeling function $f(\mathbf{x};\theta)$ to be a multi-layer neural network. This model can be decomposed into a *feature extractor* $z(\mathbf{x};\theta_z) \in \mathcal{Z}$ parametrized by θ_z, and a *classifier* $g(z(\mathbf{x};\theta_z);\theta_g)$ with parameters θ_g. The former typically consists of all hidden layers of the network, while the latter is the final linear classifier. We argue that classification is improved whenever data from each class form compact, well separated clusters in *feature space* $\mathcal{Z}$. We use a graph embedding to capture the structure of data in this latent space and propagate labels to unlabeled samples through high density areas (Section 3.1). We then introduce a regularizer (Section 3.2) that 1) encourages compact clustering according to propagated labels and 2) avoids disturbing existing clusters during optimization (Fig. 1).

3.1. Estimating Structure of Data via Dynamic Graph Construction and Label Propagation

We train f with stochastic gradient descent (SGD), sampling at each SGD iteration a labeled batch $(\mathbf{X}_L, \mathbf{y}_L) \sim \mathcal{D}_L$ of size N_L and an unlabeled batch $\mathbf{X}_U \sim \mathcal{D}_U$ of size N_U. Let $\mathbf{Y}_L \in \mathbb{R}^{N_L \times C}$ be one-hot representation of $\mathbf{y}_L$ with C classes. The feature extractor of the network produces the embeddings $\mathbf{Z}_L = z(\mathbf{X}_L; \theta_z)$ for labeled and $\mathbf{Z}_U = z(\mathbf{X}_U; \theta_z)$ for unlabeled data. We propose to dynamically create a graph at every SGD iteration over the embedding $\mathbf{Z} = \left[\begin{smallmatrix}\mathbf{Z}_L \\ \mathbf{Z}_U\end{smallmatrix}\right]$ of batch $\mathbf{X} = \left[\begin{smallmatrix}\mathbf{X}_L \\ \mathbf{X}_U\end{smallmatrix}\right]$, and use label propagation (LP) (Zhu & Ghahramani, 2002) to implicitly capture the structure of each class. Unlike Euclidean metrics, graph-based metrics naturally respect the underlying data distribution, following paths along high density areas.

We first generate a fully connected graph in feature space from both labeled and unlabeled samples. The graph is characterized by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where $N = N_L + N_U$. Each element A_{ij} is the weight of an edge between samples i and j representing their similarity, and is parametrized as

$$A_{ij} = \exp(\rho(\mathbf{z}_i, \mathbf{z}_j)) \,, \quad \forall \mathbf{z}_i, \mathbf{z}_j \in \mathbf{Z}_L \cup \mathbf{Z}_U \,, \tag{2}$$

where $\rho : \mathcal{Z}^2 \to \mathbb{R}$ is a similarity score such as the dot product or negative Euclidean distance. In this paper we use the former. The Markovian random walk along the nodes of this graph is defined by its transition matrix $\mathbf{H}$, obtained by row-wise normalization[1] of $\mathbf{A}$. Each element H_{ij} is the probability of a transition from node i to node j:

$$H_{ij} = A_{ij} \Big/ \sum_k A_{ik} \,. \tag{3}$$

Without loss of generality, elements of $\mathbf{A}$ and $\mathbf{H}$ that correspond to labeled samples L and unlabeled samples U are arranged so that

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{LL} & \mathbf{H}_{UL} \\ \mathbf{H}_{LU} & \mathbf{H}_{UU} \end{bmatrix} . \tag{4}$$

LP uses $\mathbf{H}$ to model the process of a node i propagating its class posterior $\boldsymbol{\phi}_i = p_{\text{LP}}(y|\mathbf{x}_i, \mathbf{A}) \in \mathbb{R}^C$ to the other nodes. One such propagation step is formally given by $\boldsymbol{\Phi}^{(t+1)} = \mathbf{H}\boldsymbol{\Phi}^{(t)}$, where $\boldsymbol{\Phi}^{(t)} \in \mathbb{R}^{N \times C}$. As a result, class confidence propagates from labeled to unlabeled samples. While propagation to nearby points initially dominates due to the exponential in Eq. (2), multiple iterations of the algorithm allow soft labels $\boldsymbol{\Phi}^{(t)}$ to propagate and eventually traverse the whole graph in the stationary state. Unlike *diffusion* (Kondor & Lafferty, 2002), LP interprets labeled samples as *constant sources* of labels, and clamps their confidence to their true value $\mathbf{Y}_L$, thus $\boldsymbol{\Phi}^{(t)} = \left[\begin{smallmatrix}\mathbf{Y}_L \\ \boldsymbol{\Phi}_U^{(t)}\end{smallmatrix}\right], \forall t$. Hence class confidence gradually accumulates in the graph. By propagating more easily through high density areas, the process converges at an equilibrium where the decision boundary settles in a low-density area. In practice the soft labels for the unlabeled data at the equilibrium can be conveniently computed in closed form (Zhu & Ghahramani, 2002) without the need for iterations as

$$\boldsymbol{\Phi}_U = (\mathbf{I} - \mathbf{H}_{UU})^{-1} \mathbf{H}_{UL} \mathbf{Y}_L \,. \tag{5}$$

Hereafter, let $\boldsymbol{\Phi} = \left[\begin{smallmatrix}\mathbf{Y}_L \\ \boldsymbol{\Phi}_U\end{smallmatrix}\right] \in \mathbb{R}^{N \times C}$ denote the class posteriors estimated by LP at convergence, i.e. the concatenation of the true (clamped) labels for $\mathbf{X}_L$ and the estimated soft labels for $\mathbf{X}_U$. Equation (5) has been previously used for *transductive* inference in applications where the graph is given *a priori* (see Section 2.1), hence results directly rely on *suitability* of LP as a predictor and of the graph. In contrast, we build the graph in a feature space $\mathcal{Z}$, which will be *learned* so that it simultaneously complies with the properties of LP while serving the optimization objectives. We also emphasize that instead of relying on it for inference, in our framework LP merely provides a mechanism for capturing the shape of clusters in latent space to regularize them towards a desired stationary point. This improved embedding will benefit generalization of the actual classifier g, which is trained with standard cross entropy on labeled samples $\mathbf{X}_L$, retaining its efficient *inductive* nature.

3.2. Encouraging Compact Clusters in Feature Space

Our desiderata for an optimal SSL regularizer are as follows: 1) it encourages formation of a single and compact cluster per class in latent space, so that linear separation is straightforward; 2) it must be compatible with the supervised loss to allow easy optimization and high performance.

We first observe that in the desired optimal state, where a single, compact cluster per class has been formed, the transition probabilities between any two nodes within a cluster should be the same, and zero for inter-cluster jumps. Motivated by this, we define a probabilistic version of this optimal transition matrix $\mathbf{T}$ as:

$$T_{ij} = \sum_{c=1}^{C} \phi_{ic} \frac{\phi_{jc}}{m_c} \,, \quad m_c = \sum_{i=1}^{N} \phi_{ic} \,. \tag{6}$$

Here ϕ_{ic} is the LP posterior for node i to belong to class c and m_c is the expected mass assigned to class c. We can encourage z to form compact clusters by minimizing cross entropy between $\mathbf{T}$ and the current transition matrix $\mathbf{H}$:

$$\mathcal{L}_{\text{1-step}} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} -T_{ij} \log H_{ij} \,. \tag{7}$$

[1] We note that other LP variants such as Zhou et al. (2004) that uses symmetrically normalized Laplacian could also be used.

Algorithm 1 Training for SSL with CCLP

Input: feature extractor $z(\cdot;\theta_z)$, classifier $g(\cdot;\theta_g)$, data $\mathcal{D}_L$, $\mathcal{D}_U$, batch sizes N_L, N_U, $N = N_L + N_U$ Markov chain steps S, weighting w, learning rate α
Output: Learnt network parameters θ_z and θ_g
repeat
  $(\mathbf{X}_L, \mathbf{y}_L) \overset{N_L}{\sim} \mathcal{D}_L,\ \mathbf{X}_U \overset{N_U}{\sim} \mathcal{D}_U,\ \mathbf{X} = \left[\begin{smallmatrix}\mathbf{X}_L\\ \mathbf{X}_U\end{smallmatrix}\right]$ # Samples
  $\mathbf{Y}_L \leftarrow \texttt{one_hot}(\mathbf{y}_L)$ # Labels
  $\mathbf{Z} \leftarrow z(\mathbf{X};\theta_z)$ # Forward pass
  $\mathcal{L}_{\text{sup}} \leftarrow -\frac{1}{N_L}\sum_{i=1}^{N_L}\sum_{c=1}^{C} y_{ic}\log[g(\mathbf{z}_i;\theta_g)]_c$
  $\mathbf{A} \leftarrow \exp(\mathbf{Z}\mathbf{Z}^\mathsf{T})$ # Graph
  $\mathbf{H} \leftarrow$ row normalized $\mathbf{A}$ # Transition matrix
  $\mathbf{\Phi}_U \leftarrow (\mathbf{I} - \mathbf{H}_{UU})^{-1}\mathbf{H}_{UL}\mathbf{Y}_L$ # LP
  $\mathbf{\Phi} \leftarrow [\mathbf{Y}_L; \mathbf{\Phi}_U]$
  $\mathbf{T} \leftarrow$ according to Eq. (6)
  $\mathbf{M} \leftarrow \mathbf{\Phi}\mathbf{\Phi}^\mathsf{T}$ # Labels agreement
  $\mathcal{L}_{\text{CCLP}} \leftarrow 0$
  for $s = 1$ **to** S **do**
    $\mathbf{H}^{(s)} \leftarrow (\mathbf{H} \circ \mathbf{M})^{s-1}\mathbf{H}$
    $\mathcal{L}_{\text{CCLP}} \leftarrow \mathcal{L}_{\text{CCLP}} - \frac{1}{SN^2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}\log H_{ij}^{(s)}$
  end for
  $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{sup}} + w\mathcal{L}_{\text{CCLP}}$
  $\theta_z \leftarrow \theta_z - \alpha\frac{\partial\mathcal{L}_{\text{total}}}{\partial\theta_z}, \quad \theta_g \leftarrow \theta_g - \alpha\frac{\partial\mathcal{L}_{\text{sup}}}{\partial\theta_g}$ # Updates
until stopping criterion is $true$

This objective has properties that are desirable in SSL. It considers unlabeled samples, models high and low density regions via the use of LP, and facilitates separation by attracting together in one compact cluster samples of the same soft or hard labels while repulsing different ones. It does not apply strong forces to unconfident samples, to avoid problematic optimization when embedding is still suboptimal, e.g. in early training. By being unaware of g and its decision boundary, gradients of Eq. (7) only depend on the manifold's geometry, thus they do not oppose those from the supervised loss, unlike methods suffering from confirmation bias (Section 2). We argue that one more property is important for good optimization, which is not yet covered.

During optimization, forces applied by a cost should not disturb existing clusters, as they contain information that enables SSL via the cluster assumption. To model such behavior we design a cost that attracts points of the same class along the structure of the graph. For this we extend the regularizer of Eq. (7) to the case of Markov chains with multiple transitions between samples, which should remain within a single class. The probability of a Markov process with transition matrix $\mathbf{H}$ starting at node i and landing at node j after s number of steps is given by $(\mathbf{H}^s)_{ij}$.

We are interested in modeling transitions within the same class and increase their probability, while minimizing the probability of transiting to other clusters. Our solution is to utilize the class posteriors estimated by LP, to define a confidence metric that nodes belong to the same class. For this, we use the dot product of the nodes' LP class posteriors $\mathbf{M} = \mathbf{\Phi}\mathbf{\Phi}^\mathsf{T}$. The convenient property of this choice is that the elements of $\mathbf{M}$ are bounded in the range $[0, 1]$, taking the maximum and minimum values if and only if the labels (hard/soft for $\mathbf{X}_L$/$\mathbf{X}_U$ respectively) fully agree or disagree respectively. This allows us to use it as an estimate of the probability that two nodes belong to the same class.

Equipped with $\mathbf{M}$, we estimate the joint probability of transitioning from node i to node j and the two belonging to the same class as $p(i \to j, y_i = y_j) = p(i \to j)p(y_i = y_j | i \to j) \approx H_{ij}M_{ij}$. Note that $\mathbf{M}$ is indeed a function of $\mathbf{H}$, as suggested by the conditional it estimates. Finally, we estimate the probability of a Markov process to start from node i, perform s steps within the same class and then transit to *any* node j, as the element $H_{ij}^{(s)}$ of the matrix

$$\mathbf{H}^{(s)} = (\mathbf{H} \circ \mathbf{M})^{s-1}\mathbf{H} = (\mathbf{H} \circ \mathbf{M})\mathbf{H}^{(s-1)}, \tag{8}$$

where $\circ$ denotes the Hadamard (elementwise) product.

By regularizing $\mathbf{H}^{(s)}$ towards target matrix $\mathbf{T}$ as in Eq. (7), we minimize the probability of a chain transiting between clusters of different classes, thus repulsing them, and encourage uniform probability for chains that only traverse samples of one class, which attracts them and promotes compact clustering. Notably, regularizing $\mathbf{H}^{(s)}$ of the latter type of chains towards larger values discourages disturbing clusters along their path, as this would push $\mathbf{H}^{(s)}$ close to zero. This motivates the final form of our *Compact Clustering via Label Propagation* (CCLP) regularizer:

$$\mathcal{L}_{\text{CCLP}} = \frac{1}{S}\sum_{s=1}^{S}\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} -T_{ij}\log H_{ij}^{(s)}. \tag{9}$$

This cost consists of S terms, each modeling paths of different length between samples on the graph. Larger s allows Markov chains to traverse along more elongated clusters. Note that this cost subsumes Eq. (7) for $s = 1$. Equation (9) is minimized simultaneously with the supervised loss $\mathcal{L}_{\text{sup}}$. An overview of our method is shown in Algorithm 1.

4. Empirical Analysis on Synthetic Data

We conduct a study on synthetic toy examples to analyze the behavior of the proposed method. We are interested in the forces that CCLP applies to samples in the latent space $\mathcal{Z}$ as it attempts to improve their clustering, *isolated* from the influence of model $z(\cdot;\theta_z)$. Hence we do not adopt common visualization methods that map space $\mathcal{Z}$ learned by a network to 2D (Maaten & Hinton, 2008), or plot the decision boundary of the total model f in input space $\mathcal{X}$.

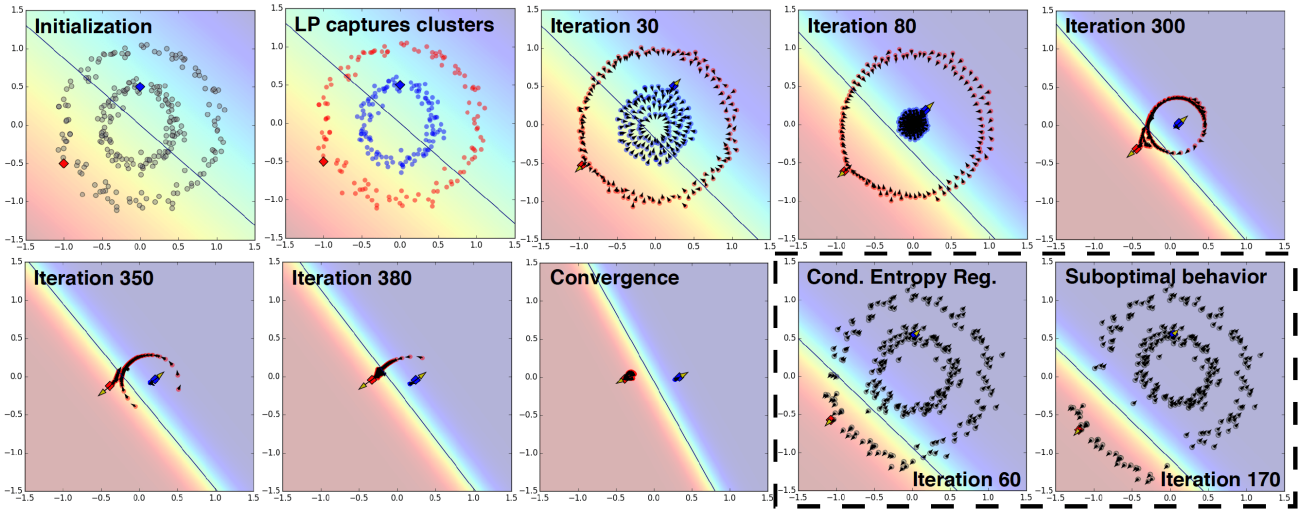

Figure 2. Two-circles toy experiment. Main figure shows the initial arrangement of two labeled (red/blue) and multiple unlabeled points, label propagation, and iterations using CCLP along with supervision until convergence. Also depicted are gradients by the supervised loss on labeled samples (yellow arrows) and by CCLP (black arrows). The dashed box shows failure case of conditional entropy regularizer.

To isolate the effect of CCLP, we consider an artificial setup in which assumed embeddings of samples $\mathbf{Z}$ are initially positioned in a structured arrangement in a 2D space, which represents $\mathcal{Z}$, and are allowed to move freely. We place the embeddings in commonly used toy layouts: two-moons and two-circles. For the role of $g(\mathbf{Z};\theta_g)$, we use a linear classifier, for which we compute the supervised loss $\mathcal{L}_{\text{sup}}$. We then perform label propagation on this artificial latent space $\mathcal{Z}$ and compute $\mathcal{L}_{\text{CCLP}}$. Finally, we compute the gradients of the two costs with respect to θ_g and the coordinates of the embeddings $\mathbf{Z}$, and update them iteratively.

In this setting, both costs try to move the labeled samples in space $\mathcal{Z}$, but only CCLP affects unlabeled data. If $\mathbf{Z}$ were computed by a real neural net $z(\cdot;\theta_z)$, which is a smooth function, embeddings of unlabeled samples would also be affected by $\mathcal{L}_{\text{sup}}$, via updates to θ_z. Our settings instead isolate the effect of CCLP on the unlabeled data.

4.1. Two Circles

We study the dynamics of CCLP ($S = 10$) on two-circles (Fig. 2), when a single labeled example is given per class. We first observe that the isolated effect of CCLP indeed encourages formation of a single, compact cluster per class. In more challenging scenarios, results are naturally subject to the effect of the model, optimizer and data.

We observe that the direction of gradients applied by CCLP to each sample depends on the manifold's geometry. Since gradients from the supervised loss are perpendicular to the decision boundary of the linear classifier, the effect of CCLP generally does not oppose supervision. By contrast, we show the effect of confirmation bias by studying conditional entropy regularization (CER) (Grandvalet & Bengio, 2005). CER gradients are perpendicular to the decision boundary and can thus oppose the effect of supervision.[2]

4.2. Two Moons

We use two moons to investigate the effect of the maximum steps S of Markov chains used in $\mathcal{L}_{\text{CCLP}}$ (Fig. 3). When multiple steps are used, here $S = 10$, gradients of CCLP follow existing high density areas in their attempt to cluster samples better. This leads the labeled samples to also move along the existing clusters on their way to the correct side of the decision boundary. Conversely, when a single step is used ($S = 1$), gradients by CCLP try to preserve only local neighborhoods, which allows the clusters to disintegrate. This breakdown of the global structure implies loss of information, which in turn may lead to misclassification.

5. Evaluation on Common Benchmarks

Benchmarks: We consider three benchmarks widely used in studies on SSL: MNIST, SVHN and CIFAR-10. We whiten the datasets, following common practice, and use no data augmentation, to isolate the effect of the SSL method. Following previous work, to study the effectiveness of CCLP when labeled data is scarce, as $\mathcal{D}_L$ we use 100, 1000 and 4000 samples from the training set of each benchmark respectively, while the whole training set without its labels

[2] If combined with an appropriate model z or a different optimizer, CER could solve this example. Here we focus on the effect under gradient descent and independently of the model z.

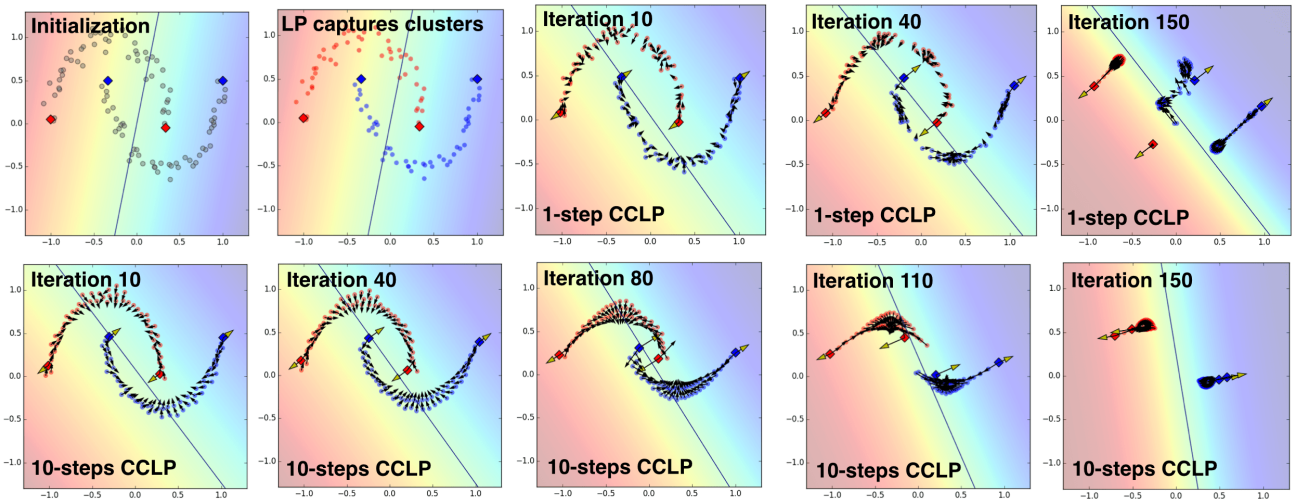

Figure 3. Two-moons toy experiment. Comparison between CCLP applied with $S=1$ (*top row*) and with $S=10$ (*bottom row*). Exploring the direction of the gradients from CCLP (black arrows) shows that optimizing over a longer chain of steps leads to a behavior that tries to preserve existing clusters when attempting to create more compact clusters.

Table 1. Performance of CCLP compared to contemporary SSL methods on common benchmarks, when limited or all available labelled data is used as $\mathcal{D}_L$ for training. Also shown is performance of the corresponding baseline with standard supervision (no SS). Error rate is shown as (mean ± st.dev.). Only results obtained without augmentation are shown. Methods in the lower part used larger classifiers.

	MNIST			SVHN			CIFAR10		
MODEL	$\lvert\mathcal{D}_L\rvert = 100$	ALL	ALL, NO SS	1000	ALL	ALL, NO SS	4000	ALL	ALL, NO SS
CONV-CATGAN(SPRINGENBERG, 2015)	1.39 ± 0.28	0.48	–	–	–	–	19.58 ± 0.46	9.38	–
LADDER(CNN-Γ)(RASMUS ET AL., 2015)	0.89 ± 0.50	–	0.36	–	–	–	20.40 ± 0.47	–	9.27
SDGM (MAALØE ET AL., 2016)	1.32 ± 0.07	–	–	16.61 ± 0.24	–	–	–	–	–
ADGM (MAALØE ET AL., 2016)	0.96 ± 0.02	–	–	22.86	–	–	–	–	–
IGAN (SALIMANS ET AL., 2016)	0.93 ± 0.07	–	–	8.11 ± 1.30	–	–	18.63 ± 2.32	–	–
ALI (DUMOULIN ET AL., 2017)	–	–	–	7.42 ± 0.65	–	–	17.99 ± 1.62	–	–
VAT (MIYATO ET AL., 2017)	1.36	0.64	1.11	6.83	–	–	14.87	5.81	6.76
TRIPLE GAN (CHONGXUAN ET AL., 2017)	0.91 ± 0.58	–	–	5.77 ± 0.17	–	–	16.99 ± 0.36	–	–
MMCVAE (LI ET AL., 2017)	1.24 ± 0.54	0.31	–	4.95 ± 0.18	3.09	–	–	–	–
BADGAN (DAI ET AL., 2017)	0.80 ± 0.10	–	–	4.25 ± 0.03	–	–	14.41 ± 0.30	–	–
LBA (HAEUSSER ET AL., 2017)	0.89 ± 0.08	0.36 ± 0.03	–	–	–	–	–	–	–
LBA (OUR IMPLEMENTATION)	0.90 ± 0.10	0.36 ± 0.03	0.46 ± 0.03	9.25 ± 0.65	3.61 ± 0.10	4.26 ± 0.10	19.33 ± 0.51	8.46 ± 0.18	9.33 ± 0.14
CCLP (OURS)	0.75 ± 0.14	0.32 ± 0.03	0.46 ± 0.03	5.69 ± 0.28	3.04 ± 0.05	4.26 ± 0.10	18.57 ± 0.41	8.04 ± 0.18	9.33 ± 0.14
Larger Classifiers									
Π MODEL (LAINE & AILA, 2017)	–	–	–	5.43 ± 0.25	–	–	16.55 ± 0.29	–	–
MTEACH.(TARVAINEN & VALPOLA, 2017)	–	–	–	5.21 ± 0.21	2.77 ± 0.09	3.04 ± 0.04	17.74 ± 0.29	7.21 ± 0.24	7.43 ± 0.06
VAT-LARGE (MIYATO ET AL., 2017)	–	–	–	5.77	–	–	14.82	–	–
VAT-LARGE-ENT (MIYATO ET AL., 2017)	–	–	–	4.28	–	–	13.15	–	–

constitutes $\mathcal{D}_U$. We also study effectiveness of our method when abundant labels are available, using the whole training set as both $\mathcal{D}_L$ and $\mathcal{D}_U$. We also report performance of our baseline trained with only standard supervision (no SSL), to facilitate comparison of improvements from CCLP with previous and future works, where quality of the baselines may differ. For every benchmark, we perform 10 training sessions with random seeds and randomly sampled $\mathcal{D}_L$ and report the mean and standard deviation of the error. We evaluate on the test-dataset of each benchmark, except for the ablation study where we separated a validation set.

Models: For MNIST we use a CNN similar to Rasmus et al. (2015); Chongxuan et al. (2017); Haeusser et al. (2017); Li et al. (2017). For SVHN and CIFAR we use the network used as classifier in Salimans et al. (2016), commonly adopted in recent works. In all experiments we used the same meta-parameters for CCLP: In each SGD iteration we sample a batch $(\mathbf{X}_L, \mathbf{y}_L) \sim \mathcal{D}_L$ of size $N_L = 100$, where we ensure that 10 samples from each class are contained, and a batch without labels $\mathbf{X}_U \sim \mathcal{D}_U$ of size $N_U = 100$. We use the dot product as similarity metric (Eq. (2)), $S=3$ maximum steps of the Markov chains (Eq. (9)). $\mathcal{L}_{\text{CCLP}}$ was weighted equally with the supervised loss, with $w=1$ throughout training. These parameters were found to work reasonably in early experiments on a pre-selected validation subset from the MNIST training set and were used without extensive effort to optimize them for this benchmarking. Exception are the experiments with $|\mathcal{D}_L|=4000$ on CIFAR, where lower $w=0.1$ was used, because $w=1$ was found to over-regularize these settings.

We also employ the method of Haeusser et al. (2017) (LBA) on SVHN and CIFAR-10, as in the original work a different network was used, while results on SVHN where reported only with data augmentation. We note that for correctness, in preliminary experiments we ensured that with data augmentation our implementation of LBA produced similar results to what reported in Haeusser et al. (2017).

Results: Performance of our method in comparison to recent SSL approaches that use similar experimental settings are reported in Table 1. We do not report results obtained with data augmentation. Note that iGAN (Salimans et al., 2016), VAT (Miyato et al., 2017) and BadGAN (Dai et al., 2017) used a deep MLP instead of a CNN on MNIST, so those results may not be entirely comparable. Our method achieves very promising results in all benchmarks, that improve or are comparable to the state-of-the-art. CCLP consistently improves performance over standard supervision even when all labels in the training set are used for supervision, indicating that CCLP could be used as a latent space regularizer in fully supervised systems. In the latter settings, CCLP offers greater improvement over the corresponding baselines than the most recent perturbation-based method, mean teacher (Tarvainen & Valpola, 2017). We finally emphasize that our method consists of the computation of a single cost function and does not require additional network components, such as the generators required for VAEs and GANs, or the density estimator PixelCNN++ used in Dai et al. (2017). Furthermore, we emphasize that many of these works make use of multiple complementary regularization costs. The compact clustering that our method encourages is orthogonal to previous approaches and could thus boost their performance further.

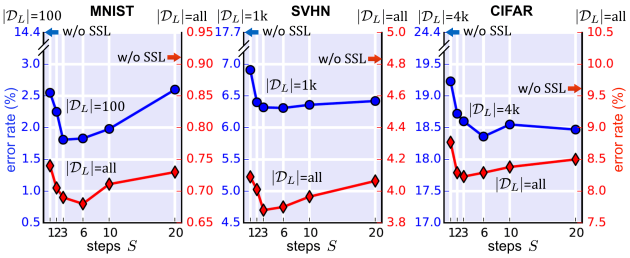

Figure 4. Validation error when CCLP is applied with varying number of steps S. Left/right vertical axis correspond to training with limited/all labeled samples respectively. Compact clustering with $S=1$ improves over standard supervision (w/o SSL). Optimizing with 3-6 steps offers a further 5-25% reduction of the error.

Ablation study: We further study the effect of CCLP's two key aspects: Regularizing the latent space towards compact clustering and, secondly, optimizing while respecting existing clusters by using multi-step Markov chains. For this, we separate a validation set of 10000 images from the training set of each benchmark. $\mathcal{D}_L$ and $\mathcal{D}_U$ are formed out of the remaining training data. We evaluate performance on the validation set when CCLP uses different number of maximum steps S (Eq. (9)). Each setting is repeated 10 times and we report the average error in Fig. 4. When $S=1$, CCLP encourages compact clustering without attempting to preserve existing clusters. This already offers large benefits over standard supervision. Optimizing over longer Markov chains offers further improvements, with values $3 \leq S \leq 6$ further reducing the error by 5-25% in most settings. Capturing too long paths between samples ($S>10$) reduces the benefits.

6. Computational Considerations

Time complexity of CCLP is $O(N^3+SN^2)$, overwhelmed by $O(N^3)$ of matrix inversion since $N \gg S$ in our settings. In practice, CCLP is inexpensive compared to a net's forward and backward passes. In our CIFAR settings and TensorFlow GPU implementation (Abadi et al., 2016), CCLP increases less than 10% the time for an SGD iteration, even for large $N=1000$. In comparison, GANs and VAEs require an expensive decoder, while perturbation-based approaches perform multiple passes over each sample.

As batch size N defines how well the graph approximates the true data manifold, larger N is desirable but requires more memory, while low N may decrease performance. Batch sizes in the order of 200 used in this and previous works (Laine & Aila, 2017) are practical in various applications, with hardware advances promising further improvements. Finally, in distributed systems that divide thousands of samples between compute nodes, CCLP could scale by creating a different graph per node.

7. Conclusion

We have presented a novel regularization technique for SSL, based on the idea of forming compact clusters in the latent space of a neural network while preserving existing clusters during optimization. This is enabled by dynamically constructing a graph in latent space at each SGD iteration and propagating labels to estimate the manifold's structure and to then regularize it. We show that our approach is effective in leveraging unlabeled samples via empirical evaluation on three widely used image classification benchmarks. We also showed our regularizer offers consistent improvements over standard supervision even when labels are abundant. Our method is computationally efficient and easy to apply to existing architectures as it does not require additional network components. Analyzing further the properties of a compactly clustered latent space, as well as applying our approach to larger benchmarks and the task of semantic segmentation is interesting future work.

Acknowledgements

This research was partly carried out when KK, DC and RT were interns at Microsoft Research Cambridge. This project has also received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 757173, project MIRA, ERC-2017-STG). KK is also supported by the President's PhD Scholarship of Imperial College London. DC is supported by CAPES, Ministry of Education, Brazil (BEX 1500/15-05). LLF is funded through EPSRC Healthcare Impact Partnerships grant (EP/P023509/1). IW is supported by the Natural Environment Research Council (NERC).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI 2016)*, volume 16, pp. 265–283, 2016.

Atwood, J. and Towsley, D. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1993–2001, 2016.

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

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

Bishop, C. M. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995.

Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pp. 92–100, 1998.

Chapelle, O., Scholkopf, B., and Zien, A. *Semi-supervised Learning*. MIT Press, Cambridge, Mass., USA, 2006.

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

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

Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., and Courville, A. Adversarially learned inference. In *International Conference on Learning Representations*, 2017.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

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

Haeusser, P., Mordvintsev, A., and Cremers, D. Learning by association – a versatile semi-supervised training method for neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

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

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017.

Kondor, R. I. and Lafferty, J. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th International Conference on Machine Learning*, volume 2, pp. 315–322, 2002.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. *International Conference on Learning Representations*, 2017.

Lee, D.-H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, pp. 2, 2013.

Li, C., Zhu, J., and Zhang, B. Max-margin deep generative models for (semi-) supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. Auxiliary deep generative models. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, pp. 1445–1454, 2016.

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

McLachlan, G. *Discriminant Analysis and Statistical Pattern Recognition*, volume 544. John Wiley & Sons, 2004.

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

Ranzato, M. and Szummer, M. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 792–799, 2008.

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

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

Scudder, H. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

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

Tarvainen, A. and Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*, pp. 1195–1204, 2017.

Weston, J., Ratle, F., Mobahi, H., and Collobert, R. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pp. 639–655. Springer, 2012.

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

Zhu, X. *Semi-supervised Learning with Graphs*. PhD thesis, Carnegie Mellon University, Language Technologies Institute, School of Computer Science, 2005.

Zhu, X. and Ghahramani, Z. Learning from labeled and unlabeled data with label propagation. *Technical Report, Carnegie Mellon University*, 2002.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pp. 912–919, 2003.