Autoencoder-based Graph Construction for Semi-supervised Learning

†Mingeun Kang[1][0000−0003−4375−4032], †Kiwon Lee[1][0000−0003−2658−4342], Yong H. Lee[2][0000−0001−9576−8554], and Changho Suh[1][0000−0002−3101−4291]

[1] KAIST, Daejeon, South Korea
{minkang23,kaiser5072,chsuh}@kaist.ac.kr
[2] UNIST, Ulsan, South Korea
yohlee@unist.ac.kr

Abstract. We consider graph-based semi-supervised learning that leverages a similarity graph across data points to better exploit data structure exposed in unlabeled data. One challenge that arises in this problem context is that conventional matrix completion which can serve to construct a similarity graph entails heavy computational overhead, since it re-trains the graph independently whenever model parameters of an interested classifier are updated. In this paper, we propose a holistic approach that employs a parameterized neural-net-based autoencoder for matrix completion, thereby enabling simultaneous training between models of the classifier and matrix completion. We find that this approach not only speeds up training time (around a three-fold improvement over a prior approach), but also offers a higher prediction accuracy via a more accurate graph estimate. We demonstrate that our algorithm obtains state-of-the-art performances by respectful margins on benchmark datasets: Achieving the error rates of 0.57% on MNIST with 100 labels; 3.48% on SVHN with 1000 labels; and 6.87% on CIFAR-10 with 4000 labels.

Keywords: Semi-supervised Learning, Matrix Completion, Autoencoders

1 Introduction

While deep neural networks can achieve human-level performance on a widening array of supervised learning problems, they come at a cost: Requiring a large collection of *labeled* data and thus relying on a huge amount of human effort to manually label examples. Semi-supervised learning (SSL) serves as a powerful framework that leverages *unlabeled* data to address the lack of labeled data.

One popular SSL paradigm is to employ *consistency loss* which captures a similarity between prediction results of the same data points yet with different small perturbations. This methodology is inspired by the key smoothness assumption that nearby data points are likely to have the same class, and a variety of algorithms have been developed inspired by this assumption [18, 37, 27, 29]. However, it comes with one limitation: Taking into account only the same

† Equal contribution

single data point (yet with perturbations) while not exploiting any relational structure across distinct data points.

This has naturally motivated another prominent SSL framework, named *graph-based SSL* [24, 43, 36]. The main idea is to employ a *similarity graph*, which represents the relationship among a pair of data points, to form another loss term, called *feature matching loss*, and then incorporate it as a regularization term into a considered optimization. This way, one can expect that similar data points would be embedded tighter in a low-dimensional space, thus yielding a higher classification performance [5] (also see Figure 2 for visualization).

One important question that arises in graph-based SSL is: How to construct such similarity graph? Most prior works rely on features in the input space [2, 43] or given (and/or predicted) labels in the output space [24]. A recent approach is to incorporate a matrix completion approach [36]. The approach exploits both features and available labels to form an augmented matrix having fully-populated features yet with very sparse labels. Leveraging the low-rank structure of such augmented matrix, it resorts to a well-known algorithm (nuclear norm minimization [8]), thereby estimating a similarity graph. However, we face one challenge in this approach. The challenge is that the graph estimation is done *independently* of the model parameter update of an interested classifier, which in turn incurs a significant computational complexity. Notice that the graph update should be done whenever changes are made on classifier model parameters.

Our main contribution lies in developing a novel integrated framework that holistically combines the classifier parameter update with the matrix-completion-based graph update. We introduce a parameterized neural-net-based autoencoder for matrix completion and define a new loss, which we name *autoencoder loss*, to reflect the quality of the graph estimation. We then integrate the autoencoder loss with the original supervised loss (computed only via few available labels) together with two additional losses: (i) consistency loss (guiding the same data points with small perturbations to yield the same class); (ii) feature matching loss (encouraging distinct yet similar data points, reflected in the estimated graph, to have the same class). As an autoencoder, we develop a novel structure inspired by the recent developments tailored for matrix completion [33, 10]; see *Remark* 1 for details.

We emphasize two key aspects of our holistic approach. First, model parameters of the classifier (usually CNN) and the matrix completion block (autoencoder) are *simultaneously* trained, thus exhibiting a significant improvement in computational complexity (around a three-fold gain in various real-data experiments), relative to the prior approach [36] which performs the two procedures *separately in an alternate manner*. Second, our approach exploits the smoothness property of *both* same-yet-perturbed data points and distinct-yet-similar data points. This together with an accurate graph estimate due to our autoencoder-based matrix completion turns out to offer greater error rate performances on various benchmark datasets (MNIST, SVHN and CIFAR-10) with respectful margins over many other state of the arts [24, 36, 38, 32, 22, 18, 37]. The improve-

ments are more significant when available labels are fewer. See Tables 1 and 2 in Section 5 for details.

2 Related work

2.1 Semi-supervised learning

There has been a proliferation of SSL algorithms. One major stream of the algorithms is along the idea of adversarial training [13]. While the literature based on such methodology is vast, we list a few recent advances with deep learning [35, 32, 22, 9]. Springenberg [35] has proposed a categorical GAN (CatGAN) that learns a discriminative classifier so as to maximize mutual information between inputs and predicted labels while being robust to bogus examples produced by an adversarial generative model. Salimans et al. [32] propose a new loss, called feature matching loss, in an effort to stabilize GAN training, also demonstrating the effectiveness of the technique for SSL tasks. Subsequently Li et al. [22] introduce another third player on top of the two players in GANs to exploit label information, thus improving SSL performances. Dai et al. [9] provide a deeper understanding on the role of generator for SSL tasks, proposing another framework. While the prior GAN approaches yield noticeable classification performances, they face one common challenge: Suffering from training instability as well as high computational complexity.

Another prominent stream is not based on the GAN framework, instead employing *consistency loss* that penalizes the distinction of prediction results for the same data points having small perturbations [30, 18, 37, 27, 29]. Depending on how to construct the consistency loss, there are a variety of algorithms including: (i) ladder networks [30]; (ii) TempEns [18]; (iii) Mean Teacher (MT) [37]; (iv) Virtual Adversarial Training (VAT) [27]; (v) Virtual Adversarial Dropout (VAdD) [29]. The recent advances include [44, 38, 4, 42, 1, 3, 46, 34]. In particular, [38, 4, 42, 3, 46, 34] propose an interesting idea of mixing (interpolating) data points and/or predictions, to better exploit the consistency loss, thus achieving promising results.

Most recently, it has been shown in [48] that consistency loss can be better represented with the help of self-supervised learning, which can be interpreted as a feature representation approach via pretext tasks (such as predicting context or image rotation) [17].

2.2 Graph-based SSL

In an effort to capture the *relational* structure across *distinct* data points which are not reflected by the prior approaches, a more advanced technique has been introduced: graph-based SSL. The key idea is to exploit such relational structure via a *similarity graph* that represents the closeness between data points. The similarity graph is desired to be constructed so as to well propagate information w.r.t. few labeled data into the unlabeled counterpart. Numerous algorithms

have been developed depending on how to construct the similarity graph [2, 12, 6, 50, 15, 41, 43, 24, 36, 45, 51, 40]. Recent developments include [51, 40] that exploit graph convolutional networks (GCNs) to optimize the similarity graph and a classifier.

Among them, two most related works are: (i) Smooth Neighbors on Teach Graphs (SNTG) [24]; (ii) GSCNN [36]. While the SNTG [24] designs the graph based solely on *labels* (given and/or predicted) in the output space, the GSCNN [36] incorporates also features in the input space to better estimate the similarity graph with the aid of the *matrix completion* idea. However, the employed matrix completion takes a conventional approach based on nuclear norm minimization [8], which operates independently of a classifier's parameter update and therefore requires two separate updates. This is where our contribution lies in. We employ a neural-net-based matrix completion so that it can be trained simultaneously with the interested classifier, thereby addressing the computational complexity issue.

2.3 Matrix completion

Most traditional algorithms are based on rank minimization. Although the rank minimization is NP-hard, Candes and Recht [8] proved that for some enough number of observed matrix entries, one can perfectly recover an unknown matrix via nuclear norm minimization (NNM). This NNM algorithm formed the basis of the GSCNN [36]. Instead we employ a deep learning based approach for matrix completion. Our approach employs a *parameterized* neural-net-based autoencoder [33, 23, 10] and therefore the model update can be gracefully merged with that of the classifier. The parameterization that leads to simultaneous training is the key to speeding up training time.

3 Problem formulation

Let $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{m}$ and $\mathcal{U} = \{x_i\}_{i=m+1}^{n}$ be labeled and unlabeled datasets respectively, wherein $x_i \in \mathcal{X}$ indicates the ith data (observation) and $y_i \in \mathcal{Y} = \{1, 2, \ldots, c\}$ denotes the corresponding label. Here c is the number of classes. Usually available labels are limited, i.e., $m \ll n$. The task of SSL is to design a classifier f (parameterized with θ) so that it can well predict labels for *unseen* examples. With regard to a loss function, we employ one conventional approach that takes into account the two major terms: (i) supervised loss (quantifying prediction accuracy w.r.t. labeled data); (ii) regularization terms (reflecting the data structure exposed in unlabeled data).

Supervised loss. We consider:

$$L_s(\theta) = \sum_{i=1}^{m} \ell_s(f(x_i; \theta), y_i) \tag{1}$$

where $\ell_s(\cdot, \cdot)$ denotes cross entropy loss and $f(x_i; \theta)$ is the classifier softmax output that aims to represent the ground-truth conditional distribution $p(y_i|x_i; \theta)$.

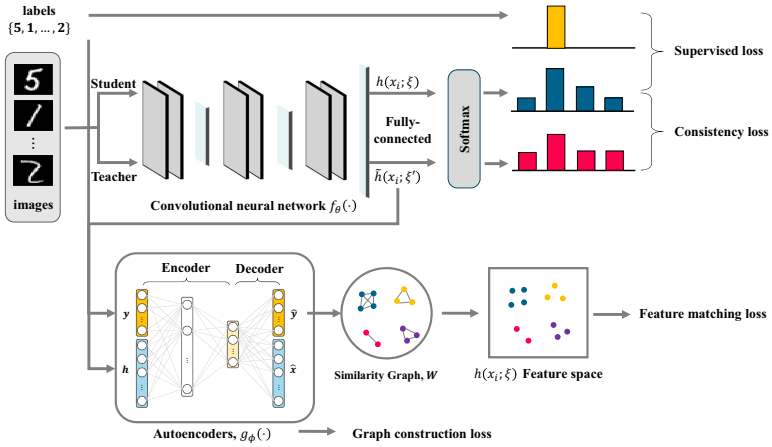

Fig. 1: A graph-based SSL architecture with a CNN classifier (parameterized with θ) and an autoencoder for matrix completion (parameterized with ϕ).

As regularization terms, we employ two unsupervised losses: (i) consistency loss (that captures the distinction of prediction results of the same data points with different yet small perturbations); (ii) feature matching loss (that quantifies the difference between distinct data points via a similarity graph together with some features). Below are detailed formulas that we would take.

Consistency loss. We use one conventional formula as in [18, 37, 27, 29, 24]:

$$L_c(\theta) = \sum_{i=1}^{n} \ell_c(f(x_i;\theta,\xi), \tilde{f}(x_i;\theta',\xi')) \tag{2}$$

where $f(x_i;\theta,\xi)$ denotes the prediction of one model, say student model, with parameter θ and random perturbation ξ (imposed to the input x_i); $\tilde{f}(x_i;\theta',\xi')$ indicates that of another model, say teacher model, with parameter θ' and different perturbation ξ' yet w.r.t. the same input x_i; and $\ell_c(\cdot,\cdot)$ is a distance measure between the outputs of the two models (for instance, the Euclidean distance or KL divergence). Here we exploit both labeled and unlabeled datasets. Notice that this loss penalizes the difference between the predictions of the same data point x_i yet with different perturbations, reflected in ξ and ξ'. We consider a simple setting where $\theta' = \theta$, although these can be different in general [18, 37, 27, 29].

Feature matching loss. Another unsupervised loss that we will use to exploit the relational structure of different data points is feature matching loss:

$$L_g(\theta, \mathbf{W}) = \sum_{x_i, x_j \in \mathcal{L}\cup\mathcal{U}} \ell_g(h(x_i;\theta), h(x_j;\theta), W_{ij}) \tag{3}$$

where $h : \mathcal{X} \to \mathbb{R}^p$ is a mapping from the input space to a low dimensional feature space; W_{ij} denotes the (i, j) entry of a similarity graph matrix $\mathbf{W}$ (taking 1 if x_i and x_j are of the same class, 0 otherwise); and $\ell_g(\cdot, \cdot)$ is another distance measure between the ith and jth features which varies depending on W_{ij}. Here the function ℓ_g is subject to our design choice [24, 36], and we use the contrastive Siamese networks [7] as in [24]:

$$\ell_g = \begin{cases} \|h(x_i) - h(x_j)\|^2 & \text{if } W_{ij} = 1 \\ \max(0, \mathsf{margin} - \|h(x_i) - h(x_j)\|)^2 & \text{if } W_{ij} = 0 \end{cases} \tag{4}$$

where margin denotes a pre-defined positive value which serves as a threshold in feature difference for declaring distinct classes, and $\|\cdot\|^2$ is the Euclidean distance.

Various methods have been developed for construction of $\mathbf{W}$. One recent work is the SNTG [24] which uses the predictions of the teacher model for the construction. Another recent work [36] takes a *matrix completion* approach which additionally exploits features on top of given labels. Here the predicted labels due to matrix completion serve to construct the graph.

SSL Framework. Here is the unified loss function of our consideration:

$$L_s(\theta) + \lambda_c L_c(\theta) + \lambda_g L_g(\theta, \mathbf{W}) \tag{5}$$

where λ_c and λ_g are regularization factors that serve to balance across three different loss terms.

In (5), we face one challenge when employing a recent advance [36] that relies on conventional matrix completion for $\mathbf{W}$. The GSCNN [36] solves nuclear norm minimization [8] for matrix completion via a soft impute algorithm [26]. Here an issue arises: The algorithm [26] has to *retrain the graph* $\mathbf{W}$ *for every iteration whenever* features of the classifier, affected by θ, are updated. In other words, matrix completion is done in an *alternate* manner with the classifier update. For each iteration, say t, the graph is estimate to output, say $\mathbf{W}^{(t)}$, from the current classifier parameter, say $\theta^{(t)}$, and this updated $\mathbf{W}^{(t)}$ yields the next parameter estimate $\theta^{(t+1)}$ as per (5), and this process is repeated. Here the challenge is that this alternating update incurs significant computational complexity.

4 Our approach

To address such challenge w.r.t. computational complexity, we invoke a parameterization trick. The idea is to parameterize a matrix completion block with a neural network, and to *simultaneously* train both parameters, one for classifier and the other for matrix completion. Specifically we employ an *autoencoder*-type neural network with parameter, say ϕ, for matrix completion; introduce another loss that we call *autoencoder loss*, which captures the quality of matrix completion and therefore the graph estimate; integrate the new loss with the other three loss terms in (5); and then simultaneously update both θ (classifier) and ϕ (autoencoder) guided by the integrated loss. We will detail the idea in the next sections.

4.1 Learning the graph with autoencoder

For a classifier, we employ a CNN model. Let $h(x_1), \ldots, h(x_n) \in \mathbb{R}^d$ be the feature vectors w.r.t. n examples prior to the softmax layer. We first construct a feature matrix that stacks all of the vectors: $\mathbf{X} = [h(x_1), \ldots, h(x_n)] \in \mathbb{R}^{d \times n}$. Let $u(y_1), \ldots, u(y_m) \in \mathbb{R}^c$ be the one-hot-coded label vectors. We then construct a label matrix stacking all of them: $\mathbf{Y} = [u(y_1), \ldots, u(y_m), \underbrace{\mathbf{0}_c, \ldots, \mathbf{0}_c}_{(n-m)\ vectors}] \in \mathbb{R}^{c \times n}$ where $\mathbf{0}_c$ is the all-zero vector of size c. Notice that $\mathbf{X}$ is fully populated while $\mathbf{Y}$ is very sparse due to many missing labels. Let $\Omega_{\mathbf{Y}}$ be the set of indices for the observed entries in $\mathbf{Y}$: $(i, j) \in \Omega_{\mathbf{Y}}$ when y_j is an observed label. Now we construct an augmented matrix that stacks $\mathbf{X}$ and $\mathbf{Y}$ in the row wise:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Y} \\ \mathbf{X} \end{bmatrix}. \tag{6}$$

This is an interested matrix for completion. Notice that completing $\mathbf{Z}$ enables predicting all the missing labels, thus leading to a graph estimate.

For matrix completion, we employ an autoencoder-type neural network which outputs a completed matrix fed by $\mathbf{Z}$. We also adopt the idea of input dropout [39]: Probabilistically replacing some of the label-presence column vectors with zero vector $\mathbf{0}_c$. For instance, when the jth column vector $\mathbf{z}_j = \begin{bmatrix} u(y_j) \\ h(x_j) \end{bmatrix}$ is dropped out, the output reads: $\bar{\mathbf{z}}_j = \begin{bmatrix} \mathbf{0}_c \\ h(x_j) \end{bmatrix}$. With the input dropout, we get:

$$\bar{\mathbf{Z}} = \begin{bmatrix} \bar{\mathbf{Y}} \\ \mathbf{X} \end{bmatrix}. \tag{7}$$

For training our autoencoder function g parameterized by ϕ, we employ the following loss (autoencoder loss) that reflects the reconstruction quality of the interested matrix:

$$L_{AE}(\theta, \phi) = \sum_{(i,j) \in \Omega_{\mathbf{Y}}} (\mathbf{Y}_{ij} - \hat{\mathbf{Y}}_{ij})^2 + \mu(t) \|\mathbf{X} - \hat{\mathbf{X}}\|_F^2 \tag{8}$$

where $(\hat{\mathbf{Y}}, \hat{\mathbf{X}})$ indicates the output of the ϕ-parameterized autoencoder $g_\phi(\bar{\mathbf{Z}})$; $\|\cdot\|_F$ denotes the Frobenius norm; and $\mu(t)$ is a hyperparameter that balances the reconstruction loss of observed labels (reflected in the first term) against that of the features (reflected in the second term). Since the quality of the feature matrix estimation improves with the learning progress (relative to the fixed label counterpart), we use a monotonically increasing ramp-up function $\mu(t)$ [18].

Let $\tilde{u}(y_i)$ be the predicted label vector for the ith example, obtained by the autoencoder $g_\phi(\bar{\mathbf{Z}})$. Let $\tilde{y}_i = \text{argmax}_k \, [\tilde{u}(y_i)]_k$ where $[\cdot]_k$ is the kth component of the vector (the estimated probability that the ith example belongs to class k). This then enables us to construct a similarity graph:

$$W_{ij} = \begin{cases} 1 & \text{if } \tilde{y}_i = \tilde{y}_j; \\ 0 & \text{if } \tilde{y}_i \neq \tilde{y}_j. \end{cases} \tag{9}$$

We consider a simple binary case where $W_{ij} \in \{0, 1\}$ as in [24]. One may use a weighted graph with soft-decision values. The constructed graph is then applied to the feature matching loss (3) to train the classifier model (CNN).

Remark 1. (Our choice for autoencoder structure): The autoencoder that we design is of a special structure: A nonlinear (multi-layered) encoder followed by a linear (single-layer) decoder. We name this the *basis learning autoencoder* [21, 20]. Note that the structure helps learning the basis via a linear decoder, as it is guided to linearly combine the vectors generated by a non-linear encoder. The rationale behind this choice is that any matrix can be represented as a linear combination of the basis vectors of its row or column space, and *the basis vectors serve as the most efficient features of a matrix*. The number of nodes in the last layer of the encoder can serve as an effective *rank* of the matrix (the number of basis vectors extracted from the encoder). We observe through experiments that matrix completion performs well when the effective rank is greater than or equal to the number of classes. Also we have empirically verified that the bases of the data are indeed learned by the autoencoder which has a linear decoder. We leave the detailed empirical analysis in the supplementary. ■

4.2 Simultaneous training

We aim to update autoencoder parameter ϕ *simultaneously* with classifier parameter θ, unlike the prior approach [36] that takes an *alternating* update. To this end, we integrate the autoencoder loss (8) with the prior three loss terms (5):

$$L_s(\theta) + w(t) \cdot (L_c(\theta) + \lambda \cdot L_g(\theta, \mathbf{W})) + L_{AE}(\theta, \phi) \tag{10}$$

where $L_s(\theta)$, $L_c(\theta)$, $L_g(\theta, \mathbf{W})$ and $L_{AE}(\theta, \phi)$ indicate supervised loss (1), consistency loss (2), feature matching loss (3), and autoencoder loss (8), respectively. Here we employ three hyperparameters: λ controls the ratio between $L_c(\theta)$ and $L_g(\theta, \mathbf{W})$; $w(t)$ is a ramp-up function that increases with epoch; and $\mu(t)$ is another ramp-up function placed inside $L_{AE}(\theta, \phi)$ (see (8)). We use the same ramp-up function as in [18, 24], since features are not reliable in the initial phase of the training. For simplicity, we apply the same weight between $L_{AE}(\theta, \phi)$ and $L_s(\theta)$.

Our proposed algorithm is summarized in Algorithm 1. In Step 4, labeled and unlabeled data are randomly sampled while maintaining the constant ratio between them for every mini-batch. Steps 6-8 indicate the feed-forward process of student model, teacher model, and autoencoder, respectively. Step 10 constructs a similarity graph by comparing pairs of the labels predicted by autoencoder. We then update all parameters (θ, ϕ) simultaneously as per the single integrated loss function that we design in (10). This process is repeated until the maximum epoch is reached.

Tables 1 and 2 show performance comparisons with several other SSLs. Among consistency-loss-based approaches, we consider Π-model [18], Mean-Teacher [37] and ICT [38]. The performance results reported for baselines come from corresponding papers.

Algorithm 1 Autoencoder-based Graph Construction

Input: Labeled dataset $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{m}$ and unlabeled dataset $\mathcal{U} = \{x_i\}_{i=m+1}^{n}$, hyperparameters $\lambda, \mu(t)$, $w(t)$ and the number of training epochs T
Initialize: $f_\theta(x)$ (student model), $\tilde{f}_\theta(x)$ (teacher model), and $g_\phi(x)$ (ϕ-parameterized autoencoder)
for $t = 1$ to T **do**
  Randomly sample mini-batches from $\mathcal{L}$ and $\mathcal{U}$
  for each mini-batch $\mathcal{B}$ **do**
    $(z_i, h_i) \leftarrow f_\theta(x_{i\in\mathcal{B}})$ obtain outputs & features
    $(\tilde{z}_i, \tilde{h}_i) \leftarrow \tilde{f}_\theta(x_{i\in\mathcal{B}})$ (for teacher model)
    $\hat{y} \leftarrow g_\phi([\tilde{h}_i, y_{i\in\mathcal{B}}])$ predict labels via AE
    for $(\hat{y}_i, \hat{y}_j)$ **do**
      Compute W_{ij} from $\hat{y}$ as per (9)
    end for
    update θ and ϕ by minimizing (10)
  end for
end for

5 Experiments

We conduct real-data experiments on three benchmark datasets: MNIST, SVHN, and CIFAR-10. In all of the datasets, only a small fraction of the training data is used as labeled data, while the remaining being considered as unlabeled data. The MNIST consists of 70,000 images of size 28×28 with 60,000 for training and 10,000 for testing. The SVHN (an image of size 32×32 representing a close-up house number) consists of 73,257 color images for training, and 26,032 for testing. The CIFAR-10 contains 50,000 color images (each of size 32×32) for training and 10,000 for testing.

We adopt the same preprocessing technique as in [24, 37, 18, 38]: Whitening with zero mean and unit variance for MNIST and SVHN; and Zero-phase Component Analysis (ZCA) whitening for CIFAR-10. We apply the translation augmentation to SVHN, and the mirror and translation augmentation to CIFAR-10. For a classifier model, we employ CNN-13, the standard benchmark architecture used in the prior works [18, 37, 27, 29, 24, 38]. We leave the detailed network architecture in the supplementary. We mostly follow hyperparameter search from [37, 24, 38]. See the supplementary for details. As for the choice of labeled data, we follow the common practice in [30, 32, 24]: Randomly sampling 100, 1000, 4000 labels for MNIST, SVHN and CIFAR-10, respectively. We also include experiments in which fewer labels are available. The results (to be reported) are the ones averaged over 10 trials with different random seeds for data splitting.

Very recently, graph-based SSL [51] has been proposed that employs a GCN-based classifier. Since the framework enforces the input of graph learning layer to be always fixed (such as pixel values or features from a CNN descriptor), the performance is limited by the quality of the input feature. Hence, no direct comparison has been made in our paper.

Models	MNIST 100 labels	MNIST 50 labels	MNIST 20 labels
ImprovedGAN [32]	0.98 ± 0.065	2.21 ± 1.36	16.77 ± 4.52
TripleGAN [22]	0.91 ± 0.58	1.56 ± 0.72	4.81 ± 4.95
Π-model [18]	0.89 ± 0.15	1.02 ± 0.37	6.32 ± 6.90
TempEns[18]	$1.55 \pm 0.49^*$	$3.33 \pm 1.42^*$	$17.52 \pm 6.56^*$
MT [37]	$1.00 \pm 0.54^*$	$1.36 \pm 0.59^*$	$6.08 \pm 5.19^*$
VAT [27]	$0.88 \pm 0.38^*$	$1.34 \pm 0.60^*$	$5.15 \pm 4.77^*$
SNTG [24]	$\underline{0.66 \pm 0.07}$	$\underline{0.94 \pm 0.42}$	$\underline{1.36 \pm 0.78}$
ICT [38]	$0.95 \pm 0.29^*$	1.29 ± 0.34	3.83 ± 2.67
Our model	$\mathbf{0.57 \pm 0.06}$ (13.6%)	$\mathbf{0.64 \pm 0.14}$ (31.9%)	$\mathbf{0.85 \pm 0.21}$ (37.6%)

Table 1: Error rates (%) on MNIST, averaged over 10 trials. The boldface, underline and the number in parenthesis indicate the best results, the best among baselines, and performance gain over the best baseline, respectively. The results that we reproduced are marked as *.

Models	SVHN 1000 labels	SVHN 500 labels	SVHN 250 labels
Supervised-only	12.83 ± 0.47	22.93 ± 0.67	40.62 ± 0.95
Π-model [18]	4.82 ± 0.17	6.65 ± 0.53	9.93 ± 1.15
TempEns[18]	4.42 ± 0.16	5.12 ± 0.13	12.62 ± 2.91
MT [37]	3.95 ± 0.19	4.18 ± 0.27	4.35 ± 0.50
VAT [27]	$3.94 \pm 0.12^*$	$4.71 \pm 0.29^*$	$5.49 \pm 0.34^*$
SNTG [24]	$\underline{3.82 \pm 0.25}$	$\underline{3.99 \pm 0.24}$	$\underline{4.29 \pm 0.23}$
ICT [38]	3.89 ± 0.04	4.23 ± 0.15	4.78 ± 0.68
Our model	$\mathbf{3.48 \pm 0.13}$ (8.90%)	$\mathbf{3.64 \pm 0.15}$ (9.02%)	$\mathbf{3.97 \pm 0.20}$ (7.46%)

Table 2: Error rates (%) on SVHN, averaged over 10 trials. The boldface indicates the best results and the underline indicates the best among baselines. The number in parenthesis is the performance gain over the best baseline. The results that we reproduced are marked as *.

5.1 Performance evaluations

Tables 1, 2, and 3 demonstrate the error rates respectively on: MNIST with 100/50/20 labels; SVHN with 1000/500/250 labels; and CIFAR-10 with 4000/2000/1000 labels. The underlines indicate the state of the arts among baselines, e.g., SNTG [24] in MNIST and SVHN; and ICT [38] in CIFAR-10. The number in parenthesis denotes the relative performance gain over the best baseline. We see noticeable performance improvements especially on MNIST dataset with 100 labels. It could seems that the absolute performance gain is not dramatic. In [24], however, this minor-looking absolute gain is considered to be significant, and it is well reflected in relative performance gain, which is 13.6% over the best baseline. Observe more significant improvements with a decrease in the number of available labels. We expect this is because our well-constructed graph plays a more crucial role in challenging scenarios with fewer labels.

Models	CIFAR-10		
	4000 labels	2000 labels	1000 labels
Supervised-only	20.26 ± 0.38	31.16 ± 0.66	39.95 ± 0.75
Π-model [18]	12.36 ± 0.31	31.65 ± 1.20	17.57 ± 0.44
TempEns[18]	12.16 ± 0.24	15.64 ± 0.39	23.31 ± 1.01
MT [37]	12.31 ± 0.28	15.73 ± 0.31	21.55 ± 1.48
VAT [27]	$11.23 \pm 0.21^*$	$14.07 \pm 0.38^*$	$19.21 \pm 0.76^*$
SNTG [24]	9.89 ± 0.34	13.64 ± 0.32	18.41 ± 0.52
GSCNN [36]	15.49 ± 0.64	18.98 ± 0.62	16.82 ± 0.47
ICT [38]	$\underline{7.29 \pm 0.02}$	$\underline{9.26 \pm 0.09}$	$\underline{15.48 \pm 0.78}$
Our model	$\mathbf{6.87 \pm 0.19}$ (5.76%)	$\mathbf{8.13 \pm 0.22}$ (12.20%)	$\mathbf{12.50 \pm 0.58}$ (19.25%)

Table 3: Error rates (%) on CIFAR-10, averaged over 10 trials. The boldface indicates the best results and the underline indicates the best among baselines. The number in parenthesis is the performance gain over the best baseline. The results that we reproduced are marked as *.

5.2 Comparison to graph-based SSL [18, 24, 36]

We put a particular emphasis on performance comparisons with the most related graph-based approaches [18, 24, 36], both in view of error rate and computational complexity. The training time is measured w.r.t. MNIST with 100 labels and without data augmentation. Our implementation is done via TensorFlow on Xeon E5-2650 v4 CPU and TITAN V GPU.

Table 4 shows both error rate and training (running) time. Notice that our matrix completion yields a higher accuracy than SNTG [24] which only exploits the softmax values in the output space. This implies that using *both* features and predictions is more effective in better estimating a similarity graph, although it requires slightly increased model parameters and therefore slows down running time yet by a small margin.

In comparison to another matrix completion approach (GSCNN [36][1]), our framework offers much faster training time (around **3.1** times faster) in addition

[1] For GSCNN, we use the same CNN structure as in this paper, and incorporate a consistency loss for a fair comparison.

Models	Error rate (%)	Running Time (s)	# of parameters
Π-model [18]	0.89 ± 0.15	18,343.81	3,119,508
SNTG [24]	0.66 ± 0.07	19,009,74	3,119,508
GSCNN [36][1]	0.60 ± 0.13	62,240.97	3,119,508
Our model	0.57 ± 0.06	20,108.16	3,255,898,

Table 4: Comparison to the other graph-based SSLs on MNIST with 100 labels without augmentation.

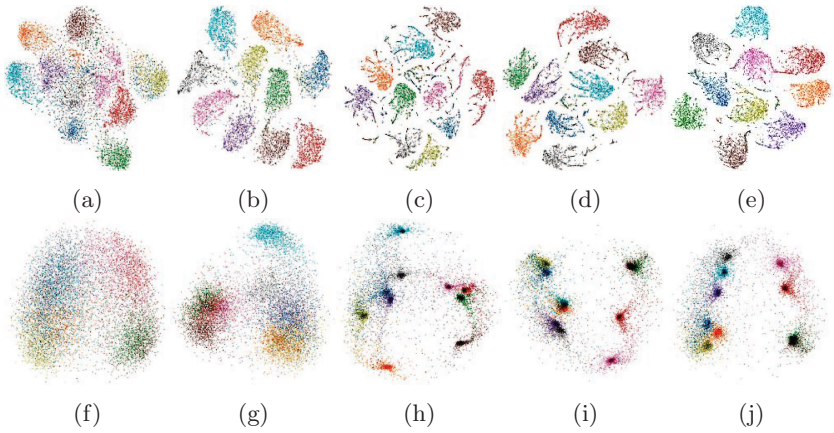

Fig. 2: Embeddings of CIFAR-10 test data with 10,000 images. These features are projected onto a two-dimensional space using t-SNE [25] (a–e) and PCA (f–j). (a) Supervised-only. (b) Supervised + Consistency. (c) Supervised + Feature matching. (d) Supervised + Consistency + Feature matching. (e) Supervised + Consistency + Feature matching with our Autoencoder.

to the performance improvement in error rate. This is because our integrated approach enables *simultaneous training* between the models of the classifier and matrix completion. While our algorithm introduces a *parameterized autoencoder* and thus requires more model parameters, this addition is negligible relative to the complexity of the CNN classifier model employed.

Under graph-based approaches, even a slightly inaccurate graph may yield non-negligible performance degradation, as the error can be propagated through iterations. An inaccurate graph is likely to appear in the initial phase of training. To overcome this, we adjusted the reliability of the estimated graph using $w(t)$ monotonically increasing function in feature matching loss as in [24]. [24] already showed stable convergence when obtaining a graph from the teacher model classifier. Empirically, the graph accuracy obtained from our autoencoder is always higher than that of [24]. We leave the empirical result in the supplementary. As a result, we did not observe any divergence in all our experiments.

5.3 Ablation study

In an effort to quantify the impact of every individual component employed in our framework, we also conduct an ablation study. This is done for CIFAR-10 with 4000 labels. We use supervised loss by default. We sequentially incorporate consistency loss, feature matching loss and graph construction via our autoencoder, as illustrated in Table 5. We also consider three representative methods (Π-

	Error rate (%)
Supervised loss	20.26 ± 0.38
Supervised + Feature matching	13.99 ± 0.20
Supervised + Feature matching with AE	13.37 ± 0.17 (+4.43%)
***Π*-model**	
Supervised + Consistency	12.36 ± 0.31
Supervised + Consistency + Feature matching	11.00 ± 0.13
Supervised + Consistency + Feature matching with AE	10.81 ± 0.13 (+1.72%)
MT	
Supervised + Consistency	12.31 ± 0.28
Supervised + Consistency + Feature matching	12.12 ± 0.14
Supervised + Consistency + Feature matching with AE	11.54 ± 0.34 (+3.35%)
ICT	
Supervised + Consistency	7.25 ± 0.20[2]
Supervised + Consistency + Feature matching	7.18 ± 0.13
Supervised + Consistency + Feature matching with AE	6.87 ± 0.19 (+3.94%)

Table 5: Ablation study on CIFAR-10 with 4000 labels. The number in parenthesis is the performance gain that we can obtain via the similarity graph due to our autoencoder.

model [18], Mean-Teacher [37], and ICT [38]) to see the effects of our method under various consistency losses.

Table 5 shows the results on CIFAR-10 with 4000 labels. Note that ICT [38] offers the most powerful consistency loss relative to *Π*-model [18] and MT [37]. Feature matching loss indeed plays a role and the effect is more significant with our autoencoder approach. This suggests that the precise graph due to our autoencoder maximizes the benefit of feature matching.

Moreover, we visualize 2D embeddings on CIFAR-10 test data using Embedding projector in Tensorflow. The feature vectors $h(x_i) \in \mathbb{R}^{128}$, extracted from the CNN, are projected onto a two-dimensional space using t-SNE [25]. We set perplexity to 25 and the learning rate to 10 as hyperparameters for t-SNE. In Figure 2, we see the impact of each loss term upon clustering structures in an embedding domain. Here ICT is employed for consistency loss. From Figure 2(c), we can see a clearer clustering structure (relative to (b)), suggesting that feature matching loss helps embedded data to move away from the decision boundary. Taking both losses (Figures (d) and (e)), we observe tighter clusters particularly with our autoencoder approach.

5.4 Wide ResNet results

Recent works [4, 28] employ a 28-layer Wide ResNet [47] architecture, instead of the standard CNN-13 that we have used for the above experiments. In an effort

[2] In Table 5, the original paper reported 7.29 ± 0.02 as the average and standard deviation for 3 runs. We replaced them with those with 10 runs.

Models	CIFAR-10				
	250 labels	500 labels	1000 labels	2000 labels	4000 labels
Π-model [18]	53.02 ± 2.05	41.82 ± 1.52	31.53 ± 0.98	23.07 ± 0.66	17.41 ± 0.37
PseudoLabel [19]	49.98 ± 1.17	40.55 ± 1.70	30.91 ± 1.73	21.96 ± 0.42	16.21 ± 0.11
Mixup [49]	47.43 ± 0.92	36.17 ± 1.36	25.72 ± 0.66	18.14 ± 1.06	13.15 ± 0.20
VAT [27]	36.03 ± 2.82	26.11 ± 1.52	18.68 ± 0.40	14.40 ± 0.15	11.05 ± 0.31
MT [37]	47.32 ± 4.71	42.01 ± 5.86	17.32 ± 4.00	12.17 ± 0.22	10.36 ± 0.25
MixMatch [4]	11.08 ± 0.87	9.65 ± 0.94	7.75 ± 0.32	7.03 ± 0.15	6.24 ± 0.06
Our model	10.87 ± 0.40	9.53 ± 0.85	7.69 ± 0.17	7.08 ± 0.14	6.24 ± 0.06

Table 6: Error rate (%) with a 28-layer Wide ResNet on CIFAR-10, averaged over 5 trials.

to investigate whether our framework can be gracefully merged with another, we also consider this architecture to conduct more experiments. We followed the simulation settings given in [4]. Table 6 demonstrates the error rates on CIFAR-10 with 4000/2000/1000/500/250 labels. The results show that our model is well merged with the Wide ResNet architecture, while offering slightly better performance relative to the state-of-the-art algorithm [4]. We do expect other techniques [3, 34, 46] can be gracefully merged with our approach (that additionally employs feature matching loss), yielding further improvements, as also demonstrated in Table 6 w.r.t. other technique [4].

6 Conclusion

We proposed a holistic training approach to graph-based SSL that is computationally efficient and offers the state-of-the-art error rate performances on benchmark datasets. The key idea is to employ an autoencoder-based matrix completion for similarity graph which enables simultaneous training between model parameters of an interested classifier and matrix completion. Experiments on three benchmark datasets demonstrate that our model outperforms the previous state of the arts. In particular, the performance gain is more distinct with a decrease in the number of available labels. Ablation study emphasizes the role of our key distinctive component: Autoencoder for graph construction. Future works of interest include: (a) a graceful merge with the state-of-the-art consistency loss-based approach [48, 3, 34, 46]; and (b) evaluations of our model on various large-scale datasets such as CIFAR-100 and ImageNet.

Acknowledgments. This work was supported by the ICT R&D program of MSIP/IITP (2016-0-00563, Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion), and Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (2020-0-00626, Ensuring high AI learning performance with only a small amount of training data).

References

1. Athiwaratkun, B., Finzi, M., Izmailov, P., Wilson, A.G.: There are many consistent explanations of unlabeled data: Why you should average. In: Proceedings of the International Conference on Learning Representation (ICLR) (2019)
2. Belkin, M., Niyogi, P., Sindwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. Journal of machine learning research **7**, 2399–2434 (Nov 2006)
3. Berthelot, D., Carlini, N., Cubuk, E.D., Kurakin, A., Sohn, K., Zhang, H., Raffel, C.: Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. In: Proceedings of the International Conference on Representation Learning (ICLR) (2020)
4. Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., Raffel, C.: Mixmatch: A holistic approach to semi-supervised learning. In: Advances in Neural Information Processing Systems (NIPS) (Dec 2019)
5. Berton, L., Andrade Lopes, A.d.: Graph construction for semi-supervised learning. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence. pp. 4343–4344 (2015)
6. Blum, A., Chawla, S.: Learning from labeled and unlabeled data using graph mincuts. In: Proceedings of the International Conference on Machine Learning. pp. 19–26 (June 2001)
7. Bromley, J., Guyon, I., LeCun, Y., Sickinger, E., Shah, R.: Signature verification using a "siamese" time delay neural network. In: Advances in Neural Information Processing Systems. pp. 737–744 (1994)
8. Candés, E.J., Recht, B.: Exact matrix completion via convex optimization. Foundations of Computational mathematics **9(6)**, 717–772 (2009)
9. Dai, X., Yang, Z., Yang, F., Cohen, W.W., Salakhutdinov, R.: Good semi-supervised learning that requires a bad gan. In: Advances in Neural Information Processing Systems (NIPS). pp. 6510–6520 (Dec 2017)
10. Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., Zhang, F.: A hybrid collaborative filtering model with deep structure for recommender systems. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence. pp. 1309–1315 (2017)
11. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: International Conference on Artificial Intelligence and Statistics (2010)
12. Gong, C., Liu, T., Tao, D., Fu, K., Tu, E., Yang, J.: Deformed graph laplacian for semisupervised learning. IEEE Transactions on Neural Networks and Learning Systems **26(10)**, 717–772 (2015)
13. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio§, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems (NIPS) (Dec 2014)
14. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2015)
15. Iscen, A., Tolias, G., Avrithis, Y., Chum, O.: Label propagation for deep semi-supervised learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5070–5079 (June 2019)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference for Learning Representations (2015)

17. Kolesnikov, A., Zhai, X., Beyer, L.: Revisiting self-supervised visual representation learning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
18. Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. In: Proceedings of the International Conference on Representation Learning (ICLR) (Apr 2017)
19. 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. vol. 3, p. 2 (2013)
20. Lee, K., Jo, H., Kim, H., Lee, Y.H.: Basis learning autoencoders for hybrid collaborative filtering in cold start setting. In: Proceedings of the 29th International Workshop on Machine Learning for Signal Processing (MLSP) (2019)
21. Lee, K., Lee, Y.H., Suh, C.: Alternaitng autoencoders for matrix completion. In: Proceedings of the IEEE Data Science Workshop (DSW) (2018)
22. Li, C., Xu, K., Zhu, J., Zhang, B.: Triple generative adversarial nets. In: Advances in Neural Information Processing Systems (NIPS). pp. 4088–4098 (Dec 2017)
23. Li, S., Kawale, J., Fu, Y.: Deep collaborative filtering via marginalized denoising auto-encoder. In: Proceedings of the 24th ACM International Conference on Information and Knowledge Management. pp. 811–820. ACM (2015)
24. Luo, Y., Zhu, J., Li, M., Ren, Y., Zhang, B.: Smooth neighbors on teacher graphs for semi-supervised learning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8896–8905 (Jun 2018)
25. Maaten, L., Hinton, G., Johnson, I.: Visualizing data using t-sne. Journal of Machine Learning Research **9**, 2579–2605 (Nov 2008)
26. Mazumder, R., Hastie, T., , Tibshirani, R.: Spectral regularization algorithms for learning large incomplete matrices. Journal of Machine Learning Research **11**, 2287–2322 (2010)
27. Miyato, T., ichi Maeda, S., Koyama, M., Ishii, S.: Virtual adversarial training: A regularization method for supervised and semi-supervised learning. In: Proceedings of the International Conference on Representation Learning (ICLR) (Apr 2017)
28. Oliver, A., Odena, A., Raffel, C., Cubuk, E.D., Goodfellow, I.J.: Realistic evaluation of semi-supervised learning algorithms. In: Advances in Neural Information Processing Systems (NIPS). pp. 3235–3246 (Dec 2018)
29. Park, S., Park, J., Shin, S.J., Moon, I.C.: Adversarial dropout for supervised and semi-supervised learning. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (Apr 2018)
30. Rasmus, A., Valpola, H., Honkala, M., Berglund, M., Raiko, T.: Semi-supervised learning with ladder networks. In: Advances in Neural Information Processing Systems (NIPS). pp. 3546–3554 (Dec 2015)
31. Salimans, T., Kingma, D.: Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In: Advances in Neural Information Processing Systems (NIPS) (2016)
32. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., radford, A., Chen, X.: Improved techniques for training gans. In: Advances in Neural Information Processing Systems (NIPS). pp. 2234–2242 (Dec 2016)
33. Sedhain, S., Krishna, M., Scanner, S., Xie, L.: Autorec: Autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web. pp. 111–112 (2015)
34. Sohn, K., Berthelot, D., Li, C.L., Zhang, Z., Carlini, N., Cubuk, E.D., Kurakin, A., Zhang, H., Raffel, C.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In: arXiv preprint arXiv:2001.07685 (2020)

35. Springenberg, J.T.: Unsupervised and semi-supervised learning with categorical generative adversarial networks. In: Proceedings of the International Conference on Learning Representation (ICLR) (May 2016)
36. Taherkhani, F., Kazemi, H., Nasrabadi, N.M.: Matrix completion for graph-based deep semi-supervised learning. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence. pp. 8896–8905 (Jan 2019)
37. Tarvainen, A., 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 (NIPS). pp. 1195–1204 (Dec 2017)
38. Verma, V., Lamb, A., Kannala, J., Bengio, Y., Lopez-Paz, D.: Interpolation Consistency Training for Semi-Supervised Learning. In: International Joint Conference on Artifical Intelligence. pp. 3635–3641 (Aug 2019)
39. Volkovs, M., Yu, G., Poutanen, T.: Dropoutnet: Addressing cold start in recommender systems. In: Advances in Neural Information Processing Systems (NIPS) (Dec 2015)
40. Wan, S., Gong, C., Zhong, P., Du, B., Zhang, L., Yang, J.: Multi-scale dynamic graph convolutional network for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing pp. 1–16 (2019)
41. Wang, B., Tu, Z., Tsotsos, J.K.: Dynamic label propagation for semi-supervised multi-class multi-label classification. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 425–432 (2013)
42. Wang, Q., Li, W., Gool, L.V.: Semi-supervised learning by augmented distribution alignment. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1466–1475 (2019)
43. Weston, J., Ratle, F., Collobert, R.: Deep learning via semi-supervised embedding. In: Proceedings of the 25th international conference on Machine learning. pp. 1168–1175 (July 2008)
44. Wu, S., Li, J., Liu, C., Yu, Z., Wong, H.S.: Mutual learning of complementary networks via residual correction for improving semi-supervised classification. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6500–6509 (2019)
45. Wu, X., Zhao, L., Akoglu, L.: A quest for structure: jointly learning the graph structure and semi-supervised classification. In: In Proceedings of the 27th ACM International Conference on Information and Knowledge Management (2018)
46. Xie, Q., Dai, Z., Hovy, E., Luong, M.T., Le, Q.V.: Unsupervised data augmentation for consistency training. In: arXiv preprint arXiv:1904.12848 (2019)
47. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: Richard C. Wilson, E.R.H., Smith, W.A.P. (eds.) Proceedings of the British Machine Vision Conference (BMVC). pp. 87.1–87.12. BMVA Press (September 2016). https://doi.org/10.5244/C.30.87, https://dx.doi.org/10.5244/C.30.87
48. Zhai, X., Oliver, A., Kolesnikov, A., Beyer, L.: S4l: Self-supervised semi-supervised learning. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (Nov 2019)
49. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: Proceedings of the International Conference on Learning Representation (2018)
50. Zhul, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. In: Technical report (2002)
51. Ziang, B., Zhang, Z., Lin, D., Tang, J., Luo, B.: Semi-supervised learning with graph learning-convolutional networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)